

2021

Tracing Learning Environment in Java Programming Language

Alhammad, Sarah

<http://hdl.handle.net/10026.1/17208>

<http://dx.doi.org/10.24382/1061>

University of Plymouth

All content in PEARL is protected by copyright law. Author manuscripts are made available in accordance with publisher policies. Please cite only the published version using the details provided on the item record or document. In the absence of an open licence (e.g. Creative Commons), permissions for further reuse of content should be sought from the publisher or author.

COPYRIGHT STATEMENT

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's prior consent.

Copyright © 2021 Sarah Alhammad



UNIVERSITY OF PLYMOUTH

TRACING LEARNING ENVIRONMENT IN JAVA PROGRAMMING LANGUAGE

by

Sarah Alhammad

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering, Computing and Mathematics

April 2021

Acknowledgements

First and foremost, all praise and gratitude is due to Allah the All-Compassionate and All-Merciful for giving me the potential and patience to persevere and reach this stage of my PhD research, in particular. Without Him, I would not have achieved anything.

I also owe a debt of gratitude to my beloved parents for their considerable encouragement and support, and passionate love and prayers for my success. Any success that might be resulted, hopefully, should help me make them proud and happy of me. May Allah reward them the best.

My unreserved love, thanks and appreciation must go to my husband (Ahmad) and children (Anoud, Abdullah, and Danah) who have been very patient, understanding, and inspiring to me throughout the PhD journey . May Allah bless them.

I should not forget my dear siblings who have been always supportive to me whenever I am in need and without any reservation. Many thanks to them.

This thesis would not have been completed on time without the invaluable guidance, wholehearted support, timely feedback and utmost professionalism from my Director of Studies Dr. Shirley Atkinson . I would like to express my special thanks and admiration to her . It has been really a pleasure and an incredibly rewarding experience to work with her and I am looking forward to continue doing so in future. I also wish to thank my second supervisor Dr. Liz Stuart for her time and efforts in making the PhD journey easier and better.

I acknowledge with grateful the government of Saudi Arabia, Ministry of Education, Princess Nora Bint Abdulrahman University , for granting me the scholarship and sponsoring my undertaking of this PhD programme.

Lastly, I would like to thank Plymouth University and special thanks to my colleagues and friends at the Centre for Security, Communication and Network Research.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

The following courses organised by University of Plymouth were attended: Unconscious bias, Diversity in the workplace, GDPR and information security, Health and safety, Overview of Library Services & Recourses for Researchers, and Research methods - STEMMD disciplines.

Relevant scientific seminars and conferences were attended at which work was often presented and several papers were published and prepared for publication in the course of this research project.

Publications:

- 1) Alhammad, S., Atkinson, S., and Stuart, L. (2016). The role of visualisation in the study of computer programming . Proceedings of the 27th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2016),Cambridge, UK, 2016, pp. 5-16.

url : <http://www.ppig.org/library/paper/role-visualisation-study-computer-programming>

PEARL:https://pearl.plymouth.ac.uk/bitstream/handle/10026.1/8051/paper_the%20role%20of%20visualisation%20in%20the%20study%20of%20computer%20programming.pdf?sequence=1&isAllowed=y

- 2) Alhammad, S., Atkinson, S., and Stuart, L. (2018). Overview of using visualisation in programming learning. International Conference on Recent Advances in Computer Science and Information Technology (ICRACSIT), Dubai, UAE, 2018, pp. 54-58.

PEARL: https://pearl.plymouth.ac.uk/bitstream/handle/10026.1/12492/paper_sarah_alhammad_ICRACSIT.docx?sequence=1&isAllowed=y

- 3) Alhammad, S., Atkinson, S., and Stuart, L. (2018). Overview of using visualisation in programming learning. International Journal of Advances in Electronics and Computer Science, Vol. 5, Issue 7, pp. 20-24.

DOI: <http://iraj.doionline.org/dx/IJAECs-IRAJ-DOIONLINE-12898>

Word count of the main body of thesis: 45,369 words

Signed.....Sarah.....

Date.....27/ 4 / 2021.....

Abstract

Tracing Learning Environment in Java Programming Language

Sarah Alhammad

The visualisation approach is one of the programming learning styles that has been taken into account in programming education. A collection of visualisation tools has emerged with the aim of assisting novice programmers in learning how to program. Each tool has its own set of features that may or may not be helpful in gaining a better understanding. The methods that we used in this study are focused on using memory referencing and visualisation to clarify what happens during individual program statement executions. Understanding the efficacy of current instructional resources is a critical component of gathering students' requirements and needs for future improvement.

The “Tracing Learning Environment” (TLE) is developed for novice programmers to help them trace the sequence of execution of a software program and the reserved place of data in the memory. The framework relies on using visualisation as the programs are run and to show the effect of each statement in the code. It provides an environment for learners to see what happens to the data while running the program.

The specification of the TLE draws largely on research regarding the role of visualisation in teaching computer programming and associated literature on tools to support learning programming.

The TLE framework has been evaluated by conducting an empirical study using a mixed-method approach with novice and expert participants. The study has included surveys, focus groups, and semi-structured interviews. Student performance was measured before and after using the visualisation tool and compared with a control group who participated in a standard teaching session only.

Early findings highlighted the need to visualise the control of the execution of code, evaluation of expressions, represent the class hierarchy along with the importance of a good interface/usability of the tool and to consider the programming languages supported. The evaluation findings are in line with the literature surrounding the benefits of using visualisation in learning to program. The findings found visualisation increased the students' performance and confidence. When compared to the regular lab activities, the visualisation contributed to better understanding and support for learning to program.

Contents

COPYRIGHT STATEMENT.....	I
AUTHOR'S DECLARATION	IV
CONTENTS	3
LIST OF FIGURES	6
LIST OF TABLES	7
CHAPTER 1 INTRODUCTION	8
1.1 INTRODUCTION	8
1.2 STATEMENT OF THE PROBLEM	9
1.3 PURPOSE OF THE STUDY	10
1.4 RESEARCH QUESTIONS	10
1.5 RESEARCH HYPOTHESES	11
1.6 THESIS ORGANISATION.....	11
CHAPTER 2 PROGRAMMING LEARNING SYSTEMS.....	13
2.1 INTRODUCTION	13
2.2 THE EVOLUTION OF LEARNING TO PROGRAM FROM THE PERSPECTIVE OF PROGRAMMING LEARNING SYSTEMS	13
2.3 COMPUTER SCIENCE IN SAUDI ARABIA.....	17
2.4 NOVICE PROGRAMMERS.....	18
2.5 E-LEARNING OR BLENDED LEARNING	19
2.6 E-LEARNING AND VISUALISATION	20
2.7 STRENGTHS OF USING SYSTEMS AND APPLICATIONS IN LEARNING PROGRAMMING	21
2.8 WEAKNESSES OF USING LEARNING SYSTEMS AND APPLICATIONS IN LEARNING PROGRAMMING	23
2.8.1 <i>The difficulty of choosing the right programming language</i>	24
2.8.2 <i>Explain errors to individual novice programmers</i>	25
2.8.3 <i>The increased complications of learning</i>	25
2.8.4 <i>Tutor preparation</i>	26
2.8.5 <i>Information retrieval</i>	26
2.8.6 <i>Other systems that require a lot of practice</i>	26
2.8.7 <i>Increased number of instructional texts</i>	26
2.9 CONCLUSION	27
CHAPTER 3 PEDAGOGICAL THEORY: THRESHOLD CONCEPTS.....	29
3.1 INTRODUCTION	29
3.2 CHARACTERISTICS OF THRESHOLD CONCEPTS	29
3.3 THRESHOLD CONCEPTS IN COMPUTER SCIENCE	30
3.4 PROPOSED MODELS FOR THRESHOLD CONCEPTS	37
3.5 CONCLUSION	38
CHAPTER 4 VISUALISATION IN PROGRAMMING LEARNING	39
4.1 INTRODUCTION	39
4.2 MEMORY REFERENCE VISUALISATION.....	39
4.3 TOOLS TO SUPPORT PROGRAMMING LEARNING.....	40
4.3.1 <i>The BlueJ tool</i>	44
4.3.2 <i>Jeliot 3</i>	45
4.3.3 <i>DrJava</i>	46
4.3.4 <i>ProfessorJ</i>	48

4.3.5	<i>WebTasks</i>	50
4.3.6	<i>The Alice tool</i>	51
4.3.7	<i>The ANIMAL system</i>	53
4.3.8	<i>Visual Logic</i>	54
4.3.9	<i>Online Python Tutor</i>	55
4.3.10	<i>The Visualiser</i>	56
4.4	VISUALISATION EVALUATION	57
4.4.1	<i>The methodology used to evaluate the tools</i>	58
4.4.2	<i>The measured factors</i>	61
4.4.3	<i>Programming topics</i>	64
4.5	CONCLUSION	66
CHAPTER 5 RESEARCH METHODOLOGY.....		68
5.1	INTRODUCTION	68
5.2	RESEARCH METHODOLOGY	69
5.2.1	<i>Deductive and inductive approaches</i>	72
5.2.2	<i>The worldview (epistemological) consideration</i>	72
1.	Positivism.....	72
2.	Constructivism	73
5.3	QUANTITATIVE METHOD	75
5.3.1	<i>Survey with experiment</i>	76
5.3.2	<i>Survey measurements</i>	77
5.3.3	<i>Data collection</i>	80
5.3.4	<i>Data analysis strategy</i>	82
5.4	QUALITATIVE METHOD	82
5.4.1	<i>Grounded Theory</i>	83
5.4.2	<i>Semi-structured interviews</i>	87
5.4.3	<i>Focus groups</i>	89
5.4.4	<i>Data analysis strategy</i>	89
5.5	ETHICAL ISSUES	91
5.6	CONCLUSION	93
CHAPTER 6 INVESTIGATING THE ROLE OF VISUALISATION IN THE STUDY OF COMPUTER PROGRAMMING		94
6.1	INTRODUCTION	94
6.2	DATA COLLECTION	94
6.2.1	<i>Interviews</i>	94
6.2.2	<i>Tool selection</i>	97
6.2.3	<i>Study tasks (threshold concepts)</i>	97
6.3	DATA ANALYSIS	100
6.4	FINDINGS	100
6.4.1	<i>Controlling the execution of the code</i>	101
6.4.2	<i>Availability of the tool</i>	102
6.4.3	<i>Error explanation</i>	103
6.4.4	<i>Interface/usability of the tool</i>	104
6.4.5	<i>Programming languages supported</i>	104
6.4.6	<i>Expression evaluation</i>	105
6.4.7	<i>Representation of class hierarchy</i>	106
6.4.8	<i>Maintaining an event history</i>	106
6.4.9	<i>Tool comparison</i>	107
6.5	VISUAL CODE FLOW TOOL	109
6.6	VISUAL CODE FLOW COMPONENTS	112

6.7	CONCLUSION	118
CHAPTER 7	EVALUATION OF THE USE OF VISUALISATION IN PROGRAMMING LEARNING ..	120
7.1	INTRODUCTION	120
7.2	OBJECTIVES.....	121
7.3	EVALUATION METHOD	121
7.3.1	<i>Selection of population</i>	123
7.3.2	<i>Instrument</i>	124
7.3.3	<i>Programming subjects</i>	124
7.3.4	<i>Quantitative method</i>	125
7.3.5	<i>Experiment preparation</i>	129
7.3.6	<i>Experiment procedures</i>	130
7.3.7	<i>Focus groups</i>	131
1.	Focus group preparation.....	131
2.	Focus group procedures	132
7.3.8	<i>Interviews</i>	134
1.	Interview preparation.....	134
2.	Interview procedures	134
7.4	FINDINGS	137
7.4.1	<i>Survey results</i>	137
1.	Students' difficulties.....	150
2.	150
2.	Students' satisfaction	151
7.4.2	<i>Qualitative findings</i>	155
1.	Focus group data.....	155
2.	Interview data.....	162
7.5	CONCLUSION	165
CHAPTER 8	CONCLUSIONS AND FUTURE WORK	166
8.1	INTRODUCTION	166
8.2	CONTRIBUTIONS AND ACHIEVEMENT OF THE RESEARCH	166
8.3	THE RESEARCH QUESTIONS	167
8.4	LIMITATIONS OF THE RESEARCH	169
8.5	FUTURE RESEARCH	169
8.6	THE IMPORTANCE OF VISUALISATION IN PROGRAMMING LEARNING.....	170
8.7	RESEARCH DISSEMINATION	171
REFERENCES	172
APPENDICES	182
APPENDIX A-	ETHICAL APPROVAL LETTER AND FORM- CONSENT FORM- INFORMATION SHEET (DATA COLLECTION)	182
APPENDIX B-	QUESTIONS FOR SEMI-STRUCTURED INTERVIEWS - STUDENTS AND NOVICES PROGRAMMERS	198
APPENDIX C-	VISUAL CODE FLOW.....	203
APPENDIX D-	ETHICAL APPROVAL LETTER AND FORM- CONSENT FORM- INFORMATION SHEET (EVALUATION OF VISUALISATION).....	204
APPENDIX E-	EVALUATION SURVEY- FOCUS GROUP QUESTIONS- EXPERT INTERVIEWS QUESTIONS	228
E-1	<i>Pre-survey for control and visualisation group</i>	228
E-2	<i>Post-survey for control and visualisation group</i>	237
E-3	<i>Questions for Focus Groups</i>	249
E-4	<i>Questions for semi-structured interviews- expert interview</i>	253
APPENDIX F-	INTERVIEWS AND FOCUS GROUPS TRANSCRIPT.....	256
F-1	<i>Interviews with programming students in the data collection Phase</i>	256
F-2	<i>Interviews with Experts in the Evaluation Phase</i>	286
F-3	<i>Focus Group Discussion with The Students In The Evaluation Phase</i>	292

List of Figures

Figure 1. Memory diagram.....	33
Figure 2. BlueJ.....	37
Figure 3. Jeliot	39
Figure 4. DrJava.....	40
Figure 5. ProfessorJ and DrScheme.....	42
Figure 6. WebTasks..	43
Figure 7. Alice..	45
Figure 8. ANIMAL system.....	46
Figure 9. Visual Logic.....	47
Figure 10. Online Python Tutor.....	48
Figure 11. The Visualiser tool.	49
Figure 12. Methodology and Experiments Diagram.....	62
Figure 13. Visual Code Flow tool interface	106
Figure 14. Visual code flow interface for the programming problem of calling method and passing parameters.....	110
Figure 15. Visual code flow interface for the programming problem of defining and using classes and objects.....	110
Figure 16. Visual code flow interface for the programming problem of class inheritance.....	111
Figure 17. Experiment components	115
Figure 18. Case A - students' confidence regarding general topics in programming	138
Figure 19. Case B – students' confidence regarding general topics in programming	141
Figure 20. Case C - students' confidence regarding general topics in programming.....	142
Figure 21. Satisfaction level of the participants regarding the use of the Visual Code Flow tool in the Case A problem.....	145
Figure 22. Satisfaction level of the participants regarding the use of the Visual Code Flow tool in the Case B problem.....	146
Figure 23. Satisfaction level of the participants regarding the use of the Visual Code Flow tool in the Case C problem.....	147

List of Tables

Table 1. The five phases of research in threshold concepts	31
Table 2. Tools/models supporting the visualisation.....	41
Table 3. Coding framework table for interviews	90
Table 4. Visual code flow components	113
Table 5. Number of students who participated in the experiment	123
Table 6. Arrangement of the students in the experiment.....	129
Table 7. Number of groups/participants in the focus group	131
Table 8. Expert interview background questions	135
Table 9. Statistical results for the visualisation group.....	137
Table 10. Statistical results for the control group	138
Table 11. Comparison between the visualisation and control groups.....	138
Table 12. Case A - Question topics	139
Table 13. Case A - Students' level of confidence in solving the problem	139
Table 14. Case B - Question topics	140
Table 15. Case B - Students' level of confidence in solving the problems	141
Table 16. Case C - Question topics	142
Table 17. Case C – Students' level of confidence in solving the problems	143
Table 18. Summary of content - general comments in a focus group for tool usage.....	156
Table 19. Case A - focus group summary of content-specific comments.....	158
Table 20. Case B - focus group summary of content-specific comments	160
Table 21. Case C - focus group summary of content-specific comments	161
Table 22. Summary of content comments for the experts' interviews.....	163

Chapter 1 INTRODUCTION

1.1 Introduction

Programming is challenging, and all learners need to acquire specific skills (Milne and Rowe, 2002). Several researchers have investigated the difficulties surrounding learning and teaching programming. For instance, Milne and Rowe surveyed students from Dundee University to uncover common problems and major areas of difficulties students and tutors encounter in an object-oriented programming course. They found the most problematic areas involved using pointers and memory-referencing concepts and that this was partially because of the students' inability to absorb what was happening in the program memory during program execution (Milne and Rowe, 2002). Siti Rosminah and Ahmad Zamzuri (2012) found the same problem in their study that showed that students' difficulties involved understanding the role of variable position in computer memory. Husain et al. (2013) found that students had difficulty in writing programs and that even students with high grade point averages (GPAs) had trouble applying the programming concepts they learned in lectures. The two most challenging factors for those students were in the design and implementation phase when they started writing their programs.

The main challenge facing novice programmers is understanding the impact of each execution statement in the program code (Evangelidis et al., 2001). There is a gap between how a student may represent the problem/solution in their mind and how it will be represented in the computer. Therefore, students might not predict the results of each instruction correctly. In predicting the output of each program instruction, it becomes difficult to know when there is insufficient programming knowledge (Fitzgerald et al., 2008).

The standard teaching of programming languages relies on lectures and labs. This is static programming and does not motivate students to think or dynamically interact with programs (Law et al., 2010; Gomes and Mendes, 2007; Al-Imamy et al., 2006). Teachers focus on syntax, rules and grammar. They often teach students how to define the identifiers, the different data types, the operators and operands and how to construct different expressions, such as conditional statements

and loops. Programming teachers sometimes follow these strategies because they lack experience or have limited lecture time. For this reason, teachers tend to focus on the format of the program rather than on how to solve problems through programs. Students have to know how to write a program as a collection of expressions in a well-structured form. The challenge in learning programming languages is knowing how the program works and how to structure it (Ala-Mutka, 2004).

The visualisation method in learning programming includes showing or visualising the impact of the programming code on the memory, which allows novices to ‘see’ what happens during program execution. Use of the visualisation method was introduced in learning programming as Memory Transfer Language (MTL) (Mselle, 1989). Since the invention of the visualisation method, researchers have investigated the benefits of using this approach in learning programming. Visualisation systems have been developed to aid students’ understanding, and some of these systems have been evaluated by conducting different experiments, including BlueJ (Hagan and Markham, 2000; Sun 2010), AnimPascal (Satratzemi et al., 2001), Jeliot3 (Moreno et al., 2004), the Memview tool (Gries et al., 2005), Turtlet (Kasurinen et al., 2008), BackStop (Murphy et al., 2008) and Alice (Salcedo and Idrobo, 2011). The evaluation of visualisation and its tools is discussed extensively in Chapter 4. The present chapter highlights the main research aims and objectives of the thesis and describes its structure.

1.2 Statement of the Problem

The visualisation method is one of the learning styles in programming and has been implemented in programming education. Visualisation tools have emerged with the aim of supporting teaching novice programmers how to program (Dekson et al., 2009). Each tool has features that may or may not be useful for better understanding. The tools considered in this research are based on using memory referencing and visualising in explaining what happens in individual executions of program statements. Understanding the effectiveness of the existing educational tools is an important part of learning students’ requirements and needs for further improvement.

1.3 Purpose of the Study

The research aim is to develop a new approach that supports learning programming. The approach explores the effectiveness of using MTL programming. A memory reference or MTL is defined as the language or device programmers use to describe the effect of code lines on computer memory (RAM). The current study will evaluate existing models to discover the needs of novice programmers and understand their issues. The results of this evaluation will be used to develop a new conceptual framework to determine whether the concept can be fully or more effectively utilised to support programming learning.

The study will try to determine and elucidate the strengths and weaknesses of programming learning systems, particularly visualisation systems and current tools that support programming education, and to characterise existing visualisation methods and evaluate the necessary features required for the new framework. The study will explore theories of education to discover the challenges faced by novice programmers; develop and implement the proposed framework based on existing tools and the needs of novice programmers; and evaluate the proposed framework by seeking beginners' comments and expert opinions to ensure it is appropriate for the requirements of novice programmers.

Thus, the study will achieve the following:

1. To investigate the effect of using visualisation by tracing the program's instructions on the students' performance.
2. To investigate the effect of using visualisation by tracing the program's instructions on students' understanding of general programming concepts.
3. To investigate the effect of using visualisation by tracing the program's instructions on students' confidence in solving programming problems.

1.4 Research Questions

This study will be able to answer the following questions:

1. What is the impact of using program visualisation on students' performance while they are tracing the program?
2. What is the impact of using program visualisation on students' comprehension of the general programming concepts?
3. What is the impact of using program visualisation on students' confidence while solving programming problems?

1.5 Research Hypotheses

- The first main hypothesis is that a significant effect of Student's performance improved by using the visualisation of the tracing of the program's instructions.
- The second main hypothesis is that student's understanding of general programming concepts improved when using visualisation of tracing program's instructions.

The aim of the study is to evaluate existing models to discover the needs of novice programmers and to understand their issues. The results of this evaluation will be used to develop a new conceptual framework to determine whether the concept can be fully or more effectively utilised to support programming learning. The study hypotheses will be discussed based on what has been seen and observed using a research theoretical framework.

1.6 Thesis Organisation

The thesis is organised as follows. Chapter 2 provides background on the programming learning systems used in programming education. The chapter discusses the integration and evolution of programming learning systems. It also describes the strengths and weaknesses that exist in programming learning systems that should be considered in the new framework.

Chapter 3 discusses important concepts in learning programming with a focus on where students predominantly encounter difficulties. The chapter describes the characteristics of threshold concepts and what the threshold concepts are among computer science (CS) concepts.

Chapter 4 discusses the memory reference visualisation method. It includes a literature review of previous studies conducted in the field and a discussion of some of the models that have been

invented and of the tools supporting visualisation. Chapter 5 discusses the methodology and research methods used. Chapter 6 presents the implementation of the research conducted to investigate the role of visualisation in programming learning and the development of the tracing learning environment (TLE) (visual code flow) tool and how it was developed based on findings. Chapter 7 presents the empirical study to evaluate the TLE framework using the visualisation method in programming learning and how this affected students' performance and confidence in programming. Chapter 8 concludes the thesis by highlighting the study's contribution and the importance of the research. It presents the limitations and suggestions for future research. The chapter discusses the importance, dissemination and impact of the research. The appendices and references are located at the end of the thesis.

Chapter 2 PROGRAMMING LEARNING SYSTEMS

2.1 Introduction

Before the development of programming learning systems, traditional means of teaching programming entailed a teacher giving detailed examples to students who were required to internalise the concepts and follow the given examples linked to a programming language. Later, radio and television channels developed such that lecturers could teach using these mediums. However, over time certain aspects of programming education changed, such as the learning environment and the learning materials. Programming education has become more of an online concept than when it was being taught in classrooms (Robins et al., 2003). How applications and systems in the learning of programming have evolved over time is discussed in Section 2.2. This includes e-learning systems that are part of programming education in general, where this thesis is located. The advantages and disadvantages of using the various systems in programming education are also discussed.

2.2 The Evolution of Learning to Program from the Perspective of Programming Learning Systems

E-learning became a common term in 1999 when it was used to refer to learning done over the Internet. Before the advent of e-learning, B.F. Skinner developed a machine that could connect one programmer with a large number of students (Skinner, 1958). Skinner had advanced an innovation by Sydney Pressey, who wanted to create an automatic teacher, an idea that never succeeded. This was the first time e-learning was used in programming, as the distance between the teacher and the learner did not matter. The machine was fitted with tests and learning items, which rewarded students as a consequence of the learning process. It required that the novice programmers be able to fill in blank spaces, and if the spaces were correctly filled in positive reinforcement would follow. There were multiple choice questions that had four possible answers the learner could choose from. The teaching machine was preferred for distant novice programmers, as the physical

presence of the teacher was not necessary. It was also less labour-intensive and therefore gained popularity in programming learning.

Skinner's programmed instructions became widespread in other fields of learning. The development of e-learning was enhanced by advancements in communication that occurred in the 1960s, which allowed people to effectively communicate irrespective of distance and time differences. Later in the 1960s, Don Bitzer advanced Skinner's efforts by inventing an automated teaching activity called Programmed Logic for Automatic Teaching Operations (PLATO). PLATO was more acknowledged than the Internet and the bulletin boards used today (Bitzer et al., 1967). The applications preceding PLATO gave birth to other applications that were adopted by academic institutions, including those that were teaching programming.

In the 1970s, the Basic Instructional Program system resulting from the intelligent tutoring system also allowed for individualised instruction and gave novice programmers a chance to learn at their own pace to better understand the programming content (Durzo, 1978). Other systems have also taken programming education to another level, including LOURA, which was developed at University of Caen. LOURA's distinctive feature is that it can represent information in the form of a graph, making it easy to understand. This is a better way of presenting information to novice programmers than complicated data and figures (Adam and Laurent, 1980). Another system is the PROUST system, which was developed at Yale University. The PROUST system is given credit because of its ability to simplify bugs for learners. Researchers continue to pursue the development of more programming systems that can make programming education interesting and less difficult, given the variety of systems that novice programmers currently have to blend (Zelhart and Wallingford, 1994).

Systems and tools in e-learning have been integrated into many fields of education, including programming. Approximately ten million courses are taught online, and there are approximately 700 e-learning companies in the US (Capper, 2001).

New technology has resulted in the incorporation of different applications to facilitate programming education. Studies estimate that the incorporation of applications began in the 1980s, and this has increased and evolved drastically over the years (Martín-Blas and Serrano-Fernández, 2009). It is fundamental to understand that the applications in programming learning are facilitated by computers and that technology has advanced to the extent of having better applications. Improved computer technology has resulted in increasingly sophisticated applications that have ensured programming learning is effective (Harris et al., 2009).

This evolution enabled the involvement of more students in classwork and learning programs. Further, the advancement of learner-centred programs has been facilitated using video tutorials that guide students in understanding more about programming (Harris et al., 2009). Some of the applications in use include Encode, Udacity, Tynker, Khan Academy and code hub. These have become more advanced, thus facilitating the solving of complex programs. The applications used in the 1980s facilitated programming education in fewer ways compared to the current applications that incorporate videos, tutorials, consultation interfaces and exercises to enable learners to learn the skills and steps required in programming.

Subsequently, the intelligent tutoring system for programming was developed in association with Stanford University. This system had an advantage over the other systems, which to some extent could not provide learners with a meaningful learning experience, mainly because they could not link a learner's prior knowledge to what they presented. This was contrary to Skinner's belief that learning should be meaningful to learners by making what they already know useful in a learning situation. The intelligent tutoring system for programming gave learners the opportunity to link what was being presented in class to what they already knew about programming (Yang, 2010).

As mentioned, the involvement of applications has changed drastically, resulting in the improvement of programming education. Nonetheless, it is important to acknowledge that times have changed, and there is a need for more sophisticated applications in educating learners about programming (Tamim et al., 2011). E-learning in programming is essential because of the increase

in the number of programmers worldwide. E-learning reaches a wider range of people compared to conventional classrooms (Ghirardini, 2011).

For instance, the probability of making learners problem solvers is increased, as learners understand each step involved in programming with less supervision from educators (Behera et al., 2013). This is possible by incorporating sophisticated applications in the system. More applications have incorporated an interface for tutors to ensure their involvement, thus facilitating holistic programming education.

In the past, learning used to incorporate fewer applications to ensure that students learned different applications in programming education. However, this has changed with the increased number of applications that can facilitate programming learning in schools. In addition, the applications related to programming have evolved in different ways, such as in their paradigms and categories. In the past, all the information was present in a single application without categorisation based on the learning level of the learner or the examples and topics covered. This has changed in recent years, as some applications have categorised programming tutorials, such as beginner-level tutorials. According to Behera et al. (2013), applications have changed drastically with the introduction of different programming applications for learners; therefore, beginners have a better way of starting to learn programming compared to the past.

In 2016, the worldwide revenue for self-paced e-learning products and services was \$46,674.7 million US (Back and Dietrich, 2017). The evolution of e-learning research has also developed quickly in the last four years. According to Rodrigues et al. (2019), three common factors in e-learning have been examined—e-learning content and elements, e-learning demands and online education. The emergence of mobile learning and cloud computing have made e-learning more popular and easier to access (Rodrigues et al., 2019).

A 2020 study conducted to evaluate the evolution of e-learning and the research on the topic revealed that teaching and learning strategies in e-learning focus on assessment methodologies and self-regulated learning. Interactive learning environments is a topic of e-learning research that deals

with the communication between teachers and students. Researchers are interested in the demographics of the students engaged in online courses and in identifying barriers, potentiates and psychological profiles (Valverde-Berrocoso et al., 2020).

2.3 Computer Science in Saudi Arabia

In light of the recent worldwide scientific and technical revolution in CS, countries have been ranked in terms of their progress based on their expertise and competencies in this science. Therefore, Saudi Arabia has made a significant effort in many fields to keep pace. It has established universities and colleges specialising in this science and has supported those institutions with the appropriate budgets, laboratories and equipment. This interest and expansion in CS and other digital sciences have led to the achievement of advanced levels in all fields and disciplines and the optimal use of the outputs.

The most prominent CS disciplines are computer programming, networking, information security, computer and network security, artificial intelligence, graphics, multimedia, machine language, databases, electronic computing and big data. Because CS encompasses many disciplines, students must be made aware of them and must be given the freedom to choose so that they benefit from the various resources to increase their skills and acquire competencies at all levels. The introduction of computers to the industrial environment is particularly important in Saudi Arabia in both private and public facilities (Al-Gahtani, 2004).

At Saudi universities that offer CS, it is mandatory that all students in the various disciplines take two courses on programming. Programming I comprises the basics of programming with C or Java, such as defining variables, calling methods, mathematic and logic expressions, conditional statements and iterations. Programming II comprises object-oriented programming, such as defining classes and objects, inheritance and polymorphism. Other programming languages, such as web programming languages, are be taught in some computing disciplines but not all. Programming I and Programming II courses have three hours of lectures and two hours of laboratory work per

week. The present study mainly focuses on the programming students in computing departments, as they are considered novice programmers (see Section 2.4).

Alakeel (2015) investigated the difficulties of learning computer programming in Saudi Arabia, research in which 90 students participated. Of the students, 79 were aged 19–23 and 11 were aged 24–28. The results showed that some difficulties might truly influence the teaching of computer programming in some areas of Saudi Arabia, including the insufficient time allotted for laboratories and tutorial sessions and the insufficient amount and quality of assignments.

Learning computer programming is not limited to a specific age group. Indeed, in some countries students are taught computer programming basics and languages in the early stages of education. However, in Saudi Arabia learning programming and programming languages begins at the university level when students choose to study one of the disciplines related to CS and software. Students start to study programming languages in the first academic year and continue until the completion of their university education.

2.4 Novice Programmers

In general, ‘novice programmers’ is a term given to beginning learners who have no programming experience. In some countries, learning programming starts very early at age five or six, while in other countries it starts at secondary school or college. Therefore, the age of the population in any study conducted on novice programmers may differ.

The present study was conducted in Saudi Arabia, where students start learning programming when they start specialising in the computing field at college from the age of 19. Learning programming is not mandatory and is not included in the school curriculum at any stage, and there is no research on why programming has not been taught at an early age. The author speculates that this is because teaching English, which is used in programming, also does not start at an early age. Recently, there has been an effort to introduce programming to children aged 7–10 using Scratch, a drag and drop tool that does not require knowledge of English. However, these efforts have only been made at

small private institutions and some international schools, and learning programming is still not mandatory for all Saudi students (Al-Othman and Almawash, 2020).

For that reason, the population in the present study is comprised of programming students in their first year of college; therefore, they are considered novices. Moreover, the study considered any student who started to study any programming concepts from the beginning is a novice .

2.5 E-Learning or Blended Learning

E-Learning is a tool that supports the educational process and transforms it from the stage of initiation to the stage of creativity, interaction and skills development. It combines all forms of electronic teaching and learning where the latest methods are used in the fields of education, publishing and entertainment by adopting the use of computers, storage media and networks. The rapid transfer of technology has led to the emergence of new patterns of learning and teaching, which has further entrenched the concept of individual or self-learning. E-learning is one of these evolving patterns of so-called distance learning in general and in computer-based learning in particular. E-learning relies mainly on computers and networks for the transfer of knowledge and skills. Its applications include web learning, computer learning, virtual classrooms and digital collaboration. Online lesson content, audio, video and CDs are offered (Aparicio et al., 2016).

E-learning and e-learning techniques are utilised in teaching in several ways, such as to supplement and support traditional learning. This can be done inside or outside the classroom. Examples of the application of e-learning before teaching is instructing students to view a particular lesson on the Internet or on a CD. Blended learning includes the integration of traditional and electronic education in the classroom or places equipped with e-learning technology and is characterised by combining the advantages of traditional and electronic education, but the role of the teacher in this case is to guide and manage the educational situation and ensure the learner has a positive role. In addition, the pure model uses e-learning as an alternative to traditional learning so that learning can be done anywhere and at any time by the learner. The network acts as a primary intermediary to provide the entire learning process; an example of its application is independent self-study (students

studying e-course individually). Studying or completing a project can be done using participatory e-learning tools, such as chat rooms and forums (Mayer, 2017).

E-learning for programming education has also offered another avenue of providing learning materials for novice programmers and providing a platform for tutors to access their novice programmers in various parts of the world. Over time, e-learning has proved to be of assistance to novice programmers, as they can access even more materials from wherever they are. For instance, e-learning provides novice programmers the opportunity to access tool books, plugins, workshops, references and conferences. These services are aimed at making e-learning easy and accessible for all interested novice programmers. These services have grown incrementally in recent years, as novice programmers have been able to access all the information they need for programming. This has attracted a large number of e-learners, making the learning of programming an option for many people. E-learning has incorporated all the features of the face-to-face learning process, thus making e-learners comfortable, as they feel equal to their peers. Further, e-learning provides platforms for novice programmers to review what they have previously learned and provides answers to their questions about whatever they fail to understand. Other systems allow tutors to monitor their pupils' behaviour, enabling them to identify the novice programmers' areas of weakness, making the learning of programming even more targeted (Anderson and Skwarecki, 1986).

From the previous description of the features of e-learning and blended learning, we can say that this research focused on using programming visualisation, which can be utilised in both e-learning and blended learning. The utilisation of some computerised educational tools or software, such as visualisation, can facilitate e-learning and blended learning if used as self-learning techniques and can help tutors present lessons.

2.6 E-learning and Visualisation

The present study focused on using visualisation to facilitate programming education. Memory reference visualisation is defined by Mselle (1989) as a language or device used by programmers to

describe the impact of code lines on computer memory (RAM). Visualisation is discussed in depth in Chapter 4; however, this section will discuss why visualisation is considered to fall under e-learning. The visualisation method can be used in e-learning and blended learning to facilitate the learning process. Based on the information in Section 2.5 regarding e-learning and blended learning, there is a clear relationship between e-learning and visualisation. The use of visualisation techniques in the learning process is not new but rather has been used in relation to maps and drawings before our world became more oriented to the use of multimedia in education. People in the current era prefer learning through visual information more than textual information and through graphs and other formats that are more accessible than block text. When using any of the visualisation methods in e-learning, it is necessary to ensure that the presented visualisation is clear and coherent and that it is not in excess of the need and will add value to learning (Aytekin, 2019; Conti et al., 2019).

2.7 Strengths of Using Systems and Applications in Learning Programming

Technological advancement has resulted in sophisticated applications aimed at programming education. According to Resnick et al. (2009), the positive impacts of applications in learning programming are considered strengths. First, incorporating applications has made it possible for learning to become more learner-centred and skill-based. Beatty (2013) argues that the paradigm shift in education insists learners be the centre of attention to facilitate programming learning. In the past, teachers concentrated on teaching learners about programming. This has changed, as tutors now refer learners to applications such as Khan Academy and Python Tutor to master the skills required for programming. This has changed drastically in recent years, as more applications involve learners. Currently, teachers can assign tasks to groups where learners are asked to develop programs and the applications enable the learners to know some of the steps, such as debugging (Schroeder et al., 2010). The tasks ensure learners are more involved in programming, thus making them more engaged in learning and mastering skills compared to the past. The incorporation of

applications makes it possible for learners to become problem solvers, as they can address more issues independently. The use of applications that offer tutorials using a step-by-step approach to solve a particular problem ensures that learners can develop solutions on their own.

Further, programming education is research-based; hence, there is a need to develop skills and knowledge regarding how to handle research. Applications that offer guidelines create a platform on which learners can develop skills and knowledge about how to solve programming problems (Schroeder et al., 2010). The incorporation of tutorials and tests after using an application increases the chances of a student acquiring more knowledge and skills, thus advancing the learning of programming. The applications are more sophisticated, which makes it easier to solve some of the complex programs, especially those relating to mathematics or research (Blikstein, 2011). Programming students can solve complex programs in a short time, which supports time management and motivates students to tackle complex programming problems.

The applications are an advantage to the teacher, who can assess a student's progress online, thus facilitating immediate feedback. Some applications include tests at the end of tutorials, which the teacher can follow up on to determine the areas or steps that challenge students. Providing feedback at an early stage makes learning programming more successful, as educators can assist learners resolve any problems (Hatziapostolou and Paraskakis, 2010). In addition, some applications increase the interaction between learners and tutors, such as Solo Learn, Khan Academy and Programming Hub. Studies have shown that parents' involvement in education encourages learners to do better. The ultimate goal of learning programming is to ensure that students achieve more by mastering all the basic steps in programming.

Using systems such as the applications and tools in programming education helps students develop diverse skills that are practical and can be applied in the job market, which demands new technologies, coding skills and general programming knowledge (Welsh et al., 2003). Some e-learning methods may also help individuals gain self-directed learning skills and increase their capabilities in the workplace, where they can work without supervision. Programming learning

systems are custom designed and thus are flexible and easily customisable to suit learners' needs. The different systems of e-learning also offer the possibility of monitoring learners while they are working on problems. Consultation in a difficult case is much easier and more accessible when using computer-based programs, as the tutor is only a click away. Novice programmers can comprehend easily and retain the knowledge because of the interactions and simulations involved in various e-learning methods. In addition, a major strength of using the different e-learning methods is that the programs are always on and therefore do not interfere with learners' other programs. They are manageable, and the learner can be trained at any time of the day. E-learning methods are cost-effective and save on time and transport costs, among other resources (Welsh et al., 2003).

In organisations, the use of e-learning programs is manageable and measurable, as the Internet can show what learners have been able to learn and the difficulties they experienced in completing the exercises. According to Franzoni et al. (2008), students who attend classes do not discover their major strengths and weaknesses because they do not have the tutor's full attention. With e-learning, the tutor has direct interaction with specific learners. Thus, it is easier to identify one's weaknesses and strengths so that necessary action can be taken. E-learning methods also offer a good platform for evaluating employees in a business situation, as computer-based programs display the scores from various tests presented to the learner. The use of analogies and practical examples offered in e-learning helps learners develop ideas that can be applied in the real world and ensures that novice programmers do not blindly fill their minds with theories and formulas. Therefore, e-learning methods allow better understanding and provide a better learning environment for students than that provided by attending class or relying on a human tutor.

2.8 Weaknesses of using learning systems and applications in learning programming

Learning programming using applications and tools has many advantages. However, the incorporation of applications has a drawback in programming education. For example, some students may overly depend on the applications to solve solutions or complete their tasks, thus making them incompetent when faced with challenges (Le May et al., 2008). The use of

applications to solve programming challenges does not help students, as it makes them dependent to the extent that they have to refer to the applications to ensure they are confident in their work. Moreover, applications can make students less motivated or dedicated to programming, as they have found an easier way of completing their assignments and acquiring programming skills. According to Le May et al. (2008), applications require close supervision. For example, long programming projects require that tutors verify different steps before learners proceed. These applications require the teacher's involvement in order for students to meet deadlines, as less supervision can result in inefficiency, which impacts programming education negatively. More drawbacks are discussed in the next section.

2.8.1 The difficulty of choosing the right programming language

Many authors have argued about which programming language should be studied first (Meyerovich and Rabkin, 2013; Parker et al., 2014; Law et al., 2010). According to some, the first code or language chosen is not important in programming education. Many systems of programming education have been introduced, leaving tutors with the challenge of choosing which system to use. Some tutors choose to blend these systems, making it even more complicated for novice programmers who know about the existence of other systems. This may leave novice programmers dissatisfied with the content they are given. There are specific languages for different levels of novice programmers. For example, Alice for beginners and intermediate learners and Python for high-level programming. The tutor may shift from Scratch while the learner still needs it, which makes it difficult for novice programmers to understand lessons (Law et al., 2010).

Scratch and Alice are used in many different environments, such as schools, museums, community centres and homes. Alice is an innovative 3D programming platform that provides an easy environment for creating storytelling animations or playing an interactive game or video on the web. Alice is a free educational tool designed to be the first exposure for students to program objects. It allows students to learn basic programming concepts in the context of creating animated films and simple video games.

Whilst Scratch and Alice are especially used for children aged 8–16, younger children can work on Scratch and Alice projects with their parents or older siblings. Some college students use Scratch in introductory CS classes. By manipulating objects in their virtual world, students can gain experience in all programming structures typically taught in an introductory programming course (Ebrahimi et al., 2013).

2.8.2 Explain errors to individual novice programmers

With the use of many programming systems, novice programmers may get confused and require individual assistance. Unlike the traditional method of teaching programming where attending to individual novice programmers was easy, teachers may not be able to correct all the students' mistakes well. This results in a lack of proper understanding. Failure to explain errors may occur because the origin of the errors is unclear, thus allowing the errors to remain unattended. Evaluations of learners' exercises are therefore not reliable, as they reflect a sum-up image and not the exact performance of each system involved in the test. A learner may be good at one system and perform poorly on another; therefore, the strengths of novice programmers are not reflected (Law et al., 2010).

2.8.3 The increased complications of learning

Programming education is perceived as being fairly difficult for some novice programmers (Siti Rosminah and Ahmad Zamzuri, 2012). The inclusion of the many systems in learning programming can make courses sound challenging. It also makes learning cumbersome because of the many references that may leave a student wondering which system to follow. It makes the use of many systems in the learning of programming a challenge. For novice programmers who learn by themselves and who only require evaluations, learning using a few systems should be recommended for them. This is because it will simplify the learning process. Using too many systems may cause a lack of understanding (Stankov, Glavinic and Rosic, 2011).

2.8.4 Tutor preparation

Even though using a combined system in programming learning makes it easier for novice programmers to study, it is difficult for tutors. Tutors must accurately borrow from the many systems and create effective teaching materials. This is time-consuming; therefore, the materials may not be available to novice programmers at the right time. Tutors may also lack the time to include all the information from the many systems, thus destroying the purpose of using the various systems (Mayer, 2013).

2.8.5 Information retrieval

If novice programmers do not retrieve the information provided from different systems, they may not be able to get the correct information. For instance, solving programming problems by referring to different applications or learning systems will mislead novices and confuse them. The effectiveness of information retrieval needs to be tested for accuracy. There is a need for novice programmers who are provided with e-learning material to have a high-quality retrieval system, failure of which will lead to inaccurate evaluations (Mayer, 2013).

2.8.6 Other systems that require a lot of practice

Systems such as STRATEX require that novice programmers practice frequently, as the system elements are to be applied in other areas while learning programming. Therefore, it is a disadvantage for novice programmers who do not prefer the given system, as they have the opportunity to choose from various systems. As a result of the combination of various systems, regular evaluations of novice programmers concerning each system are needed to ensure they have an equal understanding of all systems (Bode and Hahn, 2015; Padilla et al., 2015; STRATEX, 1997).

2.8.7 Increased number of instructional texts

An increased number of systems requires a greater amount of learning material. Having many textbooks increases novice programmers' workload, and they may end up not completing the course. Others may partially lose interest and end up failing the programming course. Learners may

not be interested in the type of evaluation offered by the many systems, and the inclusion of many systems may make the learning process difficult. Novice programmers may lose focus because of the increased workload (Resnick et al., 2009).

2.9 Conclusion

In conclusion, irrespective of the shift of programming education from a classroom environment to a having a choice of environments, e-learning has not changed the basic concepts of programming. Generally, the inclusion of many systems in the e-learning of programming has advantages and disadvantages. Therefore, novice programmers have to choose from a long list of systems of learning programming to suit their needs. Further, tutors must determine if it is in the interest of their novice programmers to use many systems, whether in classrooms or online.

The chapter discussed the evolution of using systems and applications in programming education, which can be used as a basis to build a theory about the subject under study. The chapter also discussed the environment in which the study was conducted (Saudi Arabia) and how the term ‘novice programmers’ has been defined and formulated based on programming learning in Saudi Arabia. The chapter showed how the project work includes e-learning and blended learning.

The chapter also discussed the main aspects of the learning application, and e-learning, that influence the study. These aspects relate to the strengths and weaknesses in learning applications and systems that could be exploited in designing this study’s framework. The learning application can be developed in such a way as to avoid programming learning challenges when using the TLE framework. The challenge in developing programming learning systems is to limit the obstacles that novice programmers may face. Factors to consider include choosing the right programming language based on the students’ level and needs, decreasing the level of complications by simplifying the learning process, minimising the workload and number of instructional texts to avoid losing the students’ interest, proper data retrieval to get the correct information, reducing dependency on the tutor and shifting to self-learning.

The next chapter discusses the pedagogical theory. This includes identifying the threshold concepts, which are the programming problems novice programmers struggle with, and how the development of programming applications will help reduce the obstacles and the lack of programming comprehension.

Chapter 3 PEDAGOGICAL THEORY: THRESHOLD CONCEPTS

3.1 Introduction

Meyer and Land developed the idea of threshold concepts in 2003 whereby core concepts within any discipline are considered vital for effective mastery of that discipline (Meyer and Land, 2003). It is an approach to avoid getting ‘stuck’ in the educational process (Land et al., 2005). This chapter presents the characteristics of threshold concepts and how we can recognise them. It discusses the threshold concepts in CS and in programming that were considered when designing the framework in this study. The proposed models for threshold concepts will be investigated in this chapter.

3.2 Characteristics of Threshold Concepts

Meyer and Land (2003) described the characteristics of threshold concepts that can be used to evaluate any scientific concept, whether a threshold concept or not:

1. Transformative: alters the way students view things in the discipline
2. Integrative: links the concepts together and exposes their interrelatedness
3. Irreversible: shifts students’ perspectives
4. Troublesome: difficult for students to learn
5. Limited boundary: prevents students from crossing over the boundaries of the field concepts (Meyer and Land, 2003).

Recognising threshold concepts is not easy because what are considered threshold concepts by some people may not be by others. Davies (2003) defined alternative ways of recognising threshold concepts. Davies suggested two approaches, evaluating how two disciplines analyse the same aspects and concentrating on the differences between people’s behaviour inside and outside the community (Davies, 2003).

A study by Shinnars-Kennedy and Fincher in 2013 described a new direction for classifying threshold concepts after many researchers reached ‘dead ends’ while investigating such concepts. The new direction was to exploit expert teachers’ knowledge. They suggested using content representation (CoRe) to create pedagogic knowledge through expert teachers. CoRe captures

knowledge that helps connect the theoretical aspects to the practical ones. CoRe then examines the evidence that determines whether the concept is a threshold concept or not (Shinners-Kennedy, 2016).

Regarding threshold concepts in programming, a recent study has shown that the transformative and troublesome characteristics are vital in programming education (Yeomans et al., 2019). The study extended existing research on limit concepts in programming education by studying computer program developers and undergraduate students through the use of focus groups. The results reveal compelling evidence suggesting that concepts such as ‘classes’, inheritance’ and ‘abstract classes’, which are found in object-oriented programming, are difficult to understand. The following section discusses the threshold concepts in programming in more depth based on a chronological sequence of studies and phases.

3.3 Threshold Concepts in Computer Science

Ongoing projects have enquired into CS concepts that exhibit threshold-like attributes. An example of such initiatives is that of Boustedt et al. (2007), who outlined five phases of studies on threshold concepts (Table 1). Table 1 presents further investigations concerning finding threshold concepts in programming.

<p><u>Phase four</u> (Sanders et al., 2008)</p> <p>(Zander Carol Boustedt et al., 2008)</p>	<p>Concept maps drawn by programming students</p> <p>Informal interviews (instructors) at an international conference (June 2005, Portugal)</p>	<p>The study found that students have difficulty in distinguishing the difference between classes, objects and instances.</p> <p>The study found some threshold concepts in computing from the instructors' perspective.</p>	<p>Object-oriented programming, specifically inheritance</p> <p>33 concepts The most common were abstraction, pointers, differences between classes, objects, instances, recursion-induction procedures and polymorphism</p>
<p><u>Phase Five</u> (Moström et al., 2008)</p> <p>(Moström et al., 2009)</p>	<p>Students' biographies</p> <p>Semi-structured interviews (students) and student biographies</p>	<p>The study focused on the transformative, which led to defining the abstraction as a threshold concept in computing.</p> <p>The study examined the causes and changes of the transformative in students' experience while learning to compute.</p>	<p>Abstraction, object-oriented, code reuse and design patterns, such as unified modelling language (UML) diagrams</p> <p>Abstraction, object-oriented, code reuse and design patterns, such as UML diagrams</p>
<p><u>Further studies</u></p> <p>(Rountree and Rountree, 2009)</p> <p>(Sorva, 2010)</p>	<p>Review studies</p> <p>Not defined</p>	<p>The study discovered some of the programming concepts that can be defined as threshold concepts.</p> <p>The author suggested that educators should help students overcome thresholds when they become aware of a 'transluminal' concept. The threshold then becomes more interesting. For a programmer, a transluminal concept is to cross the program dynamic threshold by viewing program execution from a non-standing perspective.</p>	<p>Abstractions, program-memory interactions, recursion, induction, polymorphism, procedures, and the difference between objects, classes and instances</p> <p>Abstraction, object interaction and dynamic programming</p>

(Sanders and McCartney, 2016)	Review studies	The study discovered some programming concepts that can be defined as threshold concepts.	Abstraction, class declaration, class, object, instance distinction, object-oriented programming, program-memory interaction, parameter passing, recursion and pointers
(Meyer et al., 2016)	Review studies	The study suggested developing a threshold concepts framework based on the construct of integrated threshold concept knowledge.	-
(Kallia and Sentance, 2017)	The Delphi method by asking computing teachers	The study found some threshold concepts based on the computing teacher's suggestions.	Parameters, arguments, parameter passing, calling a function, control flow, abstraction, recursion, variables, variable scope, return values and procedural decomposition
(Yeomans et al., 2019)	Qualitative study (focus groups)	The study found some threshold concepts based on information from the focus groups.	Classes and data structures

Object-oriented programming is one of the threshold concepts investigated in this study. Placed in phase one by Boustedt et al. (2007) (Table 1), the work by Eckerdal et al. (2006) positioned threshold concepts in the context of CS education based on a literature review by Meyer and Land (2003). The hypothesis in the present paper suggests the need to build a visualised mental model to represent concepts of object-oriented programming. In a similar vein, Eckerdal et al.'s (2006) findings highlighted the importance of building a mental model to overcome difficulties in learning programming. Program-memory interaction is also considered a threshold concept (Salakoski,

2006). Researchers arrived at this conclusion by analysing the structure and characteristics of programs written in an exam administered to 60 students.

Phase two also revealed that object-oriented programming is a major concept regarded as difficult to understand. A set of semi-structured interviews were conducted with graduate students to gather data and address this troublesome topic in the study by Boustedt et al. (2007). The students agreed on five vexing issues—the memory model, objects, control statements, parameters and sequential thinking. The authors argued that two concepts—object-oriented programming and pointers—are ideal candidates that satisfy the criteria for evaluating an idea as a threshold concept. The evidence collected from students was quoted in the paper (Boustedt et al., 2007). McCartney et al. (2007) recommended the development of visualisation as a means of resolving miscomprehension after conducting semi-structured interviews with students to discern the approaches and strategies they use to free themselves from an impasse as they learn about computing. The researchers categorised these strategies as follows: *inputs/interactions*, which encompass reading, searching for information on the Internet and using tools; *concrete tasks/‘doing’* activities, such as practicing on the basis of examples and using visualisation and tracing methods; *high levels of learning*, such as using abstractions and relating them to real-world examples; and *using the force*, which involves relying on their instincts to solve problems.

In phase three is the study by Eckerdal et al. (2007), who conducted semi-structured interviews with students. The study focused on the liminal space and found that liminal space can be a metaphor for the learning process concept. They also found that some liminal space features can be particularly useful in computing, such as those related to abstractions.

In phase four are studies that delve into class inheritance as a threshold concept. For example, Sanders et al. (2008) explored students’ understanding of object-oriented concepts using concept maps. The researchers collected 119 maps from 107 students from six institutions in three countries from 2006–2007. The findings revealed the difficulty students encounter in identifying differences amongst *classes*, *objects* and *instances*. The researchers connected the data and behaviour to

classes, and the students mentioned inheritance but not abstractions or polymorphisms in their answers (Sanders et al., 2008).

In an experimental study, Zander et al. (2008) carried out informal interviews with various instructors from several countries during an international conference to seek their opinions on threshold concepts and relevant criteria. The instructors expressed common concerns over difficult-to-learn computing ideas, amongst which the most frequently cited were program abstractions, pointers, objects, classes, instances and recursion. The instructors claimed that some CS theories, such as constructivism, abstractionism and object orientation, relate to threshold concepts because these theories share the same characteristics with the concepts. Eckerdal et al. (2007) suggested that the ability of students to build a mental model and their misconceptions are not threshold concepts because they are not troublesome. Zander et al. (2008), however, contended that both mental models and misconceptions are transformative.

In phase five are two studies by Moström et al. (2008, 2009). In Moström et al. (2008), the authors examined the causes and changes of transformative in students' experiences as they learn computing. The researchers collected data from students in the US, the UK and Sweden through semi-structured interviews and written biographies. The findings indicated that students changed their behaviours and confidence levels after learning about threshold concepts. They began thinking in a manner similar to how computer scientists think about ideas and experienced positive change in their identity via a transition from having no experience to being involved in the learning. When the students were asked about the concepts that caused the change, some of the items they mentioned were design patterns, the Big-oh, analysis, the memory model and the halting problem (Moström et al., 2009).

Other researchers have contributed to defining computing ideas that count as threshold concepts and have developed solutions to identification issues. These studies are also summarised in Table 1. Rountree and Rountree (2009) reviewed several studies that examined whether CS ideas are threshold concepts and reported that most scholars agree on the following ideas as computing-

related threshold concepts: abstractions, program–memory interactions, recursion, induction, polymorphism, procedures and differences amongst objects, classes and instances. They also noted that what some consider a threshold concept may not necessarily be viewed as such by others. Accordingly, they suggested examining how practitioners feel about different subjects when defining threshold concepts. According to Sorva (2010), abstraction (information hiding) and object interaction can serve as ideal examples of computing threshold concepts. To this list, he added dynamic program notions (Sorva, 2010), an inclusion grounded in the argument that these notions are transformative, integrative, irreversible, troublesome and bounded. Sorva (2010) also suggested that educators should help students overcome difficulties associated with thresholds when they become aware of a ‘transluminal’ concept, which renders thresholds more interesting. For a programmer, a *transluminal concept* involves crossing a program’s dynamic threshold by viewing program execution from a non-standing perspective.

Based on the review, the most frequently investigated threshold concepts are object-oriented concepts, abstractions, control statements, arrays, recursion and passing parameters (Sanders and McCartney, 2016). Meyer et al. (2016) suggested developing a threshold concepts framework based on the construct of integrated threshold concept knowledge.

Kallia and Sentance (2017) conducted a Delphi method investigation by interviewing experts in computing teaching who have experience of over seven years. Yeomans et al. (2019) conducted a qualitative study aiming to address some of the threshold concepts. The study involved focus groups with students and professional software developers who agreed on classes and data structures as threshold concepts.

The studies implied that building a mental model best describes and explains threshold concepts. However, despite the insights derived from these studies, none presented optimal solutions to overcome difficulties in understanding such concepts. The next section discusses some of the proposed threshold concept models that aided the improvement of the framework underlying the current research.

Given that one of the objectives in designing the framework in the present study is to discover the challenges that confront novice programmers, it also investigated the five phases to uncover programming ideas that can be regarded as threshold concepts. Unravelling such concepts in programming helped in recognising programming problems that should be incorporated into this paper's framework design, such as passing parameters, loops, arrays, recursion and object-oriented programming, including class inheritance and differences between classes and objects. Covering threshold concepts in the present paper led to an in-depth evaluation of visualisation in programming learning. The framework in this paper and Chapter 6 suggest the construction of a mental model through visualisation. The discussion in this section presents findings from the five phases, which recommend using mental models and tools to overcome the miscomprehension that stems from threshold concepts.

3.4 Proposed Models for Threshold Concepts

Vagianou (2006) put forward a model as a solution to the problem presented by Salakoski (2006). The author introduced the concept of working program storage (WPS), which is a model to advance students' transition from end-user to programmer mode. The model employs a mental model and an external graphical representation. Khalife (2006) agreed with other researchers that one of the threshold concepts programming students need to know relates to developing a mental model that elucidates how the internal parts of a computer operate during program execution. Internal components rely on instruction sets that use memory visualisation to display the effects of program execution. Khalife (2006) adopted a UML diagram to represent activities and actions in a mental model, wherein instructions are categorised into *declaration*, *input*, *output* and *assignment*. The sequence of solutions is then organised in the appropriate order using the UML diagram. Meyer et al. (2016) suggested developing a threshold concepts framework based on the construct of integrated threshold concept knowledge.

3.5 Conclusion

This chapter discussed the characteristics of threshold concepts and how to recognise them. It also presented a literature review of important studies on CS concepts that are deemed of the threshold variety. The aim of focusing on threshold concepts was to understand the kinds of problems this project endeavoured to resolve. Below are summarised the insights derived from studies conducted to recognise threshold concepts in programming.

- Object-oriented programming is the paradigm most frequently indicated as an ideal representation of threshold concepts in the studies reviewed (Sanders and McCartney, 2016; Yeomans et al., 2019). The main issue raised in this regard concerns differences amongst classes, objects and instances. This gap was considered in developing the framework in this study to identify a solution to the ambiguity of the aforementioned differences. The representation of inheritance and class hierarchy, in general, was also discussed in the reviewed works. Researchers such as Sanders et al. (2008) and Eckerdal et al. (2006) found that students' fixation on understanding inheritance concepts often prevents them from correctly demonstrating class hierarchy.
- Building a mental model that shows what happens in memory during program execution is suggested to overcome students' misunderstanding. Note, however, that although mental models have been put forward by Khalife (2006) and Vagianou (2006), these representations have not been evaluated.

Chapter 4 VISUALISATION IN PROGRAMMING LEARNING

4.1 Introduction

There is a high demand for mechanisms that support the teaching of programming, particularly in finding a solution to the problems and difficulties present in programming education. In addition, using different methods of teaching can improve the learning process and motivate students to learn (Mohorovičić and Strčić, 2011), such as the systems and methods discussed in Chapter 2. These methods improve the thinking and creativity that lead to defining and analysing the problem to obtain ideal solutions. Approaches such as visualising code or using a memory diagram have made a vital contribution to the process of teaching and learning how to program. These approaches will be extensively discussed in this chapter. The visualisation method gives an overview of the studies that have been conducted in visualisation to support programming education. Tools that follow the visualisation and memory–referencing approach will be investigated in this chapter.

4.2 Memory Reference Visualisation

Memory reference visualisation or MTL is defined by Mselle (1989) as a language or device used by programmers to describe the impact of code lines on computer memory (RAM). Visualising the impact of each line of code on RAM allows novices to grasp what each instruction does and its impact. Thus, students' comprehension will improve because they can predict the result during the execution time.

A carefully designed RAM diagram can be used as a pedagogical tool to facilitate students' understanding of programming. The RAM diagram shows a direct relationship between code and its effect. Unlike a flowchart, where students need to know the meaning of symbols and their connections, it does not require that students learn any concepts. A RAM diagram is a portable, flexible and scalable tool, as it is machine- and language-independent. It can also be used as a code design and testing tool (Mselle, 1989) (see Figure 1).

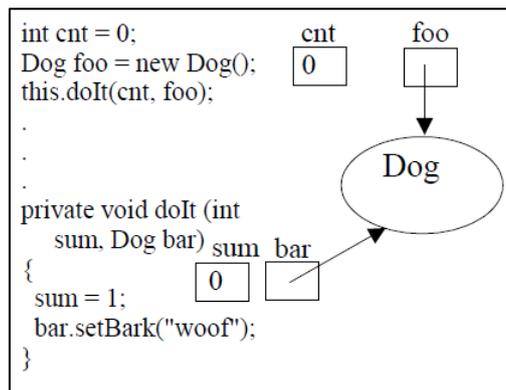


Figure 1. Memory diagram. Taken from Holliday and Luginbuhl (2003), ‘Using memory diagrams when teaching a Java-based CS1’, ACMSE.

4.3 Tools to Support Programming Learning

This section gives an overview of some tools that have been used in programming learning at academic institutions to increase comprehension among students or novice programmers. Table 2 summarises the tools with data type and programming language support, the features and defects and whether the tools have been evaluated. Note that not all tools have been evaluated; however, until the time of writing this thesis, there is no reference or evidence for the evaluations. In the case a tool has been evaluated, the evaluation study was mentioned in the previous section.

The criteria used to review the tools appear in Table 2 and are: whether the tool supports data types, whether the tool supports programming languages, whether the tool supports object-oriented programming and whether the tool has been evaluated and how it was evaluated. The researcher focused on these criteria when implementing the visualisation tool. The criteria are analysed in Sections 4.3 and 4.4 to understand their importance.

Table 2. Tools/models supporting visualisation

Tool/Model	Data types	Programming language	Features/pros	Defects /cons	Evaluated or not
The BlueJ (Hagan and Markham, 2000)	Represents classes and objects	JAVA	<ul style="list-style-type: none"> - Supports OOP - Uses the UML diagram 	<ul style="list-style-type: none"> - Does not provide dynamic visualisation 	✓
AnimPascal (Satzgemi et al., 2001)	Standard data types in Pascal	Pascal	<ul style="list-style-type: none"> - Recording programmer action - Focus on debugging - Dynamic visualisation 	Does not support OOP	✓
DrJava (Allen et al., 2002)	All data types focus on OOP	JAVA	<ul style="list-style-type: none"> - Interaction window - Testing and debugging features - Detects syntax errors 	<ul style="list-style-type: none"> - Has no control over the execution - Has no expression evaluation 	× The authors suggested testing the tool's usability
ProfessorJ (Gray and Flatt, 2003)	All data types	JAVA	<ul style="list-style-type: none"> - Testing and debugging features - Detects syntax errors - Highlights the keywords and variables - Contains three levels: beginners, intermediate, and advanced 		× The authors suggested testing the tool's usability
LIVE (Campbell et al., 2003)	Primitives, pointers, records	Converts any code to JAVA or C++	<ul style="list-style-type: none"> - Uses diagrams - Supports operations, such as allocating memory reference and recursion - Subprograms and the same variable names with different scopes are also supported by LIVE 	LIVE is a procedural system, not an object-oriented one,	×
CMeRun (Etheredge, 2004)	Primitives	C++	Dynamic execution	<ul style="list-style-type: none"> - Has no error checking; it works with syntactically free error code - Has restricted constraints on 	× The author suggested intensive testing for the tool.

				the format	
Jeliot3 (Moreno et al., 2004)	All data types	JAVA	<ul style="list-style-type: none"> - Provides dynamic visualisation - Error explanation - Ease of use - Uses UML notations to represent the objects and their relations 		<p style="text-align: center;">×</p> <p>The authors suggested an evaluation for the tool, but there is no reference.</p>
Memview (Gries et al., 2005)	All data types	JAVA; it is an extension to DrJava	Focus on tracing the objects and their interactions	Because it is built in DrJava debugger, users should learn how to use DrJava	✓
VIP (Virtanen et al., 2005)	Primitives, structs, pointers and references	C++	<ul style="list-style-type: none"> - Open-source - Controls code execution - Expression evaluation 	Does not support OOP	✓
BackStop (Murphy et al., 2008)	All data types	JAVA	<ul style="list-style-type: none"> - Handles exceptional errors during runtime - Provides suggestions and error correction 	It focuses on simplifying the runtime error and not visualising the outputs.	✓
Turtlet (Kasurinen et al., 2008)	Primitives datatype	Python	Uses games and lecture demonstrations for programming constructs	As students suggested, the tool should support algorithms and problem-solving. No program tracing or function explanation.	✓
EduVisor (Moons and De Backer, 2009)	Created specifically for OOP	JAVA	<ul style="list-style-type: none"> - Uses different shapes to represent classes and objects - Specified areas for variables and methods 	It is a proposed work; there is no reference yet about developing the tool.	×
Dekson Model (Dekson et al., 2009)	Primitives	C, C++	<ul style="list-style-type: none"> - 3D animation, online chat, video - Encourages distributed learning - Automatic answering 	It is a proposed framework (demonstration)	×
WebTasks (Rößling, 2010)	All data types	JAVA	<ul style="list-style-type: none"> - Open-source and run completely in a web browser - A database that has predefined methods and 	<ul style="list-style-type: none"> - Has a database for predefined methods and examples. Therefore, the user has no 	<p style="text-align: center;">×</p> <p>Note that students' positive feedback has been collected</p>

			examples	flexibility in writing any code	
The ANIMAL system (Rößling, 2010)	Designed for complicated data structures, such as binary trees and graphs, or for sorting and searching algorithms	Predefined script language (ANIMALS CRIPT)	<ul style="list-style-type: none"> - Visualise and animate algorithms and data structure - User-friendly navigation 		<p>×</p> <p>Note that students' positive feedback has been collected</p>
Visual Logic (Gudmundsen et al., 2011)	Primitives	No specific language; it uses icons	<ul style="list-style-type: none"> - Uses flowcharts 	<ul style="list-style-type: none"> - Has no code to be written - Does not support object-oriented programming 	×
The Alice (Aktunc, 2013), (Salcedo and Idrobo, 2011)	All data types	There is no codes; just provide animation to support the understanding of OOP	<ul style="list-style-type: none"> - Drag-and-drop interface to program a 3D world with interaction 	<ul style="list-style-type: none"> - No code is used 	✓
Online Python Tutor (Guo, 2013)	All data types	Python, JAVA, C, C++, Ruby, Javascript	<ul style="list-style-type: none"> - Web-based programming open-source tool - The use of navigation buttons - Dynamic execution of the program - Explanation of errors 	<ul style="list-style-type: none"> - Steps through execution line by line - Does not show the details of expression evaluation - Scalability - it does not support a large data structure 	✓
Visualiser (Nyamawe, 2014)	Primitives	VB.net	Uses animation	The proposed framework (demonstration) has not yet been developed	×
OOPVisual (Moussa et al., 2016)	Created specifically for OOP	Java	3D interactive visualisation tool	The proposed tool has not yet been evaluated	×
Visualised Learning Tool (VLT-OOP) (Su and Hsu, 2017)	Created specifically for OOP		2D graphical web-based tool	The proposed tool has not yet been evaluated	×

This section reviews several common tools that can be used when designing the TLE framework of the current study, where the advantages and disadvantages of the tools can be exploited during the design of the framework.

4.3.1 The BlueJ tool

The first use of BlueJ to teach Java was in 1999 in a CS introductory course. The BlueJ tool was implemented based on UML and JAVA language (Hagan and Markham, 2000; Kölling et al., 2003; Kölling, 2018). It uses a graphical representation to represent classes and objects within a project (see Figure 2). Students can create an object through a graphical user interface without writing codes and can make this object interact with other objects. This can help students strengthen their understanding of the concept of object-oriented programming (Sun, 2010; Bennedsen and Schulte, 2010).

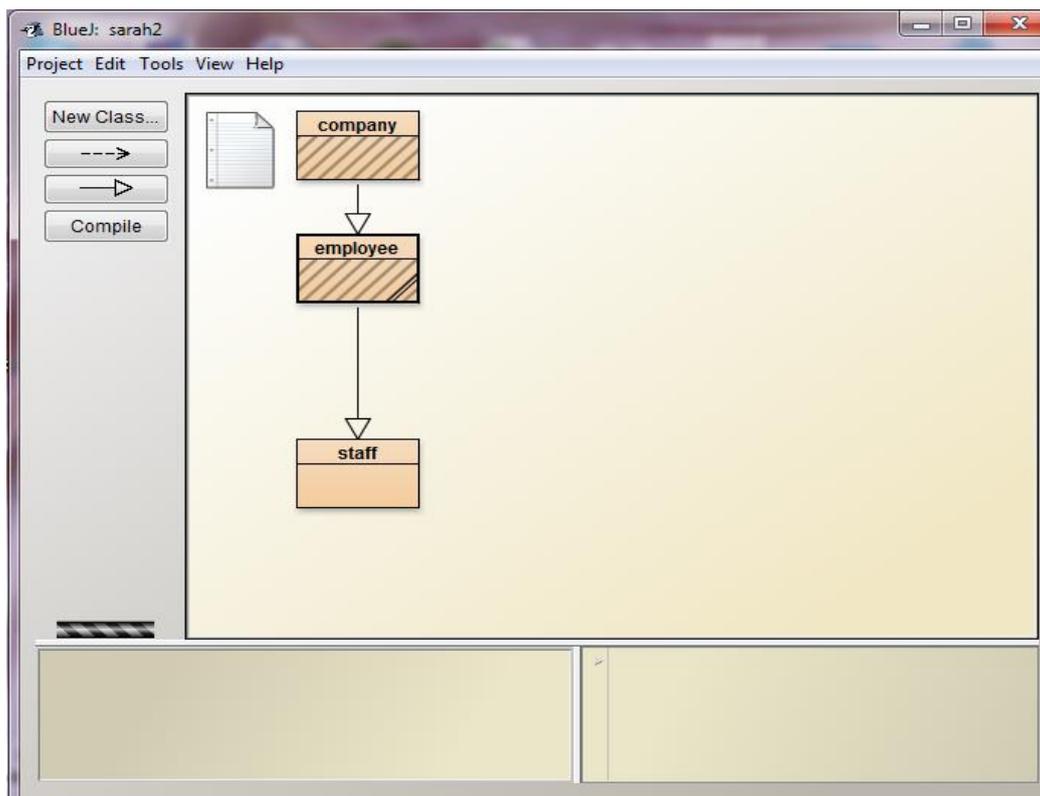


Figure 2. BlueJ. Taken from Sun, B. (2010). 'Java teaching based on BlueJ platform', *2nd International Conference on Information Engineering and Computer Science - Proceedings, ICIECS 2010*, pp. 2–5. <http://doi.org/10.1109/ICIECS.2010.5677726>.

BlueJ was selected and discussed in this research due to similarity with the current search because it is considered a program used in the teaching of Java very easily, and also easy to deal with class,

object, and method, and it is considered a very great starting point for beginners, and use it convenient for applicants, of course the program just IDE such as JCreator and Eclipse But in a different way suits educational environments such as universities due to its ease of handling. BlueJ was developed by three universities to help students understand Java language easier and faster. The idea of the BlueJ tool can be summarised as demonstrating that the class and object are physical to facilitate imagination and linking things to the learner. It should be noted that BlueJ does not need the main method to run the program; it deals mainly with the object (Barnes et al., 2006).

4.3.2 Jeliot 3

Visualising object-oriented programs was developed with Jeliot 3 after several generations. The first generation was Eliot, which was designed to produce algorithm animations. Next, Jeliot I was specially designed for Internet use, and Jeliot 2000 was dedicated to novice programmers. What makes Jeliot 3 different from the previous generations is the extension of visualising object-oriented concepts. Jeliot 3 is different from BlueJ in that it provides dynamic visualisation. In Jeliot 3, many features have been added to suit user requirements. These include ease of use, consistency and continuous and complete visualisation that is extensible both internally and externally. The visualised components that show up in a separate window of Jeliot 3 are designed to simplify the use of the tool with an animation frame structure (Figure 3, captured from the researcher's personal computer). Error explanation and the ability to highlight the line of error is also provided in Jeliot 3. Jeliot 3 uses UML notations to represent objects and their relations to clarify object-oriented concepts for students (Moreno et al., 2004; Hongwarittorn and Krairit, 2010).

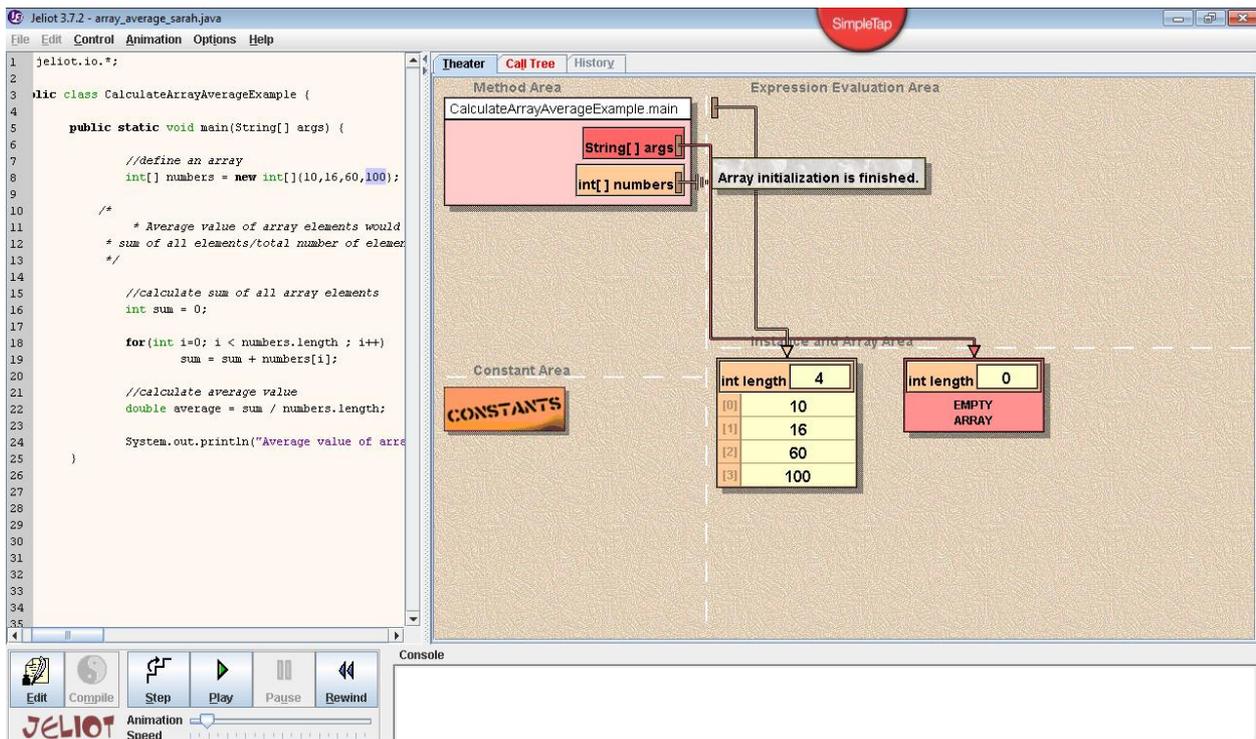


Figure 3. Jeliot - screenshot from the researcher's personal computer

Jeliot 3 is one of the tools used in Java education. It is a visualisation tool that can be used in introductory programming courses. The tool has wide range of Java programs and can interact with BlueJ IDE. Jeliot 3 has been tested and found to be useful for beginner students who have difficulties in learning programming. Jeliot 3 is an animation tool that aims to support beginner programmers and CS students in the introductory stages of learning programming and presents object-oriented Java object implementation by activating source code evaluation. Most Java is currently supported by Jeliot 3. The tool has been developed to support object-oriented programs and to redesign the structure of programs for the better (Moreno et al., 2005).

This tool was used because it helps students in different educational environments. In addition, this program is one of the tools that will contribute significantly to teaching Java to beginners and will help motivate novice programmers and encourage them to program.

4.3.3 DrJava

The purpose of the DrJava tool is to teach students how to design programs in Java and how to test and debug programs. It consists of a window with two panes linked by an integrated compiler

(Figure 4). The interaction pane is used to input Java expressions, and the definition pane is used to enter and edit class definitions. The features in DrJava include the interaction window, which has a ‘read-eval-print’ loop (REPL) to enable access to program components without recompiling. Testing and debugging features are also available using REPL to test the methods individually. Moreover, students can debug the code without needing to learn the debugging mechanisms. DrJava includes an editor to detect syntax errors, and it can highlight the parentheses. DrJava also has an integrated compiler that is bundled with the Java compiler (Allen et al., 2002).

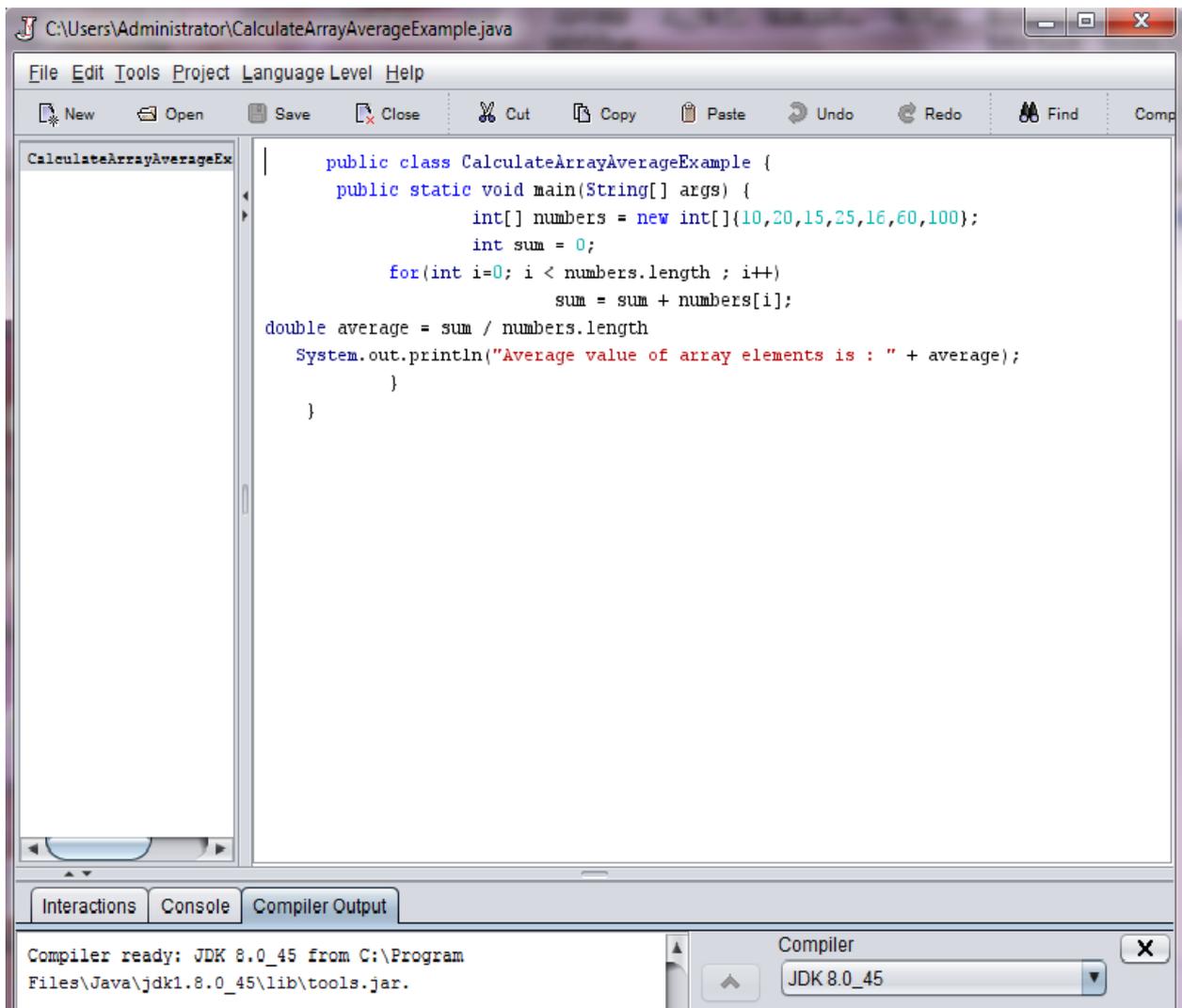


Figure 4. DrJava. Taken from Allen, Cartwright, and Stoler (2002), ‘DrJava: a lightweight pedagogic environment for Java’, a paper presented at the SIGCSE.

The DrJava tool is a newly designed program to help beginners and school students to get a development environment for applications in the way Java easily, making it easier for them to run a document programmed in an easy way, with plenty of tools to help manage applications

interactively. DrJava is a powerful program that provides users with an intuitive programming environment in order to help them create Java applications and apply their creations. This program and all its features are designed without any complications to help students understand the structure of documents and to identify errors easily. It is a program that allows users to work with multiple documents at the same time and allows the possibility of switching between documents. DrJava provides a lot of advanced search and navigation tools as well as the possibility to facilitate the management of bookmarks. In addition, this program is characterised by a user-friendly interface that is fairly intuitive and customisable, meaning its appearance can be changed at any time. The program was specifically designed to help students learn and use the Java programming language. The program features a range of easy-to-use tools. It also has a toolbar and a set of buttons, colours and window positions. In addition, DrJava is a lightweight Java IDE that combines sophisticated tools with ease of use to provide an easy environment for learning the introductory programming language (Allen et al., 2002).

Given the above-mentioned, the DrJava tool was chosen as one of the Java application development environments, as it is a tool that makes it easier for novice students to learn programming in the Java language because it has a simple programmed document operating environment with many tools to help manage applications interactively. It has an easy-to-use and somewhat intuitive interface, as it was specially designed to help students learn and use Java.

4.3.4 ProfessorJ

ProfessorJ is a pedagogical environment that presents an interface for the Java compiler. It is implemented based on DrScheme (Figure 5). The ProfessorJ interface consists of two windows, a definition window containing the code and an interaction window that provides an REPL to experiment with the code. It has three levels of difficulty—beginner, intermediate and advanced. In beginner mode, students can define the declaration construction and its restriction. The intermediate mode starts with teaching object-oriented programming. The advanced mode introduces loops and arrays. The code in ProfessorJ highlights the keywords and variables. It contains a check-syntax

tool. Students can track their variables by binding an instance of the variable to all its uses with arrows. ProfessorJ has a feature that highlights errors outside of the debugging environment and that can stop the execution at any time during the debugging mode (Gray and Flatt, 2003).

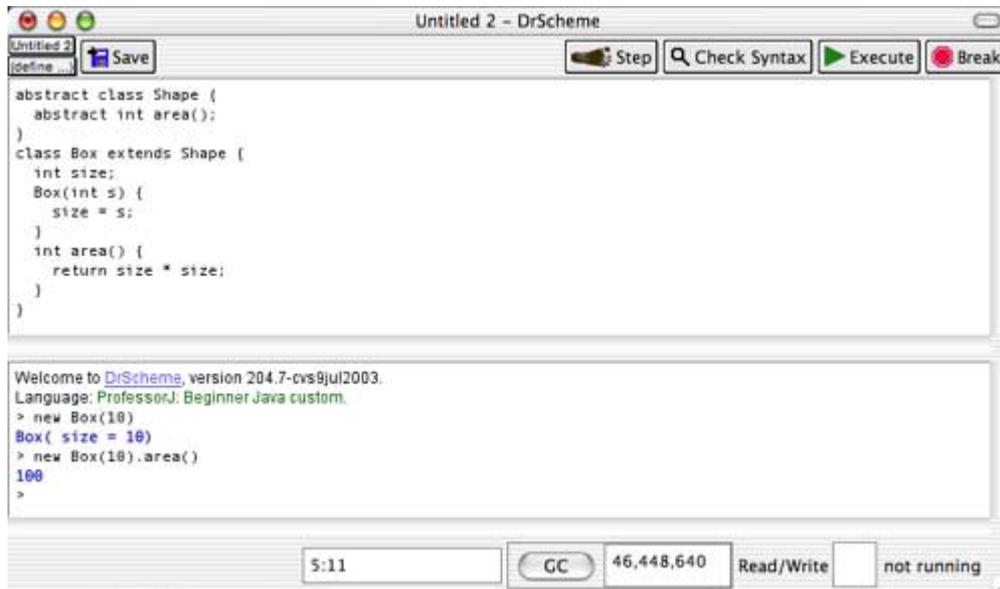


Figure 5. ProfessorJ and DrScheme. Taken from Gray, K., and Flatt, M. (2003). 'ProfessorJ: a gradual introduction to Java through language levels'. Companion of the 18th Annual ACM SIGPLAN, 170–177. <http://doi.org/10.1145/949344.949394>.

ProfessorJ is one of the tools used to teach programming for beginners in Java. It allows multiple class declarations in the definitions window. It also works the same way as DrScheme and DrJava, so that students using ProfessorJ can access classes written in the Definitions window within the Interactions window after pressing the Execute button. Each level can interact with other levels, allowing students to reuse applications in previous levels and allowing teachers to provide full support libraries implemented in full Java that adapt to educational needs. ProfessorJ is characterised by a programming environment that faithfully implements a series of pedagogical subgroups and thus reduces confusion among students, saves the teacher the time it takes to explain details of a language that is not yet relevant and encourages students in general to think in terms of well-defined behaviour. In addition, ProfessorJ, a new Java programming environment, offers a series of languages. ProfessorJ language levels are specifically designed to avoid too much confusion and to provide support for the novice programmer through a Java-like programming language. The ProfessorJ environment offers different levels of language (beginner, intermediate,

advanced) and provides a simplified interface for the Java compiler and a virtual machine for the novice student. Each of these levels is a subset of the language to gradually introduce students to grammatical and semantic details. This tool supports the learning process at both the beginner and intermediate levels. ProfessorJ customises the Java language and error messages to the needs of students. Due to their evolving needs, ProfessorJ offers several language levels, from Beginner Java to Full Java (Georgantaki and Retalis, 2007; Hsia et al., 2005).

4.3.5 WebTasks

WebTasks is a programming task database tool that runs entirely in a web browser and does not require downloading any program (even Software Development Kit (SDK) on the students' devices). WebTasks contains Java Server Pages (JSP) pages that run on the Apache Tomcat Server (Figure 6).



1st step: 1 3 2
2nd step: 1 4 5 2
3rd step: 1 5 9 7 2
=> { 1, 5, 9, 7, 2 }

Please solve this task using only loops, not recursion!

```
1 public static int[] studentsMethod(int[] array_in, final int steps) {  
2  
   int currentLevel = 0;  
   int[] old = array_in;  
   int[] result = null;  
  
   if (steps == 1 || array_in == null)  
       result = array_in;  
  
   // loop over the target levels  
   for (int level = 1; level <= steps; level++) {  
       // create array  
       result = new int[old.length + 1];  
3  
4  
5   return result;  
6 }
```

Abschicken Zurück

Figure 6. WebTasks. Taken from Rößling, G. (2010). 'A family of tools for supporting the learning of programming', *Algorithms*, 3, pp. 168–182. <http://doi.org/10.3390/a3020168>.

The students pick an assignment and then write the body of the methods; most of the methods have a predefined header. The code is then tested using JUnit. The series of testing continues as recursion until the problem is solved. The Department of Computer Science at the Technical University of

Darmstadt (Germany) developed a system using WebTasks to solve about 118 Java programming tasks. CS students can log in to the system and try all the tasks, which encourages them to write Java programs, submit them and receive quick feedback on corrections (Rößling, 2010).

WebTasks is one of the programs used in teaching programming for beginners and is a functioning platform that allows building applications without a server. WebTasks allows one to build applications without thinking about infrastructure. One just writes the server-side logic of the topic or application one wants to create, deploying its functionality via WebTasks and accessing the backend of servers via HTTP. With a preference for code over configuration, the WebTasks platform comes with a familiar programming model and excellent authentication and authorisation support to ensure a pleasant development experience. The WebTasks features a familiar programming model, an easy-to-use Command Line Interface (CLI) and a robust infrastructure to help students or novice programmers achieve their goals. WebTasks allows one to securely connect to application programming interfaces (APIs) that require secret keys, set up web links that run after certain actions in special applications or talk directly to the current or background database. Server-free computing is a fairly recent trend in software development that allows developers to focus on writing application logic and not worrying about server provisioning and management (Baldini et al., 2017).

Based on the above, the researcher has chosen this tool and reviewed its advantages and disadvantages in teaching programming in the Java language to junior students. WebTasks' easy-to-use features and strong structure help beginning programmers achieve their goals.

4.3.6 The Alice tool

The Alice tool uses a built-in drag-and-drop interface to program a 3D world with interaction (see Figure 7). Alice is a project of Carnegie Mellon University in the US and was designed to introduce young people to programming. They learn the fundamentals of object-oriented programming through the graphical representation of objects (Salcedo and Idrobo, 2011).



Figure 7. Alice. Taken from Salcedo, S.L. and Idrobo, (2011). ‘New tools and methodologies for programming languages learning using the scribbler robot and Alice’, *Proceedings of the Frontiers in Education Conference, FIE*, pp. 1–6. <http://doi.org/10.1109/FIE.2011.6142923>.

The Alice tool is one of the programs used in Java education. It can create 3D animation, and for non-professional programmers Alice is the best choice. Alice was initially designed to teach the basic concepts of programming and is used in many colleges and courses. The program has a simple and easy-to-use user interface. Alice is an open-source educational application based on object-oriented programming in an integrated development environment. It was developed by Java and has drag-and-drop functionality to create animated 3D computer graphics. Alice’s application is characterised by avoiding the problems and obstacles that exist in other tools that make them unsuitable for the educational curriculum. Most programming languages are designed to produce code intended for a commercial product, but Alice is intended for educational purposes only. Alice is associated with the integrated development environment and supports object-oriented programming. It is designed for people not associated with programming, such as school students, and has drag-and-drop functionality (Sykes, 2007). Based on the above, and because Alice is one of the tools used in teaching Java, particularly for non-professional programmers and Java beginners, it is considered one of the best options for teaching the basic concepts of programming.

4.3.7 The ANIMAL system

Different methods for visualising and animating algorithms are used to address more complicated problems in data structures, such as binary trees and graphs, or to sort and search algorithms (Figure 8). The process helps students understand the behaviour of these structures and algorithms. The ANIMAL system is one such method. It follows the concept of visualising the representation of source code and highlighting the current line being executed (Röbbling, 2010).

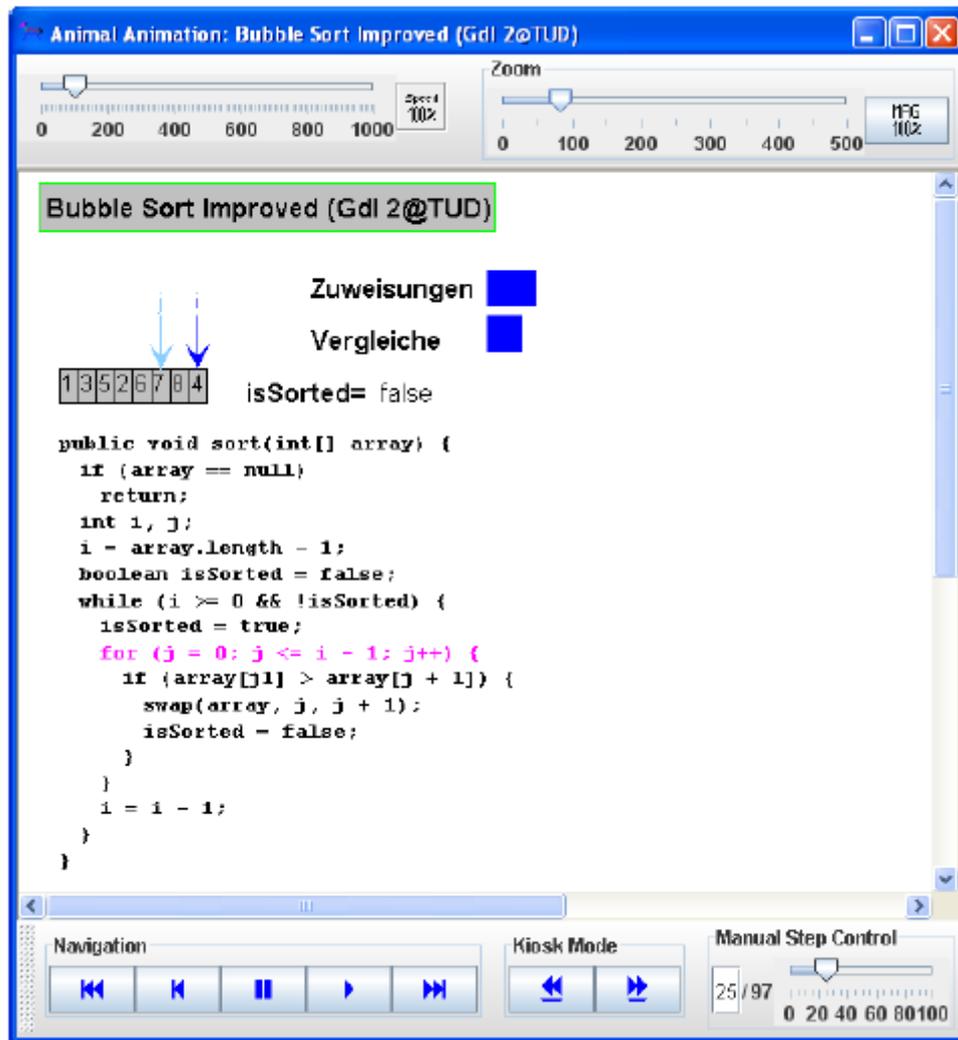


Figure 8. ANIMAL system. Taken from Röbbling, G. (2010). 'A family of tools for supporting the learning of programming', *Algorithms*, 3, pp. 168–182. <http://doi.org/10.3390/a3020168>.

The ANIMAL algorithm animation tool is used in teaching programming for beginners. It is a new tool for developing animation to be used in lectures. It provides a small but powerful set of graphical operators. The animation is created using a visual editor via scripting or via API calls. All

animations can be edited visually. ANIMAL provides visual animation editing and is therefore easy to use. A simple scripting language and animation API are also provided. Further, ANIMAL has a set of powerful features that can be easily integrated to create and display animations of algorithms, data structures and many other things. ANIMAL is used in introductory CS courses at universities due to its ease of use and the fact that little knowledge of software is needed. It is considered a basic starting point for graphics and programs (Rößling, Schürer and Freisleben, 2000).

4.3.8 Visual Logic

Visual Logic uses the concept of iconic programming (icons and flowcharts) to visualise programs. Visual Logic has no code to be written (Figure 9). Instead, the user creates a flowchart that represents the code. Subsequently, the tool traces the flow of the code. The tool demonstrates the outcomes of executing each icon in the flowchart in popup windows. Visual Logic does not support object-oriented programming (Gudmundsen et al., 2011).

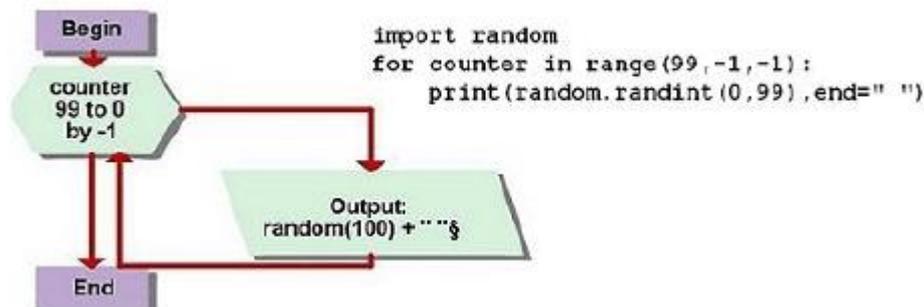


Figure 9. Visual Logic. Taken from Gudmundsen et al. (2011). ‘How to use executable flowcharts to enhance learning in general education, CS0, and CS1 courses: tutorial presentation’, *The Journal of Computing Sciences in Colleges*, 26(6), pp. 107-109.

Logic programming techniques are adopted for the characterisation, allotment and analysis of formal languages and natural languages. Visual languages are adopted in this study because they broaden human–computer interaction. Visual languages are languages that utilise diagrams or other spatial, graphical representations. Logic, which is a proven framework for handling sequential, textual languages, can be well employed as the formal basis of such a framework (Meyer, 1993).

Based on the above, the reason for choosing Visual Logic in this study is that visual languages depend on expanding the interaction between humans and computers.

4.3.9 Online Python Tutor

Online Python Tutor is a web-based programming tool that uses extensive visualisation. It is open-source software where the user embeds their code into the web page, as shown on the left in Figure 10. Subsequently, the code can be traced by using navigation buttons (Figure 10). The visualisation of the code is shown on the right in Figure 10. This visualisation enables users to watch the program's dynamic execution. As the program executes, it depicts changes to frames and objects. Additionally, it has a program output area. The tool provides explanations of errors, with indicators pointing to the line on which the error occurred (Guo, 2013; Karnalim and Ayub, 2018).

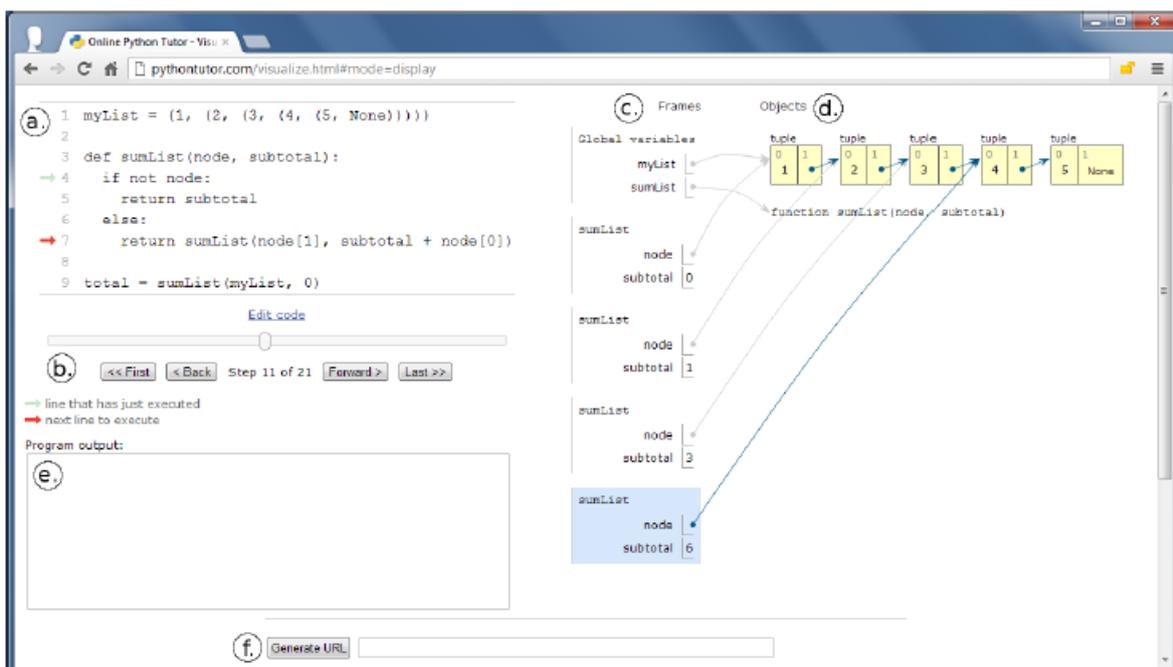


Figure 10. Online Python Tutor. Taken from Guo, P.J. (2013). 'Online python tutor: embeddable web-based program visualisation for CS education', in *Proceedings of the 44th ACM technical symposium on computer science education*, pp. 579-584.

Python is a multi-purpose explanatory language widely used in many fields, such as in building independent programs using known graphical interfaces and in web programs. It is also used as a scripting language to control the performance of some of the most popular programs or build extensions. In general, Python can be used by beginners to create simple programs and to

accomplish large projects. Beginning programmers are often advised to learn this language because it is among the most popular languages. Python is fairly easy to learn. As Python has unusually easy structures, it is a very powerful but simple way to do object-oriented programming, particularly in comparison to languages like Java (Ishizue et al., 2018). Online Python Tutor was chosen in this study because it is one of the tools used by beginners to create simple programs and because it is both popular and easy to use.

4.3.10 The Visualiser

Nyamawe (2014) presented a framework for a visualiser tool based on MTL (Figure 11). A visualiser is a tool used to visualise code execution. It explains the hidden processes during the program runtime, as it displays the line-by-line execution of the computer program. Additional features should be added to the visualiser, as Nyamawe suggested, such as the ability to replay the execution, pause the execution and select the execution speed. This proposed framework needs to be evaluated to assess its functionality and effectiveness among novice programmers.

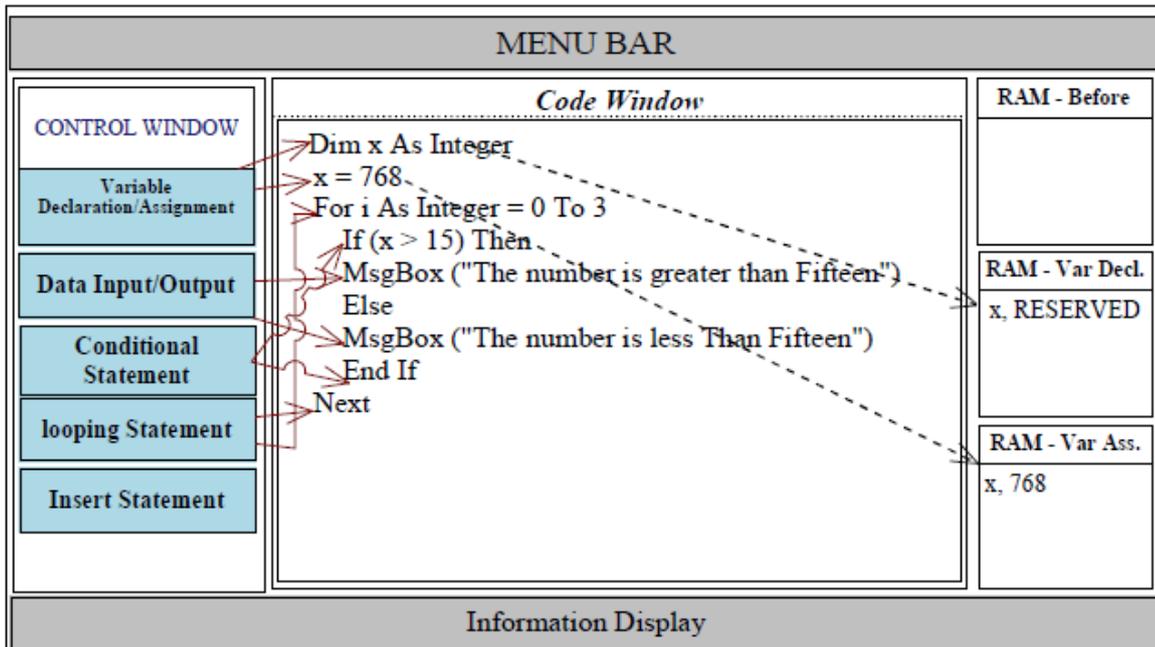


Figure 11. The Visualiser tool. Taken from Nyamawe, A.S. (2014). ‘A proposed framework for development of a visualizer based on memory transfer language (MTL)’. arXiv preprint arXiv:1408.2564.

Visual communication is used every, particularly on the Internet. Visualisation is an aspect of all fields, from construction to engineering to architecture to geography to chemistry. Additionally,

digital technology has become an essential component, especially in terms of the discovery of new knowledge, principles and shifts in the perception of existing theory. This was still significantly recognising by most teachers, students and pupils. Visualisation is forming a mental image , whereby outcomes are perceived by the visual receptors. In education, visualisation is associated with the application of rules. The visualisation of information is a universal technique regardless of language, decoding speed or relativity. The danger lies in rational operations, hypertrophy of sensory impressions and ambiguity of the information. The current trend vis-à-vis visualisation is speeding up communication using a single and comprehensive tool that facilitates communication not only of routine matters but also of scientific and technical knowledge. It is of key importance in education at school and in lifelong learning (Shatri and Buza, 2017).

4.4 Visualisation Evaluation

Since the beginning of using visualisation in programming education, many researchers have pointed out its advantages (Kasurinen et al., 2008; Mselle and Twaakyondo, 2012; Fouh et al., 2012; Mather, 2015). Researchers have applied different methodologies to evaluate visualisation learning systems based on a common hypothesis, which is whether using visualisation or ‘memory diagrams’ enhances students’ ability to program. Some studies have examined other aspects, such as usability, complexity, interest and ease of use.

This section evaluates the use of visualisation in previous studies to analyse the current state of the art and identify emergent issues. The theoretical literature and previous studies related to the conceptualisation of programming education will be reviewed in three ways—based on the methodology followed in the study procedures, based on the measured factors in each study and based on the studies related to programming topics mentioned in each study. The nature of a study plays a vital role in its findings and the research accuracy. The following are the aspects explored in the discussion.

4.4.1 The methodology used to evaluate the tools

This section discusses how some visualisation tools have been evaluated. The evaluation criteria differ, and some tools have been evaluated based on their impact on students' performance or on experts' opinions. The purpose in this section is to study the different ways of evaluation and to choose the most suitable for evaluation in the present study. Different evaluation methods (mixed methods) are required to evaluate students' comprehension and the factors that affect it. Using mixed evaluation methods where both qualitative and quantitative data are gathered leads to the development of better activities to be undertaken in any educational innovation. Using an evaluation methodology to collect qualitative data may fail to provide information on students' performance, such as scores, drop-out or fail rates and programming errors. However, if only quantitative data is used student feedback may be lost and the factors that affect their performance might not be discovered.

In reviewing related works, it was found that some studies did not make adequate use of the methodology. For instance, in Virtanen et al.'s (2005) study, a questionnaire was presented to students to evaluate the visual interpreter (VIP) tool, an open-source visualisation tool for teaching C++.

VIP attempts to overcome the problem of creating new visualisations in which most instructions differ from those for using visualisation tools. VIP's features support editing the code, controlling code execution, showing the state of the program and providing expression evaluation. The questionnaire answers revealed positive feedback about the tool. However, as using the tool was not mandatory, the study results were not broad enough to measure the tool's usefulness. Therefore, the researchers opted to use only the quantitative data to evaluate the tool and did not consider the qualitative data, which may well have affected the results. In the same manner, Sun's (2010) study evaluating BlueJ as a pedagogical tool was done by conducting an experiment to solve an object-oriented programming problem. Students benefit from the tool by learning the concept of object-

oriented programming using the interactive mode, but the author failed to provide qualitative data supporting the findings.

Other studies reviewed failed to present qualitative data to support the statistical data gathered. For instance, a study by Mselle and Twaakyondo (2012) counted the errors committed by two groups of students. The study aimed to determine the role of MTL in reducing misconceptions and errors made by novice programmers. The statistics indicated that students using MTL committed fewer errors (208) than the control group (392). The authors suggested that further experiments should be conducted with larger groups and with more diverse populations from different universities and organisations. However, they seemed to overlook that the number of errors may represent misunderstanding or other factors, such as students' interest or confidence, which could have been determined using qualitative methodology.

Mselle's (1989) study has the same issue, that is, it only considered students' scores in evaluating the use of a memory diagram in programming education. A class experiment was conducted with 100 students at the Kigali Institute of Science and Technology. The students were divided into two groups, one using a traditional approach (control group) and the second (experiment group) being instructed by a lecturer using a memory diagram. The performance evaluations of both groups were made based on their scores on the same final examination. There was a significant difference in the test scores of the two groups; the average score was 64.67% for the experiment group and 60.02% for the control group, so the hypothesis was accepted to a certain degree. However, the researcher stated that certain conditions should be considered when judging the two groups, such as the possibility that the experiment group was smarter or that their teacher was better than the teacher of the control group. To address these issues, more class experiments and other qualitative methods should be carried out to best determine possible reasons for any difference between the two groups' scores.

The study conducted by Zheng et al. (2017) used a closed questionnaire to evaluate the role of visualisation variables in teaching C programming students. The study used quantitative data to

measure students' performance and satisfaction. Even though the results showed a significant improvement in the program construction of novice programmers, students' scores were the same for the group that used the visualisation and for the control group that did not. The results of the satisfaction questionnaire showed the students were not significantly satisfied with their program construction capabilities. The limitation of this study is that a qualitative experiment using, for example, open-ended questions was needed to offer a realistic explanation of the quantitative findings.

In some cases, evaluations lacked statistical evidence, such as in Gouyon and Dixon's (2005) study that tested code memory diagram (CMD) software used by the author/developer. The evaluation was conducted during a teaching session. The results revealed that the software enhanced students' comprehension during the teaching session, but the main weakness of the study was its failure to address the impact of the tool on the teaching process. The study could have been improved by including aspects such as lecturer preparation time and students' grades and experience. Including these aspects would have afforded a better explanation of the study's findings.

In Gouyon and Dixon (2005), three lecturers were interviewed to learn about their experiences using the software. The lecturers suggested some modifications to the CMD editor and viewer to make the interface easier to use. Gouyon and Dixon's argument relies too heavily on quantitative analysis. Another weakness of the study is that the findings are limited to the lecturers' perceptions. An objective comparison of students' performance and lecturers' perceptions seems to have been needed.

Another effect of the methodology on the evaluation accuracy is the number of participants involved; this plays an essential role in the findings. In their first evaluation of the CMD, Gouyon and Dixon used various methods to collect data, including a questionnaire, video observation and participant observation by the teachers and authors. Twenty-five students (of 26) answered the survey, and 23 indicated the software facilitated their comprehension. The remaining two said they previously had no difficulties in understanding the concepts, so the software did not add new

material for them. The main weakness of this study was the number of students who participated in the experiment. It is likely that increasing the population would increase the accuracy of the results.

4.4.2 The measured factors

Regarding the use of visualisation in programming education, researchers have specified certain factors to be measured during their investigations, such as in the studies by Kasurinen et al. (2008) and Hagan and Markham (2000). In some cases, these measured factors are not enough to judge visualisation. More than one factor is needed to fully evaluate the visualisation learning systems, such as compiling students' scores or errors committed and the sociological aspects relating to the students, such as their feelings, interests, opinions, ability to program, confidence and/or satisfaction. The reason for selecting more than one aspect is to better understand the findings that will lead to improvement. Additionally, the evaluation should be based on different perspectives, those of students, lecturers, tutors and/or developers. What students suggest may not be noticed by lecturers and vice versa. An example of missing factors can be seen in a study by Hagan and Markham (2000). Their experiment was conducted at Monash University with 350 students who started writing a single class then gradually began writing more classes. The students used the interactive visual environment of BlueJ, so they advanced from that environment to learn the object-oriented concept. During the semester, surveys were used to evaluate the students' expectations and their performance in BlueJ. The study collected information about the students and their background knowledge in programming and then evaluated their BlueJ experiences. Even though the survey showed the majority of students had a positive shift in their attitude towards BlueJ, they still doubted its stability and reliability. The study's weakness, which leads to not understanding the reason for the positive shift in the students' attitude, is the lack of other factors that should have been measured, such as students' performance.

Similar to evaluating the 'Turtlet' tool, which has been used as a visualisation approach in programming education (Kasurinen et al., 2008), the researchers collected students' opinions using two surveys. The surveys measured aspects such as complexity, interest and ease of use. Although

the findings showed that 35% of the students liked the tool, 81% of the students preferred doing the course project with Turtlet, 96% enjoyed the exercise that was created using Turtlet and 85% preferred the demonstration. However, the study fails to consider other aspects that would explain the reason behind this interest. One question that needs to be asked is whether students' scores were affected after using Turtlet. In a study by Salcedo and Idrobo (2011), the authors evaluated the Alice tool by comparing it with different visual tools, such as SCRIBBLER, Microsoft Robotic Developer Studio 2008R3, NXT Tech Virtual Robotic Worlds and Lego NXT2.0 and with traditional teaching. The experiment conducted at Icesi University involved 15 lessons covering the basics of programming. Each lesson included video phones and a workshop, and the videos included tutorials covering specific topics with three parts—a theoretical introduction, a step-by-step exercise and results. A questionnaire was distributed at a workshop held after the experiments to examine the effectiveness of using the tools. The results suggested that using Alice increased the interest in programming; however, students generally had a negative view of programming languages before the experiment, as 58% of them found them hard to understand, 92% said they had enough experience with the lessons and about 67% of them said they had fair or good experiences with programming languages such as Java and C++. The main weakness of the study is that it relied on the students' interest and did not consider performance, which can be used to further explain the findings.

Murphy et al. (2008) evaluated the BackSTOP tool that has been used for debugging runtime errors. The aim of developing BackSTOP was to handle exceptional errors during runtime. It provides clear error messages and suggestions on how to fix them when exceptions occur in Java programs. The tool was designed to explain 22 different types of errors and exceptions. Two studies were conducted to evaluate BackSTOP by measuring two aspects, the time needed to find the logical errors and the time needed to solve them. The first experiment recorded the time needed by 17 students (13 used BackSTOP) at Columbia University to identify and fix the errors related to the task. The students' answers were collected after some subjective questions about solving the given

task. Of the 13 students, 76% recognised the errors and fixed them within eight minutes. However, the students in the control group who were selected from the top students fixed the errors in less than five minutes. The reason it took so long was that the messages in BackSTOP were too long and it took time to read them. The second experiment was conducted to find logical errors in the program and involved all 17 students; 47% found the errors within a maximum time of one minute. The students claimed that the debug tool helped them find errors. A limitation of the study was that the researchers considered the debugging only. More aspects could have been considered, such as the ability to write a program with the least number of errors. The authors also suggested comparing the debugging features when compiled with runtime error handling.

A study conducted by Yang et al. (2018) evaluated the JaguarCode tool that supports Java programming along with UML diagrams by using static structure and dynamic execution traces for the program. The authors carried out quantitative and qualitative experiments to investigate the tool's effectiveness and the participants' satisfaction. The experimental results revealed that using both static and dynamic visualisations had a positive impact on correctly understanding the program and the tracing problems. The quantitative results showed two aspects, the correct responses for each question and the response time. The qualitative results showed the participants' satisfaction and visualisation usability. The study's main weakness was the failure to address how JaguarCode can support students in designing and implementing Java through practical programming exercises. The study also only collected data from the students' viewpoint. More investigation with experts is suggested to better improve the tool. Contrary to that study, which used a questionnaire that involved questions about the static structure and dynamic execution traces for the program, Zhang et al. (2013) relied on asking students about data in the static mode of the program and not during runtime. The study evaluated the use of visual language in animation programs and the impact it has on students' comprehension. The findings showed that visualisation increased the students' comprehension with less effort and less time. However, the questions asked did not focus on the behaviour of the animation in the program, such as the functions and change of variables during

runtime. Instead, it focused on data and control flow. Extending the tool with additional visualisation based on dynamic analysis would better assist users. Another limitation of the study is that it measured students' performance and the usability of the tool. As the authors stated, they need to investigate other aspects, such as how the visualisation approach helps people implement animations, remix animations and develop programming skills by exploring animations, debugging and/or extending animations.

4.4.3 Programming topics

Some of the studies on evaluating visualisation systems only investigated one programming concept or solved one programming algorithm or one problem\example. These studies are not considered weak; however, the present study looks at a wider range of programming concepts, such as the threshold concepts discussed in Chapter 3. The reason is that some of the programming concepts may have an effect on the understanding of other concepts. Programming concepts are considered structures built on each other. This is what this study has tried to demonstrate—how understanding one concept may lead to the improved comprehension of others. In the following examples, the benefits of these studies are shown, even though they fail to cover a wide range of concepts.

The focus is on object-oriented programming, such as in the studies by Holliday and Luginbuhl (2003, 2004) and Sun (2010). The study by Holliday and Luginbuhl (2003), conducted at Western Carolina University, evaluated the use of memory diagrams in object-oriented programming. A memory diagram is a model that represents entities in object-oriented languages. Entities are described using shapes, such as circles for objects, rectangles for variables, diamonds for classes and arrows for references. When students use this model, they can see the program code. The memory diagram next to the code values of the data written inside the shapes dynamically changes based on the code execution. The memory diagram was used with CS1 students to help them understand object-oriented programming concepts. Periodically, the students were asked to construct a diagram by themselves for better comprehension and to improve their abstract thinking. The university proposed to study the diagram's effect on program comprehension in the fall

semester of 2002 by including a problem on the final exam that showed code and required the students to draw a memory diagram. There was a high correlation between the score on the system diagram question and the entire test score. The study concluded that performance on this issue could predict a student's performance on the whole test.

In 2004, Holliday and Luginbuhl (2004) conducted an experiment that used a memory diagram to assist students' comprehension. This assessment was then used as feedback to the students to correct their incomprehension. The students applied the memory diagram on a given example to evaluate their comprehension. On the given code, some common misunderstandings appeared, which allowed the instructor to know the students' weak points. Most of the weak points related to common misunderstandings when assigning an object and how they related fields to objects. Two more studies were conducted to evaluate the memory diagram as an assessment tool. The goal of the first study was to find a correlation between students' performance on drawing a memory diagram for a given test question and to use their performance as an indicator to measure their ability to solve the remaining questions on the same test. The goal of the second study was to find a correlation between students' performance in drawing a memory diagram and their performance on the entire course assessment. The authors found that the correlation existed and that a memory diagram could be used as an indicator of students' performance on the test and on the course assessment.

Sun (2010) also evaluated the use of visualisation. That study evaluated BlueJ as a pedagogical tool in implementing a project-building clock. The problem is an object-oriented example where students divide the problem into two classes, DisplayNumber and DisplayClock. Students benefit from the tool by learning the concept of object-oriented programming using the interactive mode.

Satratzemi et al. (2001) evaluated AnimPascal as a visual tool. What makes AnimPascal different from other debugging tools is its ability to record programmers' actions. These recorded actions could be used by instructors to recognise students' misconceptions. AnimPascal can edit and compile standard Pascal code and allows observers to watch the dynamic execution.

4.5 Conclusion

As this thesis looks at the visualisation method as a proposed solution for the study gap, this chapter provided a literature review examining the contribution of visualisation to programming learning. From the previous literature, it can be concluded that different aspects need to be considered to better evaluate the use of visualisation in programming learning.

Broader and more programming concepts need to be evaluated. In evaluating the methodology used in programming education, it is better to investigate the use of visualisation in different programming problems. The findings may be different from one programming concept to another.

Several common tools were also discussed in the chapter; therefore, this study could potentially benefit from them in designing the TLE framework. The features and defects of the tools could be exploited in designing the framework.

It is possible to motivate and encourage novice programmers to program using visual tools. The pedagogical environment presented in the visual tools allows for easy understanding of the theoretical concepts. The tangible results of evaluating visualisation in programming learning discussed in this chapter prove the benefits of using visualisation to increase students' comprehension and interest in programming. The various tools and models—BlueJ, Alice, DrJava, Online Python Tutor, VIP and AnimPascal—can become a part of strengthening the theoretical concepts. Visualisation can show the theoretical concepts through interactive environments and the use of animations, making them easier to understand. However, the field of visualisation should be expanded and undergo continuous improvement and evaluation. In general, when designing a visualisation model, one should focus on other aspects besides the graphical representation. One of these aspects is the theoretical concepts in programming and how they could be fully or partially covered. Object-oriented programming has been used in some of the tools presented in this chapter. However, it could be employed and improved in different ways. Still other aspects should be considered, such as the programming language, the amount of practice, type of examples, data type support and error handling. The next chapter presents the methodology used for both recognising

the aspects considered in developing a visualisation pedagogical environment from the novice's perspective and evaluating visualisation in programming education.

Chapter 5 RESEARCH METHODOLOGY

5.1 Introduction

The research method is the technique used to collect data using instruments such as questionnaires, structured interviews and participant observation (Creswell, 2014). The use of any research method mainly depends on the nature of the research and its objectives.

To better achieve the research objectives of this study, a combination of quantitative and qualitative methods was chosen. The reason for using both methods is discussed in this chapter in general and more specifically in Section 5.2.

The study's research objectives include the following. The first was to evaluate the existing visualisation tools from the perspective of novice programmers. The novices' requirements were collected to discover the strong and weak areas in the visualisation tools for the purpose of improvement. Then the collected characteristics of the visualisation tools were formalised and considered as features of this study's visualisation tool. The novice programmers' requirements and their feedback were collected via semi-structured interviews. During the interviews, three visualisation tools were presented to the interviewees to gather their opinions and suggestions for improvements.

The second research objective was to evaluate the developed visualisation tool based on novices' performance (students' scores), novices' confidence while solving programming problems, novices' confidence in their understanding of the programming concepts and novices' satisfaction. A series of pre-survey and post-survey experiments were conducted to achieve the research objectives. In addition, focus groups were conducted with the novice programmers after attending a visualisation session to collect their feedback. Finally, expert evaluation was sought by conducting semi-structured interviews with experts for further feedback from their perspectives.

5.2 Research Methodology

The research methods used are within the context of the epistemological approach. Of the two main paradigms proposed by Creswell (1994, 1998), this study uses a qualitative and quantitative analysis model. Figure 12 shows the qualitative and quantitative methods used.

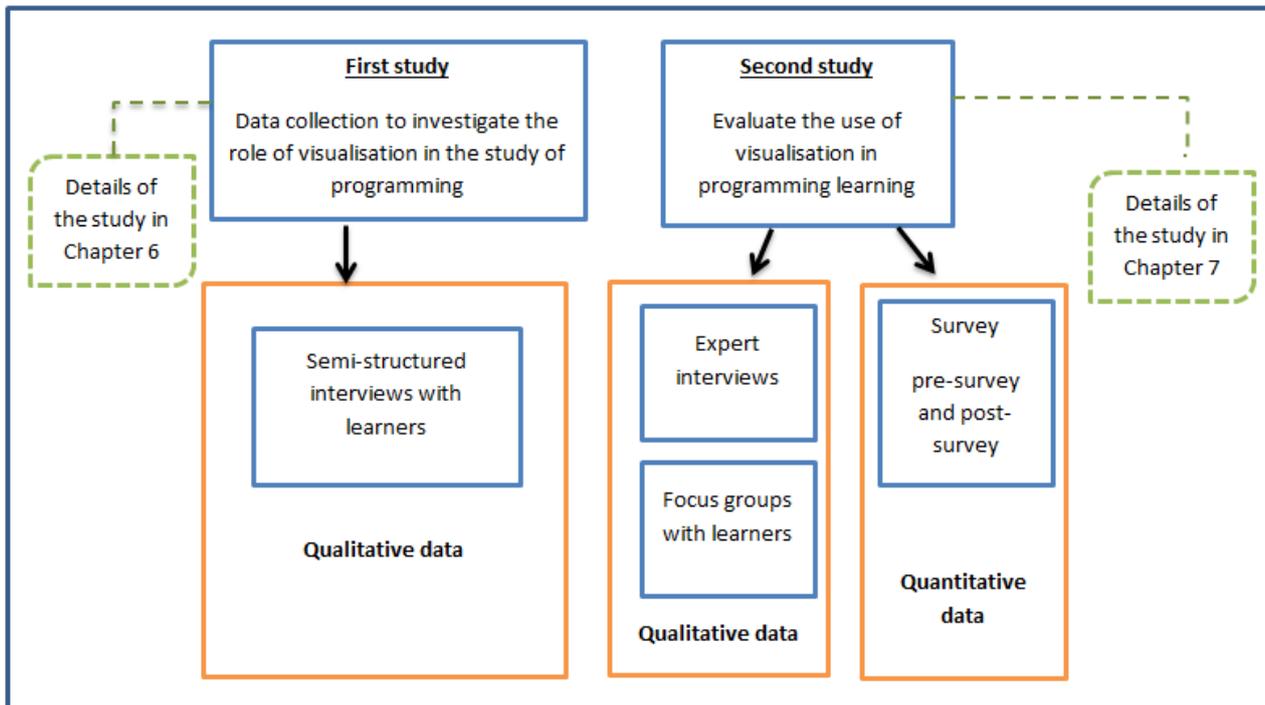


Figure 12. Methodology and Experiments Diagram

A qualitative study is based on the process of understanding social or human factors through the creation of a dynamic, holistic image in a natural setting. The people chosen to participate in this study are likely to be confident with the empirical approach, and the researcher can be put in an active learning role rather than an expert role (Creswell, 1998).

Three qualitative investigation techniques were employed:

- 1- Semi-structured interviews with programmer learners in the data collection stage (discussed in Section 5.4.2 and Chapter 6, Section 6.2.1).
- 2- Semi-structured interviews with experts in the evaluation stage (discussed in Section 5.4.2 and Chapter 7, Section 7.3.6).

3- Focus groups with novices programmers (discussed in Section 5.4.3 and Chapter 7, Section 7.3.5).

A qualitative approach is used to provide a clear view of a problem to help develop solutions, ideas or hypotheses for potential quantitative research. It is also used to drive and identify reasons for opinions and motives. Qualitative information aggregates different methods that include unstructured and semi-structured methods. Some common methods used in qualitative approaches are focus groups (group discussions), personal interviews and participation/feedback. This approach is used when the sample is small and where individuals are selected to achieve a certain quota. A qualitative approach gives more detail and greater amounts of information, as it gives participants more space to explore their ideas (Richard, 2013).

These types of research are used to answer ‘why’ and ‘how’ questions about human behaviour, opinions and experiences. They are useful to describe health, disease, society and cultural and political issues to understand the personal and subjective experiences of individuals and groups. They can be divided into the following: language as a means of discovering the process of communication with specific community groups, describing and translating personal meaning about circumstances and activities and building theory by discovering patterns and links in qualitative information. There are criteria for assessing the quality of a qualitative study, including those relating to good practice in the implementation of a study (methodological rigor) and the standards associated with the honesty of interpretations given (interpretative rigor) (Lune and Berg, 2016).

Each different method is suitable for collecting a particular type of information. For example, in-depth interviews are useful to gather data on history and personal perspectives. Another useful tool is observation, which is appropriate when collecting data on behaviours that occur in their natural surroundings. Still another useful tool is focus groups, which are optimal in reading information about the common cultural norms of a group or to form broad insights on issues of concern about representative cultural groups (Marshall and Rossman, 2014).

In addition to a qualitative approach, this study also uses a quantitative approach, which is commonly used to identify problems by creating numerical information that can be converted into statistical numbers. It is used to identify literature, attitudes and perspectives. Quantitative data collection methods are more general and structured than qualitative data collection methods.

The quantitative method is focused on evaluating a hypothesis consisting of variables measured with numbers based on a numerical analysis of those values. Nevertheless, the application of values to variables was suggested in the evaluation stage of this research to exclude important factors and theoretically remove important elements that may be significant but difficult to measure. Another problem with the quantitative approach is that statistical tests require a precise measurement approach (Galliers and Land, 1987), which is not possible here due to the subjective nature of evaluating visualisation in programming learning. Likert scales (Likert, 1932) (refer to Section 5.3.2) are interesting examples of attributing numerical values to something subjectively calculated, which can then be subject to statistical analysis. In the evaluation process, Likert scales were used as a way of measuring different attitude values, which is further addressed in Section 5.3.2.

In addition, quantitative data collection approaches include many types of screening, such as online questionnaires, paper questionnaires, mobile surveys, face-to-face interviews, telephone interviews and online survey monitoring (Richard, 2013). Therefore, a combination of quantitative and qualitative approaches was used in this study to generate more in-depth and accurate information on the research subject.

The research conducted in this study includes both collecting data to discover the visualisation method characteristics and evaluating the use of visualisation in programming education. To develop the research, quantitative and qualitative methods were used. The research has a connection with building or testing theories (Bryman, 2017). Therefore, the following section discusses the relationship between theory and research to make clear the difference between quantitative and qualitative research (Bryman, 2017).

5.2.1 Deductive and inductive approaches

The deductive approach includes developing a hypothesis from existing theory. It is commonly used with scientific investigation. The researcher using a deductive approach collects data to prove the correctness of any hypothesis generated from the theory. It is mainly associated with quantitative research (Bryman, 2017). In the present study's second experiment, data was collected to test whether visualisation can improve novice programmers' performance.

The inductive approach is concerned with forming a new theory that emerges from the data. To do this, researchers use observation and data collection. This approach is mainly associated with qualitative research (Bryman, 2017). The present study used qualitative data collected from interviews to discover the visualisation characteristics.

5.2.2 The worldview (epistemological) consideration

Philosophical worldviews have been brought to the study. The researcher can design the research related to these assumptions and specific methods or procedures and translate this into practice (Creswell, 2014). The term worldview is defined by Guba (1990) as 'a basic set of beliefs that guide action'. Other researchers call worldviews epistemologies and ontologies (Crotty, 1998). The kind of beliefs held by the researcher will lead them to choosing a quantitative, qualitative or mixed-methods approach for their research. The following discussion will present the common worldviews or beliefs considered in this research—positivism and constructivism.

1. Positivism

Positivism is sometimes called the scientific method or scientific research that determines effects or outcomes (Creswell, 2014). Positivism's assumptions hold more for quantitative research than for qualitative research. Positivism studies problems that reflect the importance of identifying and supporting the causes that influence outcomes, such as found in experiments (Creswell, 2014). It reduces problems to small sets to be tested, such as variables that form a hypothesis or research

question. This quantitative study employed positivism that entails the following (Phillips and Burbules 2000):

The knowledge – It was thought that the evidence regarding the effectiveness of using visualisation in programming learning might be insufficient.

Testing the theory by making claims and generating hypotheses.

Gathering data, evidence and other information that form knowledge – In this study, data was collected to measure the participants' performance using pre- and post-surveys.

Developing a relationship among variables, such as how improvement in the students' scores may affect their confidence in their programming.

Applying validity and reliability in quantitative research is an essential aspect that has been considered in this research.

2. Constructivism

Constructivism, or social constructivism, means that individuals can develop their understanding and subjective meaning in a situation based on experience and social actors (Mertens et al., 2009; Crotty, 1998). Constructivism works mainly with the qualitative approach that requires constructing meanings (Creswell, 2014). One goal of the research relied on participants' (novices and experts) views of and experience with visualisation. The open-ended questions in the interviews and focus groups were too broad to construct the subjective meaning of a situation. Gathering information in qualitative research requires the participants' engagement with their world to understand the context (Crotty, 1998).

Constructivism is distinguished by its focus on how an individual participates intelligently in building knowledge from social construction. It states that knowledge and meaning are built

historically and culturally through social processes and work. Constructivism and social structure are used to try to solve the problems of traditional teaching and learning (Young and Collin, 2004). Constructivism is a synthesis of multiple theories incorporated into one form. It expresses the assimilation of both behavioural and cognitive ideals. Constructivism maintains that learning is the process of constructing meaning; this is how people understand their experience. Constructivism is widely described as an approach to investigate students' level of understanding and to show that this understanding can increase and change to higher-level thinking. Thus, constructivism describes how students can understand material and how to teach material effectively. Considering constructivism as educational theory, teachers should consider what students know and should allow their students to put their knowledge into practice. There are two main methods of the structural perspective: the constructing perspective and the socio-cultural perspective (socio-constructing perspective) (Amineh and Asl, 2015).

Social constructivism is the sociological theory of knowledge that looks at how social phenomena or things of consciousness develop in social contexts. Social constructivism considers that knowledge systems are nothing but human mental structures, and many influences and constraints have contributed to them, such as the politics of governance and ideologies prevalent in society and the religious and moral values carried by individuals who make knowledge. They in turn are subject to laws that safeguard their personal interests and the imperatives of preserving their social status. These laws are then subject to social constructivism (Samy and Robertson, 2017).

Social constructivism also emphasises that knowledge is the accumulation of cognitive particles that accumulate through the ages and complement each other and invalidate each other. It is a product of countless human choices.. Social constructivism is usually placed in opposition to fundamentalism, which sees phenomena in inherent conditions, cross-historical abstracts independent of human rule (Mouzelis, 2016).

The difference between constructivism and social constructivism theory appears in terms of the role of the teacher, the role of the learner, goals, the content of education, teaching strategies, reinforcement processes, the system within the classroom and evaluation. However, both structural and social constructivism theory make the learner the focus of the educational process and his role positive, active and effective, and not only influencing it, and making the teacher plays the role of guidance and guidance, and that it is a source of knowledge, and making learning an active and continuous process of experience, whether direct or not direct, and the idea of linking (new information with old information) as a prerequisite for learning to take place, and learning to discover (i.e. teach the student how to discover the information himself) through research processes such as observation, interpretation, organisation, inquiry, conclusion, creative and critical thinking and problem-solving, search for Knowledge, not imparting it in an integrated manner. In addition to discussion and dialogue is the role of the environment and cooperative social interaction in imparting knowledge to the individual; therefore, the environment must be enriched. There is also the idea of organising the content of the subject matter so that it takes into account the inclinations and individual differences between students (Vall Castelló, 2016).

5.3 Quantitative Method

To evaluate the framework, quantitative analysis was used to measure the students' performance. Quantitative methods are considered tools for statistical data to save time and resources. Statistical data provide clear objectives and guidelines instead of using guesswork (Daniel, 2016).

For a better understanding of the quantitative method, it is necessary to understand variables and types of variables. Variables are the characteristics of an individual or of groups that can be measured or observed and that may vary among individuals or groups during the study (Creswell, 2014). According to Creswell (2014), there are different types of variables. However, the present study focused on three types of variables that play a role in the research: (i) independent, (ii) dependent (iii) and control variables.

- i. Independent variables are variables that affect outcomes. In this study's evaluation of visualisation, the independent variable is the use of visualisation or lab resources when participants attend either a visualisation session or a lab session that affect their scores.
- ii. Dependent variables are the outcomes or results affected by the independent variable. In this case, the scores and confidence are examples of dependent variables.
- iii. Control variables are special types of independent variables that play a role in the quantitative results and influence the dependent variables. In this study's experiment, students' age and experience with programming was considered when collecting the data.

The students were on the same academic level in terms of their programming courses.

Based on the above, we can describe a theory as a 'set of variables, definitions, and propositions that present a systematic view of phenomena by specifying the relations among variables with the purpose of explaining natural phenomena' (Kerlinger, 1979).

The present study used the quantitative method to compare the performance of two groups (a control group and a visualisation group), as discussed in Chapter 7, Section 7.3.1. Each group relied on different teaching methods. The performance of the group that used the visualisation method was compared before and after using the method in a pre- and post-test. Therefore, the statistical data provide an appropriate interpretation of the findings relating to the comparison.

5.3.1 Survey with experiment

A survey is a data collection tool that contains questions to carry out survey research (Pinsonneault and Kraemer, 1993). According to Creswell (2014), 'a survey design provides a quantitative or numeric description of trends, attitudes, or opinions of a population by studying a sample of that population'. After that, the researcher can generalise inferences to the population. Surveys have internal and external validity, which means the findings for the sample group can be generalised to a wider population. Surveys are flexible, which means they can be combined with other methods,

such as interviews and focus groups. Surveys have ethical advantages, as the identity of the participants can be anonymous (Creswell, 2014).

This study used a survey in an experiment that involves two groups to compare the performance of the two groups exposed to two different processes. Each process was considered an attribute for the survey (Krosnick, 1999). The experiments included a questionnaire in addition to a problem-solving question. There are several benefits of having a survey in the experiment. First, the findings can be generalised over the population using either visualisation or regular lab activities in programming learning. Second, the sample included novices and programming students, and the questionnaire may help explore the diverse needs and obstacles faced by novice programmers. Third, the comparison of the findings for both surveys helps in evaluating the framework and provides evidence.

The basic purpose of the experimental design is to evaluate the impact of the treatment or the process on an outcome while controlling any other factors that can affect the results (Creswell, 2014). The survey design, the experiment preparations and the procedures are described in Chapter 7.

5.3.2 Survey measurements

Situated in the framework of the surrounding epistemic approach are the research methods used. Creswell (2014) suggests underpin human or scientific discipline analysis, that of qualitative and quantitative inquiry, the analysis ways happiness to the qualitative realm were thought of to be the most relevant. The investigation in the experiment during the evaluation phase sought to evaluate the participants' confidence in solving the test and in their knowledge of programming topics. It also measured their satisfaction with using visualisation. Confidence and satisfaction are subjective in nature and are typically considered qualitative factors. However, during the evaluation of this research, applying values to variables was considered to eliminate necessary factors that are subjective in nature, such as confidence and satisfaction, which are difficult to measure. Likert

scales (Likert, 1932) should be mentioned here. These provide an example of numerical values being attributed to something that is subjectively measured, which might then be subject to statistical interpretation. A Likert scale is a multiple-item measure of a set of attitudes relating to a particular thing. It is used to measure the level of agreement. The purpose of using a Likert scale is to measure the intensity of feelings about the topic in question or the level of agreement with a statement or set of statements (Bertram, 2006).

Bryman (2017) describes the Likert scale as being one of the most common approaches to investigating attitudes towards specific ideas. It is also one of the most popular formats for assessing usability. With a Likert scale, the list of statements should be piloted and refined previous to using it for a wider audience; however, this was not done in this study due to time constraints (Kent, Hutcheon et al., 2001).

The questionnaire was the vehicle to measure the respondents' perspectives regarding visualisation and provided values for the analysis. Questionnaires often use a rating scale. In this study, statements were created concerning specific components of the visualisation tool that correlative to the necessities. Likert scales are used to measure people's opinions and attitudes. A Likert scale typically has statements such as 'Please state whether you agree/disagree with the following' (Jamieson, 2004). Usually, the format of the level of agreement indicator is a five-point scale going from 'strongly agree' to 'strongly disagree'. This study used a five-point scale to measure the level of agreement among the participants. The five-point Likert scale consists of five answer options, with 5=Strongly agree and 1=Strongly disagree. The intermediate point is 'Neutral', which has to be provided in the questionnaire for the respondents who have a neutral response that is in between or who are undecided. In determining respondents' satisfaction or level of agreement, it is most common to use a 3-point or a 5-point scale. However, the preference in this study was to use a 5-point scale because it more accurately captures the feelings and attitudes of the respondents. For instance, when a person agrees with something, the level of agreement could be very strong or very

weak, so it is better to provide two options, 'Strongly agree' and 'Agree' instead of just the one option 'Agree' found in a 3-point scale. The same idea applies regarding disagreement (Taherdoost, 2019).

This study used a 5-point Likert scale to measure the participants' level of confidence in programming in the first section of the survey. The statements for this were:

- I can use variables
- I can use class inheritance
- I can use a class constructor

A 5-point scale was also used in Section 3 of the survey (Appendix E), where the level of participants' satisfaction was measured. The statements for this were:

- Using the tool helped me to understand classes
- Using the tool helped me to understand objects
- Using the tool helped me to set the variable values defined in the class
- Using the tool helped me to learn Java
- The tool was easy to use

The participants then chose the level of agreement for each statement.

According to Bryman (2012), several points should be considered when constructing a Likert scale.

These are as follows:

- Each item should be presented as a statement and not a question.
- The items should all relate to the same object (learners, students).
- All the items should be interrelated (Bryman, 2012).

The Likert scale has been criticised for a lack of validity and its ability to be reproduced. Therefore, the scale was used solely to offer guideline rankings and to better explore the respondents' feelings regarding visualisation. Respondents' answers may solely be used as reflections of their feelings about the model at that moment in time instead of as a generalisable measure.

Here, the survey in the second experiment did not have a pilot version. This is because the survey was to be presented at two specific times, right after having the new lesson on programming (the pre-survey) and right after the presentation of the visualisation method (the post-survey). Therefore, there was limited time to distribute the survey. Conducting a pilot survey in that timeframe would delay the actual survey, which may affect the overall performance.

5.3.3 Data collection

An experimental design was used for the data collection. According to Bryman (2017), an experimental design involves two groups, an experimental group and a control group. The experimental group (visualisation group) received experimental treatment—in this case a visualisation session—but the control group does not receive the same treatment. The control group relied only on the lecture classes and lab activities for their programming knowledge. Some experiments include a control group that has the same treatment as the experimental group except it does not undergo the experimental process, which in this case is use of the visualisation tool. Therefore, a control group was needed to compare their performance with that of the group that used the visualisation tool in order to see whether the use of visualisation in programming learning has or does not have benefits over attending regular classes and labs. Without the control group, it cannot be determined whether it was the visualisation or other elements that caused the outcome of the experiment.

As mentioned in this chapter, collecting data involves a pre-survey and a post-survey (Appendix E) with an experiment conducted with two groups (here, the visualisation group and the control group). The pre-survey includes a pre-test and subjects that are manipulating the independent variable for both groups and post-survey respondents. Here, the students' performance was the dependent variable that was measured before and after the experiment manipulation.

The survey was the preferred data collection procedure of the study to gather participants' opinions. The economical nature of the survey design and the rapid overthrow in data collection are

the benefits of using the survey in this study's experiment design (Gliner et al., 2009). The experiment preparation and procedures used to collect the data are discussed in Chapter 7.

The cross-sectional method was used in the survey design. According to Creswell (2014), in the cross-sectional design the data collection happens at one point in time. Cross-sectional studies or surveys measure both the exposure and outcome in a sample of the population at a point in time. A major challenge of cross-sectional studies is ensuring that the sample selected and included in the survey is representative of the population of interest.

To apply the cross-sectional design to the pre-survey and the post survey, the design went through different phases:

1. Identify the research question
2. Specify target and accessible population
3. The major challenge of cross-sectional studies is choosing a sample that is representative of the population of interest. This study was careful to choose random participants for both groups (control and visualisation groups). The participants in both groups had the same programming level and background knowledge in programming. However, the sample sections were composed according to their exposure or their attending the visualisation session.
4. Measure variables of interest (performance; confidence of both groups in solving the test and in their background knowledge; and satisfaction).

The present study is considered a correlative study because it measures the effect of using visualisation by following the program's instructions on student performance. It also reflects the cross-sectional research correlation because it shows students and the behaviour of the proposed visualisation program in a specific time period.

5.3.4 Data analysis strategy

The analysis objectives for the quantitative data concern reducing the amount of data collected, finding the relationship between variables and how to develop ways for data presentation (Bryman, 2017). The following steps were considered in the analysis:

- Report the number of participants for each group (visualisation and control groups) in both surveys (pre-survey and post-survey) (refer to Table 5 in Chapter 7).
- Provide a descriptive analysis for dependent and independent variables, such as mean, standard deviation and skewness (Section 7.4.1).
- Compare both groups in terms of variables to test the inferential questions or hypotheses so inferences can be drawn from the sample and applied to a population (Creswell, 2014).

5.4 Qualitative Method

Qualitative methods express meaning, definitions, concepts and descriptions of things. Qualitative methods include, for example, open-ended questions, focus groups, observation and in-depth interviews that expand on quantitative data. The qualitative method provides a broad understanding of the participants' needs and behaviours. Collecting user requirements relies on non-numerical data in the form of words and descriptions that lead to descriptive information (Daniel, 2016).

In the qualitative method, the researcher starts by gathering information from participants from interviews or observations. Then the researcher asks open-ended questions and gathers information that is placed into categories or themes. Finally, the researcher forms generalisations or theories from the categories or themes before posing theories or generalisations from the literature or previous experience (Creswell, 2014).

Different approaches are used to develop theories or generalisations from themes and categories. Grounded theory (GT) is the approach used in this research to generate the theory about using visualisation in programming learning. GT gives a different end point that is grounded in

information from the participants (Strauss and Corbin, 1994). The following section discusses the procedures of GT.

5.4.1 Grounded Theory

The design of GT procedures consists of a constructed set of concepts that provide a theoretical explanation of the social phenomena under study (Corbin and Strauss, 2008). The data from GT can be gathered in different ways, such as interviews, focus groups or observation. GT has specific procedures for collecting and analysing data. The procedures are as follows (Corbin and Strauss, 1990):

1. The process of data collection and analysis should be interrelated. The analysis begins once the first data and information are collected from participants. The benefit of starting the analysis at the same stage as data collection is the ability to use the results to direct the next interview or discussion. It may influence the discussion and prompt new questions.
2. Concepts are the basic units of analysis. The purpose of data collection in the qualitative method is to build theory, which cannot be built without observing activities and actions that have not been analysed.
3. Form the categories. This means developing a higher level of abstraction that is higher than the concepts. It involves grouping the participants' feedback and opinions into categories based on similarities between concepts.
4. Comparison of similarities and differences. In this case, a comparison between participants' opinions and feedback regarding their preference of visualisation tool and the difference between their functionalities was made. Thus, the comparison helped to achieve precision.
5. Writing theories that start generating from the first coding sessions for the categories.

To apply these steps in this research, the steps to analyse the qualitative data gathered from the interviews in the data collection phase were followed:

1. Data collection was achieved through open questions in the interviews. For example, the interviewer asked ‘What are the strengths and weaknesses of the tool?’. All the answers for one question were collected (all the answers grouped by the one question in Appendix F-1).
2. The coding was done for the answers and finding the repeated themes with keywords and phrases. For example, some answers were repeated by more than one interviewee, such as the appearance of the windows, code font and colours (coding with the transcript is in Appendix F-1).
3. All the repeated answers were grouped and categorised and put into subcategories, so any codes that have similarities have the same subcategory. For instance, the answers related to tracing the code and animation are in one group, and any answers related to the appearance of the objects, classes and inheritance are in one group (a table of the subcategories is in Appendix F-1).
4. Each subcategory was grouped into a main category. For instance, tracing the code and animation subcategories are in one category called Expression evaluation. Any subcategories that describe the appearance of the tool are in the category called Interface (a table of the main categories is in Appendix F-1).
5. Finally, we end up with a theory about the strengths and weaknesses of the tools based on the main categories. It was found that some characteristics are more important than others from the students’ perspective. For instance, Expression evaluation was more important than Tool availability. All the findings and characteristics are discussed in Chapter 6.

GT is a research method to generate theory that is ‘grounded’ in data that has been systematically collected and analysed. It is used to detect such things as social relations and the attitude of groups, recognised as social operations. GT was developed by Glaser and Strauss in their study ‘Awareness of Dying’. It is a public methodology for improving theory that is grounded in systematically gathered and analysed data. The advantages of GT (Noble and Mitchell, 2016) are as follows:

- Data collection and analysis happen simultaneously.
- Categories and analytic codes are developed from data. Pre-existing conceptualisations are not to be applied; this is known as theoretical sensitivity.
- Theoretical sampling is applied to refine groups.
- Abstract categories are constructed inductively.
- Social operations are discovered in the data.
- Analytical notes are applied between coding and writing.
- Categories are inserted into a theoretical framework.

Studies that use the GT approach are essentially a phase to conceptual thought and theory structure sort of experimental testing of the theory. Therefore, a qualitative research approach is applied in this kind of research. In particular, conceptual thinking and theory building are why investigators generally employ an inductive, constructivist GT method. It involves the systematic improvement of theory in a social framework and relies upon inductive approaches that are suitable for studies where one of the aims is theory development. Furthermore, the research questions and literature review by themselves support conceptual thinking and theory building rather than empirical testing of the theory. This type of study follows an inductive theory-building approach. Moreover, deductive logic is used to test a hypothesis and deny or adjust a theory based on empirical data, while inductive reasoning requires finding a bound principle and constructing generalisations, relationships and even theories by analysing the data gathered for this purpose. However, inductive operations may still have some pre-existing theories or ideas relating to the problem. Nonetheless, it does not follow to approve or negate the present theories; rather, an effort must be made to create outlines, stability and importance by gathering data (Khan, 2014).

The role of the literature review in GT is to realise theoretical sensitivity; the investigator must start with as few predetermined notions, especially hypotheses, as possible so the data can be as critical to the data as possible. This does not mean that the investigator must start with a tabula rasa, as is

often supposed. It is how prior knowledge is applied that makes the difference; it must be applied to understand the analysis rather than to direct it. Literature can also be used as 'data' and constantly match with the emerging group to be integrated in the theory (Charmaz and Belgrave, 2007).

The suggestion of GT was the reaction to positivism, which was followed by scientific fraud and positivism (positivism: the belief that everything can be boiled down to mathematical evidence and that rationality is all-powerful). Wherever, it is similar to the theoretical orientation of the theory based on the symbolic reaction theory (Priest et al., 2002). Symbolic interactive interaction is evidenced by the fact that 'meaning is negotiated and understood through interactions with others in social processes' (Starks and Trinidad, 2007, p. 1374).

GT is based on two unique characteristics, continuous comparative analysis and theoretical sampling (Glaser and Strauss, 1967). Specifically, data collection and analysis are parallel in GT, and the procedure is neither linear nor sequential. Corbin and Strauss (1990) proposed the following as evaluation criteria for GT: accuracy in the coding and research process, quality of concepts, the methodological relationship between concepts, theoretical density, range of differences and specificity, the importance of theoretical results and theoretical sensitivity .

The focus of GT is the development of theory (Strauss and Corbin, 1994). GT is appropriate when there is no theory or when the theory is too abstract to be tested, but it is not suitable for testing theory or generating knowledge from objective reality (Martin and Turner, 1986; Suddaby, 2006).

GT follows coding processes and focuses on creating relationships between groups or building theories.

The present research used semi-structured interviews with programming novices and students to collect user requirements. The researcher relied on GT to make the categories of visualisation in programming learning from the data collected in the interviews with the programming students (Chapter 6). Moreover, the researcher conducted in-depth interviews with experts (programming

lecturers and tutors) to evaluate the framework. Focus groups were also conducted for a better understanding of the quantitative data collected for the experiment.

5.4.2 Semi-structured interviews

A semi-structured interview is a qualitative method using open questions that gives the interviewer a chance to explore the subject. Semi-structured interviews are used to understand how to interpose work and how the questions could be improved. They also allow respondents to suggest issues that researchers may not have considered (Blandford, 2013).

Interviews are data collection tools that can be used in qualitative research to investigate individual respondents' views, perceptions, values and motives. They provide a deeper understanding of a specific research topic and are best suited when individual participants need to give detailed insights, particularly when quantitative data (mainly collected from questionnaires) is not considered enough (Brinkmann and Kvale, 2014). There are three types of research interviews:

- **Structured interviews.** These take the form of a questionnaire with a set of predetermined questions administered orally. Such questions are put to the respondent with little or no variance and with no intention of continuing with follow-up questions for answers that may need further elaboration. Structured interviews are therefore fairly quick and easy to conduct and can be useful if clarification is needed for specific questions or if the respondents have issues with literacy. Generally, if further depth is required, they are of little use.
- **Unstructured interviews.** Generally, these do not follow any particular principle and are carried out with little or no coordination at all. They usually begin with an opening question such as 'Please explain your experience of doing this to me' and progress based on the response. Unstructured interviews are usually time-consuming and can be hard to handle. As there is a lack of predetermined questions, there is little guidance on what to speak about. Unstructured interviews require excellent interviewer skills, particularly in areas where virtually nothing is known.

- Semi-structured interviews. These fall between the categories mentioned above. Usually, semi-structured interviews start with several key questions that help define the areas to be explored and allow both parties to deviate if an idea or opinion needs to be pursued in more detail. This offers more versatility compared to structured interviews because it requires more knowledge, exploration and elaboration. At the same time, concepts that have not been articulated before can easily arise from the respondents in the same way as in unstructured interviews.

Preparing an interview guide is an important first step in conducting an effective interview (Gill et al., 2008). The following important points were taken into account during this process:

- The interviewer must introduce himself and explain the purpose of the interview so that the interviewee is fully aware of the intent of the study.
- The interviewees must be ensured of the privacy and anonymity of the interview.
- Prepare the questions, taking into consideration that:
 - Questions should be applicable to the subject of the research.
 - Questions must obey a sequence of logic.
 - Questions should be structured in such a way that one can easily navigate to another subject.
 - Questions should be clear and easy to understand, taking into account the interviewee.
 - Questions do not give rise to a specific response but rather encourage people to feel free to offer their own honest answers.

The type of interview chosen in this study was the semi-structured interview because the strengths of both structured and unstructured interviews are mixed. Most precisely, the participants were asked a few key questions about their occupation, job description, job responsibilities and years of experience at the beginning of the interview. Then the interview continued in a more open and unstructured way on the topic of learning programming using visualisation.

Semi-structured interviews were used in the data collection phase to obtain more information by asking open-ended questions. Semi-structured interviews were also used with experts for the framework evaluation to give them the freedom to explore issues further. In general, interviews ensure mutual understanding of the subject. In this case, the interviewer guarantees that interviewees understand the subjects and may paraphrase the questions. As a result, accurate data can be collected from the interviews. Moreover, the data obtained from the interviews can be recorded. Thus, the researcher can review the data many times to ensure more accurate results (Alshenqeeti, 2014).

5.4.3 Focus groups

A focus group is a group discussion on a particular topic that is guided, monitored and recorded by the researcher (Gill et al., 2008). It is defined by Krueger and Casey as ‘carefully planned series of discussions designed to obtain perceptions on a defined area of interest in a permissive, non-threatening environment’ (Williams and Katz, 2001). There is no specific rule specifying the number of participants in a focus group, as researchers in this field have argued for different sizes ranging from approximately 4–12 participants (Masadeh, 2012). The session duration has been specified in some of the sources to be 60–90 minutes (Evaluation Research Team, 2008). The questions in a focus group should usually follow a sequence, starting from a broad discussion and then narrowing to specific questions and ending with exit questions. The starting questions should prepare participants for an in-depth discussion. Following that are exploration questions that should engage the participants in the in-depth discussion. Finally, the researcher should ask exit questions to add any further comments regarding the topic (Eliot and Associates, 2007, 2005).

5.4.4 Data analysis strategy

The qualitative data was analysed from the first coding process through initial and final coding. This type of coding was chosen to find the keywords and group the features. Text analysis was used to conduct the first cycle of coding to determine the phrases that were common amongst

interviewees. Phrases appeared as word clouds, which were analysed and encoded with suitable category labels. Next, a second cycle pattern coding method was used to recognise similarly coded data and to summarise it further into subcategories or to consolidate it. Finally, the findings were narrated, as they related to the implications of the study (Corbin and Strauss, 2008; Saldaña, 2016). As an example of the analysis for the interviews, a question and the answers of one of the participants are shown in Table 3. The answers have been analysed to form subcategories (refer to GT in Section 5.4.1 and Chapter 6) in the first coding process for the text analysis. A second round of analysis was done to group the subcategories according to similarities to main categories. The complete analysis of the interviews is in Appendix F (F-1 and F-2).

Table 3. Coding framework table for interviews

Interview transcript	First coding process
<p>Interviewer: ‘What are the strength(s) and weakness(es) of the Jeliot\Online Python Tutor\Visual Logic tools?’</p> <p>P1: Jeliot: ‘Like: the appearance of execution which is next to the code directly Dislike: 1- the tool cannot go back to previous steps or forward to next steps (no control on the execution steps) 2- in case of representing the classes and inheritance, it was not clear because the classes cascaded and not represented as hierarchy’</p> <p>Online Python Tutor: ‘Dislike: 1- she does not like the online tool because of any Internet issues or problems 2- the execution process like the arithmetic and logic operation is not clear enough since there is no expression evaluation and the tool produces the final result without any details Like: she likes the control of execution (the existence of next and back buttons)’</p> <p>Visual Logic: ‘It is an advanced tool that could be used to learn the flow of programming but not how to write program since there is no code to be blogged and evaluated it is suitable to learn the structure of programming and program semantics, nothing about memory referencing.’</p>	<p>Windows preference Execution preference Object-oriented representation Tools availability Expression evaluation preference Tracing the program code</p>

The same strategy was followed to analyse the focus group discussion; however, the answers of different students in different discussion groups were grouped according to the type of question. For instance, for the question ‘How did you find the animation? What things did you like and dislike?’, the answers were grouped under the ‘animation’ category. The following is a sample of the answers; the rest of the discussion transcript is in Appendix F, Section F-3.

B.1.1 ‘Better than using the manual trace’

C.1.6 ‘Prefer the tracing that shows the steps of code flow’

C.1.2 ‘Great for tracing and having an application for the tool would be easier for students to use rather than a website.’

Note that the participant was given a code, for example C.1.6, which means (programming problem code. group number. participant number); refer to Section 7.3.5.

5.5 Ethical Issues

Ethical issues concern behaviour and ethical treatment and the introduction of ethical principles in dealing with right and wrong. Morality conveys moral grace, which conforms to the principles of true universal behaviour, in particular the principles of collective practice. Therefore, research requires adherence to universally agreed ethical standards(Resnik et al., 2015).

Ethics in scientific research reinforce the core values of collaborative action, such as trust, accountability, mutual respect and justice. Cooperation and coordination between many individuals in different disciplines and organisations are typically involved in research. Moreover, as many ethical designs help to ensure that research is accountable to the public, research focuses on different ethical rules. In addition, the rules of writing, copyrights, patents, data distribution strategies and privacy aim to safeguard intellectual property while supporting cooperation. Ethics are concerned with avoiding harm and achieving positive results. In addition, applying appropriate

ethical attitudes helps eliminate damage. Thus, it is important to be ethical to protect the participants and the research (Resnik et al., 2015).

According to Tadajewski (2004), a researcher must act appropriately with respect to the rights of anyone who participated in the research and thus became a subject of the work or who was consequently affected by it in order to make the research valid. Therefore, ethical practices are vital for social researchers. Tadajewski (2004) gives two distinct principles that emphasise just like the method. First, describing the interests of all reactions is critical, and therefore there should be no deterioration in the form of the people whose data has been collected between the beginning and the end of the research.

For this study, part of the research plan necessitated that ethical approval be obtained from the Ethics Committee of the School of Computing at Plymouth University. It showed bias and respect for ethical considerations. The researcher told the participants that the data is only for scientific research purposes and that the privacy of the data would be maintained. In addition, in order to validate the validity and integrity of the study, the researcher made the questionnaire simple and easy to read so it could be easily answered by study participants. The expressions in the questionnaire were made suitable for participants at several levels. The interview questions were straightforward, clear and uncomplicated. A copy of the documentation and the Ethical Approval Application are included in the appendices (Appendices B and E).

Prior to any interview or experiment or focus group, an information sheet and a consent form were given to the participants. The information sheet has a description about the topic of the research, the aim of the research, a description of the overall procedure and the time needed to accomplish each task. It also states the benefit of the research, the right to withdraw and how the researcher will ensure data confidentiality. Copies of the information sheets for the data collection interviews and evaluation experiment are in Appendix A and Appendix D, respectively.

Participants had to sign a consent form that includes a description of the research, the purpose and the objectives. In the consent form, the participants are guaranteed the freedom to withdraw from the research at any time and anonymity. They are also asked to give permission for audio recording during interviews and focus group discussions. The researcher also guaranteed avoiding any kind of risk. A copy of the consent form for the data collection interviews and the evaluation experiment can be found in Appendix A and Appendix D, respectively.

Regarding sending the interview transcript to participants for approval, this was not done because the interviews were recorded and any difference in the transcripts would be found on the recorder.

5.6 Conclusion

The chapter presented the research methods used for both collecting user requirements and framework evaluation. Qualitative and quantitative methods were employed to achieve better data that contributes to the continuous investigation and improvements in the use of visualisation in programming learning for novices. The next chapter presents a study that helped determine the aspects considered in developing a pedagogical visualisation environment from the novices' perspective.

Chapter 6 INVESTIGATING THE ROLE OF VISUALISATION IN THE STUDY OF COMPUTER PROGRAMMING

6.1 Introduction

Visualisation tools have been introduced into programming education at many academic institutions. Some of these tools achieve the intended goal and make tangible contributions to programming education, at least from the tool developers' perspective (Kasurinen et al., 2008). However, visualisation tools need to be evaluated continuously to have maximum benefit.

This chapter presents research to investigate some preselected and sample visualisation and memory reference tools. The purpose is to gain an understanding of the requirements of novice programmers. Understanding novice programmers' requirements can help in finding solutions to the challenges they face in learning programming. Semi-structured interviews with students and novice programmers were conducted to collect their feedback and opinions about using the visualisation tools while learning to program. By collecting the students' opinions and analysing them, the study can conclude whether visualisation could be better exploited and can determine the features that need to be improved. This chapter also discusses the presentation and use of visualisation categories in the Visual Code Flow tool, as well as the interface components and their functionality.

6.2 Data Collection

This research aimed to evaluate the current visualisation tools used to support novice users learning to program. In particular, the focus was on identifying the strengths and weaknesses of these tools. This research was conducted with CS students from different levels who are either in their second year of studying programming or higher to gather information about their experience in using visualisation tools, to understand their needs better and to gather their opinions about these tools.

6.2.1 Interviews

The study included semi-structured interviews with 20 participants over 22 years old. The participants were either undergraduates or students who had recently graduated. During interviews,

it was verified that each participant had studied programming concepts in depth, including the core aspects of the use of a loop, object-oriented programming and the calling of procedures. The interview aimed to elucidate the novice programmers' experience and to establish their background in using visualisation tools to support the process of learning to program.

Three tools were presented to the students as a sample of available visualisation support tools. These tools were Jeliot, Online Python Tutor and Visual Logic (as discussed in Section 4.4). The students discussed the tools and their usability and determined each tool's strengths and weaknesses. The tools were compared based on their usefulness in solving programming problems.

The programming problems were selected based on the threshold concepts, including object-oriented programming, loops and calling procedures (as discussed in Sections 3.4 and 3.6). Opinions were also obtained regarding their assessment of the best and worst tool. Interview questions can be found in Appendix B, and the transcript can be found in Appendix F, Section F-1.

Some of the details of the interview procedure are as follows:

Step 1: The students were welcomed as they arrived (5 minutes)

- The interviewer asked the students if they were willing to participate in the session.
- If they were, the interviewer asked the participants' permission to audio record the session.
- If they were still willing to participate, the interviewer explained that they could withdraw

from the study at any time and that they could subsequently withdraw their data from the study and how to do so.

- The interviewer explained that all personally identifiable information would be held separate from the core dataset and that it would only be used if the student gives consent.

- Once the students were briefed and agreed to participate, the interviewer asked them to sign the consent document before proceeding. The interviewer provided them with an overview of the session activities and the duration. This gave a breakdown of how long each step should take. This

ensured that the students were fully informed about the process. See Appendix A for the consent form and information sheet.

Step 2 – The students were given an introduction (5 minutes)

- Initially, the students were introduced to the subject in general and to the objectives of the study. This gave the students the background they needed to do the tasks.

Step 3 – The participants attempted tasks 1, 2 and 3 (45 minutes for all). See Section 6.2.3 and Appendix B for more details on the tasks.

- o First: The students' background in using any tool for the purpose of tracing and observing the program behaviour during their programming study was determined. The discussion expanded to their experience and whether they benefit from the tool(s). The features and drawbacks of the tools were discussed from the students' perspective. This session was designed to last no more than 10 minutes.

- o Second: The second part of the discussion involved evaluating the three selected tools. The tools have a common methodology, which is the program visualisation method. However, each tool has its own features. See Section 6.2.2 for details on tool selection. The evaluation was done by presenting three activities that students usually get stuck on. This session was designed to last no more than 25 minutes.

- o Third: The third part of the discussion concerned the comparison of the tools in terms of solving the three kinds of activities. This session was designed to last no more than 10 minutes.

Step 4 – Final feedback and thanks (5 minutes)

- After all the tasks were completed, the students were asked to comment more generally about the tools. For example, they were asked their opinion on the method that these tools follow and whether they found it useful and effective.

- Because it is important that the students be happy with this process, their right to withdraw their data was reaffirmed.

- It is also important to give the students the right to ask questions about the research and the process. Thus, they were made to feel valued and an important part of the process.
- Finally, the participants were thanked (handing over the thank you card with their code on it) and asked if they would be willing to participate again at a later stage.

Ethical approval for recording the sessions was specifically sought so that note taking would not interfere with the process, particularly the discursive process when the participants were describing the problems they experienced when using the tools.

6.2.2 Tool selection

As most of the participants in the study were expected to have little or no experience using visualisation tools, a sample of the visualisation tools presented in Section 4.4 was selected. The tools all shared the common method of using memory referencing to show the effect of execution on each line in the code. However, each tool had its own specific features. Tool selection was done carefully to make a comparison of students' viewpoints clearer. The tools differ in many aspects, such as in the way they trace the code, how they present the output, how they create error explanations, the programming languages they support and whether they are available online or offline. Tool selection was also based on the students' experiences. The tools were designed for novice programmers, so they were suitable for the participants in this study and required no training or special skills.

6.2.3 Study tasks (threshold concepts)

The 'threshold concepts' in numerous disciplines are gaining increasing acceptance in the literature on teaching and learning in higher education and were recently advocated in Erik Meyer and Ray Land's *Overcoming Barriers to Student Understanding: Threshold Concepts and Troublesome Knowledge*. The first objective of this piece is to challenge this trend by viewing that 'threshold concepts' outlined as are unidentifiable even in principle and by highlighting that different authors understand 'threshold concepts' in different and incompatible ways. The second is to reformulate

the notion in a superior way, a way that appears to be correct to the determined of most of its users, and to investigate numerous consequences for related empirical research that have not yet been recognised. Broadly, a threshold concept is described as ‘akin to a portal, opening up a new and previously inaccessible way of thinking about something’, in alleged contradistinction to a ‘core concept’.

In the present work, the author collected data in the first research phase based on three programming problems taken from the literature discussed above that are threshold concepts—loops, calling methods and inheritance. Many studies (Boustedt et al., 2007; Bühlmann, 2011; Eckerdal et al., 2006; Sanders et al., 2008) have shown that loops, object-oriented programming and the calling of procedures are threshold concepts that students find difficult to understand. Therefore, these core threshold concepts are embedded in the three tasks investigated in this study (Chapter 6).

In the second research phase, the use of visualisation in programming learning is evaluated (Chapter 7). The loop problem was changed because the findings in the data collection phase (Chapter 6, Section 6.3.9) show that the existing visualisation tools, especially Jeliot, were preferred by all the participants. Therefore, for the second study three programming problems were adopted (calling a method, classes and objects and class inheritance). After reviewing the previous theoretical literature, the choice of threshold concepts was changed due to their relevance to the current study, where the concept of inheritance is one of the processes in which an object acquires the characteristics of another object. A base type is derived, taking into account all the fields and functions of the base type members. Inheritance is most useful when one needs to add functionality to an existing type. For example, all .NET classes inherit from the system. Object class, so that the class can include new functions as well as use the functions and properties of the existing object class as well. Added to that, object classes facilitate rapid development because they reduce the semantic gap between code and users. System analysts can speak to both developers and users using essentially the same vocabulary. Object classes often facilitate rapid development because most

object-oriented environments come with powerful tools for debugging and testing. Instances of classes can be checked at runtime to check system performance, as expected.

When one starts programming in Java, there are many new concepts to learn. There are classes, methods, exceptions, builders and variables. The Java method is a collection of statements that are grouped together to perform an operation. When one calls the `System.out.println ()` method, for example, the system actually executes several statements to display a message on the console. To use the method, it should be called. Method calls are made directly to the class and cannot be called in the instance instances. Static methods are often used to create utility functions. The calling method is simple. When the program calls a method, the control in the program is moved to the method that is called.

Java is a prototype language, and each object in Java has a hidden internal property called `[[Prototype]]` that can be used to extend object properties and methods. Classes in Java do not offer additional functionality and are often described as providing ‘syntactic sugar’ to prototypes and inheritance in that they provide a cleaner and elegant syntax. Because other programming languages use classes, the class syntax in Java makes it easier for developers to move between languages.

The problem areas considered in this study are the three programming concepts described earlier as threshold concepts. Many studies (Boustedt et al., 2007; Bühlmann, 2011; Eckerdal et al., 2006; Sanders et al., 2008) have shown that loops, object-oriented programming and the calling of procedures are threshold concepts that students find difficult to understand. Therefore, these core threshold concepts were embedded in the three tasks investigated in this study.

1 Task 1 (the loop task)

For this task, Program 1 was coded (Appendix B). Initially, this program populated an integer array data structure of five elements with predefined integer values. The overall aim of Program 1 was to evaluate the average value of the integers stored in the array. To achieve this, Program 1 established

the number of elements in the array. It used a 'FOR loop' to calculate the total of all the values. Subsequently, the average was calculated.

2 Task 2 (the object-oriented program)

For this task, Program 2 was coded (Appendix B). The task used the inheritance from the superclass (polygon) to calculate the area of the subclass (square). The main program defined an instance of the class square and passed the length of the side to the method 'get_area()', which is inherited from the superclass (polygon). The area was calculated by the inherited method 'get_area(),' and the result was inherited by the subclass (square). Finally, the result printed in the main program was an element of the instance from (square) class.

3 Task 3 (the procedure call)

For this task, Program 3 was coded (Appendix B). Task 3 was used to highlight the difference between passing parameters by value and passing by reference. The main program defined a *variable* (parameter) as a global variable and sent its value to the procedure (change_value) as a pass by value and not by reference. The value of the parameter was changed locally in the procedure, but the change did not affect the parameter in the main program. The variable was printed in the main program to show whether or not the value of the variable had been changed.

6.3 Data Analysis

The researcher's intention was to collect feedback about the existing visualisation frameworks to determine their benefits and to develop a new framework. The qualitative data supports the research in focusing on the basic features and aspects that novice students need regarding educational tools. The strategy used is described in the research methodology in Chapter 5 (Section 5.4.4).

6.4 Findings

During the semi-structured interviews, a substantial amount of data was recorded from the participants, including what they liked and disliked about each tool. During the data analysis, it was realised that most of this feedback was concentrated on eight core categories of support tools. All

the categories were features of the presented tools. Therefore, the researcher adopted this list of eight categories as a means of categorising the data analysis. Following is the list of categories:

1. Control of the execution of the code
2. Availability of the tool (online or offline)
3. Error explanation
4. Interface/usability of the tool
5. Programming languages supported
6. Expression evaluation
7. The representation of the class hierarchy
8. Maintaining an event history

6.4.1 Controlling the execution of the code

The analysis stage of this study revealed that the manner in which the participants controlled the execution of code was important, as the interviews indicated that the students clearly understood the importance of how the visualisation was being controlled. Of the students, 90% (18 out of 20) reported that they preferred to have precise controls for the line-by-line execution of the code and its visualisation.

When this study was designed, careful attention was given to the selection of which support tools to include. The outcome was that the tools chosen represented various mechanisms for the control of execution. Further, the participants realised there were different kinds of execution controls, and they subsequently used them to compare the tools.

During the interviews, the participants used various phrases to describe the control of execution. These phrases included animation, control buttons, execution speed, tracing and visualisation. For instance, Participant 5 complained about the inability to have full control of the execution, such as the ability to go back and forth with each statement.

Participant 5: *'The execution control is not enough; I prefer that I can control the execution more, for example, going back and forth for each statement.'*

Conversely, Participant 19 appreciated the animation and used it to visualise the execution.

Participant 19: *'I like the animation used to visualise the execution.'*

Regarding the speed of the execution and the ability to control it, Participant 7 suggested a slower speed.

Participant 7: *'I want to slow down the execution speed.'*

Furthermore, Participant 11 suggested an overall more flexible execution process, which gave more control to the user.

Participant 11: *'I like the control buttons that make the control of execution more flexible.'*

It seems that the participants liked animation in general. However, adding more control for the animation, such as pause, rewind and going back and forth, would make the tool more useful and make it easy to track the program execution.

However, these preferences varied. Of the 18 participants, seven appreciated the code animation, which had few controls. Conversely, five preferred the user to have complete control using the control buttons. The remaining six participants (30% of the participants) suggested that a mixture of both methods would be the optimal solution. This was based on the fact that the type of control required was about the same length of code involved, and the users were familiar with the code.

6.4.2 Availability of the tool

The data analysis revealed that establishing whether the tool was available online and offline is a relevant aspect for participants. It was clear that this characteristic would influence their choice of tool. However, for some participants, the tool's online availability was a genuine concern, as shown in the feedback below.

Participant 1: *'I do not like the online tool because of any Internet issues or problems.'*

Participant 7: *'I don't like the online tools to avoid the connection problems.'*

Conversely, for others online availability was assessed as a positive aspect of the tool:

Participant 17: *'I like that the tool is an online tool.'*

Thus, the answers given by participants varied regarding this characteristic. In total, 16 of the 20 participants highlighted this characteristic as important. Of these 16, three expressed a preference for the tool to be downloaded to their devices. They explained that their preference was because of concerns about their Internet speed and connection. Conversely, nine participants preferred the online mode, explaining they would have access to the tool at any time and anywhere. The remaining four participants preferred the option of a mixed mode (combining both online and offline), where the user could choose the mode required.

6.4.3 Error explanation

The data analysis revealed that the explanations of errors and support for debugging are important aspects for participants. The participants reported that they required (i) support for finding an error, (ii) a meaningful explanation of the error and (iii) a suggestion of a suitable means to correct the error. They noted that ambiguity in the explanation of the error might impede their understanding of the cause of the error in the code, as shown in the following feedback.

Participant 10: *'The error explanation was not good.'*

Participant 17: *'The correction suggestion for the error is not helpful enough.'*

Subsequently, this may have a negative effect on their learning. When the participants were asked about the reason for their opinions, they said they could not understand the cause of the error and the suggestions were ambiguous. Therefore, clarity in error explanation is a high priority. In total, 13 of the 20 participants highlighted this characteristic as important. All 13 participants expressed a preference for a support tool that explains the error in detail and provides various ways to correct the error.

6.4.4 Interface/usability of the tool

The data analysis revealed the style and appearance of the interface were also important aspects for participants. Of the 20 participants, 13 (65%) reported on the interface and its usability from different perspectives. Note that they used different terms to describe their opinion of the interface of each tool, including font, colour, window and ease of use. In conclusion, it is clear that the colour, the font and the windows' appearance played a significant role in attracting the participants to the tool.

The participants commented on the font type, size and colour used in the text.

Participant 9: *'The code font is small.'*

Some of the feedback gathered from the participants was about how the windows and how the tool presented the final output:

Participant 1: *'I like the appearance of execution, which is next to the code directly.'*

The participants also gave various comments about the use of an indicator icon to point out the statement currently being executed.

Participant 2: *'[I like] the use of green and red pointers (arrows) to indicate the statement execution whether it is under execution or will be the next statement to be executed.'*

This opinion was reinforced by another participant.

Participant 6: *'I like the use of red and green arrows as an indication of statement execution.'*

6.4.5 Programming languages supported

Next, the data analysis revealed that both the number and range of programming languages supported by a tool is important to participants. Again, the participants' feedback was mixed. Some participants preferred to use multi-programming language tools, while others expressed a preference for a bespoke tool for each programming language.

Participant 10: *'I don't like that Jeliot supports only one programming language.'*

Participant 18: *'I like the Online Python Tutor tool because it supports more than one language.'*

Of the 20 participants, nine (45%) recommended a tool that could be used for multiple programming languages. The remaining 11 (55%) reported that the choice of programming language was not a huge issue to consider and that supporting only one language is enough when it comes to using the tool.

6.4.6 Expression evaluation

The data analysis revealed that details on how and when an expression is evaluated are important aspects for participants. The participants reported that they had not recognised when expressions were being evaluated, although there was various feedback on this characteristic. For example, in Program 1 where the condition of the loop was evaluated, the participants wanted to 'see' the working out of evaluating the condition of the loop at every iteration.

Participant 4: *'The design needs to be developed, similar to the manual tracing and the absence of statement expression.'*

Participant 18: *'One of the strengths of the tool is the existence of expression evaluation.'*

Participants reported that this evaluation of expressions was primarily used to understand 'what's going on'. This is a vital aspect for increasing learners' comprehension.

In total, 5 of the 20 participants highlighted this characteristic as important. The participants reported that they relied on the expression evaluation for evaluating the condition of the control statement. All the participants who reported the use of the expression evaluation agreed that they liked it because it is similar to what they had done on manual tracing. Therefore, there was a requirement to present the user with the automated version of what they would otherwise do manually.

6.4.7 Representation of class hierarchy

The analysis stage of this study revealed the manner in which the tool represented the class hierarchy was critical to users. Task 2 presented the participants with code from an object-oriented program. The program is an example of the concept of inheritance, as it inherits variables and methods from a predefined superclass. The participants assessed how well each tool represented the classes and their hierarchy. They also reported how much the tool aided their comprehension of the classes in the code. For example, one participant made the following statement about Jeliot, with which another participant agreed:

Participant 5: *‘In case of representing the classes and inheritance, it is not clear because the classes cascaded and were not represented as hierarchy.’*

Participant 3: *‘There was no weakness in the tool’ but then went onto to highlight one exception: ‘its way to represent the class hierarchy and the variables and methods which are inherited or private, all of these were not satisfied.’*

6.4.8 Maintaining an event history

The final characteristic of the support tools identified as important by the study participants was the tool’s ability to record the events (history) of everything that occurred during execution. The participants used the phrase ‘save history’ to describe their requirements for the system to automatically generate a list of events, which took place throughout the whole process of execution. Participant 2 complained about the tools being unable to save the history. Participant 3 supported this by indicating that they would also like a tool that kept results for each execution and that did not omit events. Based on this feedback, it is clear that users place a high value on having this history of events to trace through after execution.

6.4.9 Tool comparison

The final part of this study focused on comparing the three tools. This comparison was based on how each tool presents the execution of the three tasks (mentioned in Section 6.4.2) and on the participants' opinions about which tools they preferred.

The students excluded the Visual Logic tool from the comparison, as it received a lot of negative feedback on its usefulness in supporting students learning to program. It was a clear outlier that they were not interested in pursuing. The reason for this negative feedback was that Visual Logic does not include any program code and relies too heavily on tracing through the flowchart.

Regarding Task 1, solving the FOR-LOOP task, all 20 participants preferred the Jeliot tool. The reason for this unanimous decision was that Jeliot had 'expression evaluation.' This 'expression evaluation' feature shows the evaluation of the condition of the loop for each iteration. Further, it shows the loop counter and the loop body.

Participant 3: *'I choose Jeliot, in case I am a novice programmer because it has the feature which is expression evaluation.'*

Participant 4: *'Jeliot was clearer than the other tool; I like how it shows the expression evaluation to clarify the loop counter, loop condition and the body of the loop.'*

These opinions were reinforced by other participants:

Participant 7: *'Jeliot was clearer because it represents the dynamic change on the loop counter and condition.'*

Participant 15: *'Jeliot is better because the loop was clearer in how the counter changed each time and how the loop conditions checked every time.'*

Regarding Task 2, based on object-oriented programming, seven participants expressed a preference for using Jeliot. They attributed this choice to the fact that cascading the classes is more

effective than representing them in a hierarchy. This is particularly the case when there are numerous classes where the screen space will be insufficient to demonstrate the class hierarchy.

Participant 2: *‘Jeliot is better in case of long code and plenty of classes because the class cascaded, which makes space add classes.’*

Participant 4: *‘I prefer Jeliot because of how representing the classes were clear specifically if the number of classes was large.’*

Conversely, most of the participants (11 of 20) preferred the Online Python Tutor tool, as it represents classes sequentially. Therefore, the participants reported they ‘could see’ the variables and methods of each class.

Participant 10: *‘Online Python Tutor because the classes were presented sequentially, so it is clearer than representing the classes on Jeliot.’*

Participant 18: *‘Online Python Tutor is better because I like how it draws boxes sequentially for objects and uses arrows to represent the references.’*

Only two of the participants were impartial. Those two participants stated that both class representations were clear and helped them understand the class inheritance equally.

Finally, regarding Task 3, the researcher asked the participants to compare the tools when calling procedures and passing parameters. The majority of the participants, 18 of 20, agreed that the Online Python Tutor tool was best. The reason they gave was that the process of parameter passing was significantly clearer than in the Jeliot tool.

However, there were different opinions based on the clarity provided by the Online Python Tutor tool.

Participant 2: *‘Online Python Tutor is clearer on how the transition did from the main to the procedure and how it affects the value of parameters.’*

Participant 11: *‘Online Python Tutor was clearer when calling procedures and giving the returning value; it always writes what the return value is from any procedure even if it is “void”.’*

The remaining two participants said they understood the task equally when using both Online Python Tutor and Jeliot.

6.5 Visual Code Flow Tool

From the findings in Section 6.3 collected from the semi-structured interviews, the author started developing the Visual Code Flow tool (Appendix C). This is a visualisation tool that can trace a program line by line and show the effect of the code in the memory during execution. It shows how variables, classes and objects are created and how they are changed, as well as how methods are initialised and called. Recalling from Section 6.3, the findings were categories that were mentioned during the interviews. Most categories that were mentioned were considered in developing the Visual Code Flow tool. These are:

1. Control of the code’s execution
2. Availability of the tool (online or offline)
3. Interface/usability of the tool
4. Expression evaluation
5. The representation of the class hierarchy

Control of the code’s execution

Control of the execution was added as a feature in the Visual Code Flow tool because based on the data collected in the semi-structured interviews 90% of the students (18 of 20) reported that they preferred to have precise controls for the line-by-line execution of the code and its visualisation. Of these 18 participants, seven appreciated the code animation, which had few controls, and five preferred the user to have complete control using the control buttons. The remaining six participants

(30% of the participants) suggested that a mixture of both methods would be the optimal solution. Therefore, in the Visual Code Flow tool the author offered control buttons that can give the user the option to either play and watch the animation or to control moving through the lines in order to go line by line.

Availability of the tool (online or offline)

The Visual Code Flow tool was made available online and offline through the application because based on the data collected in the semi-structured interviews it was found that 9 of the 20 participants preferred the online mode, explaining they would have access to the tool at any time and anywhere. Three expressed a preference for the tool to be downloaded to their devices. They explained that their preference was because of concerns about their Internet speed and connection. The remaining participants preferred the option of a mixed mode (combining both online and offline), where the user could choose the mode required. For the user's convenience, and based on the feedback collected from the participants, the tool works in both online and offline modes.

Interface/usability of the tool

The tool's interface and its usability are features that were considered in its development. The ease of use was considered so that novice programmers could benefit from the tool. The tool components and how they are used are discussed in Section 6.6.

From the investigation and the analysis of the semi-structured interviews, it was found that 13 participants (65%) out of the total of 20 reported on the interface and its usability from different perspectives. They mentioned the font type, colours, and window parts.

Expression evaluation

The expression evaluation refers to details of what happens to variables during execution. The expression evaluation suggests the characteristics that contribute to a better understanding of what

happens in the memory. The visualisation method is all about the expression evaluation, as it visualises the change while the program is in the execution process. Of the 20 participants, five highlighted this characteristic as important. The participants said that they liked the existence of expression evaluation because it is similar to what they had done on manual tracing. Therefore, there was a requirement to present the user with the automated version of what they would otherwise do manually.

Representation of class hierarchy

Representation of class hierarchy is available in the case of using classes and objects. The tool represents each class with its attributes (variable and methods) and links each class with arrows to illustrate the relationship between the other classes. In this case, the tool uses the icons to demonstrate the hierarchy. The analysis stage of this study revealed that the manner in which the tool represented the class hierarchy was critical to users, that is, either sequential or cascading representation for the inherited classes. Therefore, most of the participants (11 of 20) preferred the Online Python Tutor tool, as it represents classes sequentially. The author therefore used sequential representation for the inherited classes in the implementation of the Visual Code Flow tool.

6.6 Visual Code Flow Components

The Visual Code Flow tool has three panels—a code panel, a memory panel and an output panel (Figure 13). It uses animation and variable transition from the code panel to/from the memory panel and the output panel that shows how the code execution affects the variables, classes and objects. It shows how the variables' values, which are defined inside methods or classes, have been set or received (Appendix C).

The main interface is divided into five key areas, as follows.

- The code area: where the code is presented and expression evaluation occurs
- The visualisation area: a memory area simulator where the tracing results are presented in addition to the class hierarchy (if it exists), classes, objects, variables and methods stored
- The control execution area, which has two options—either to monitor the execution in animation mode or use the control buttons.
- The output area: where the program displays the output(s)

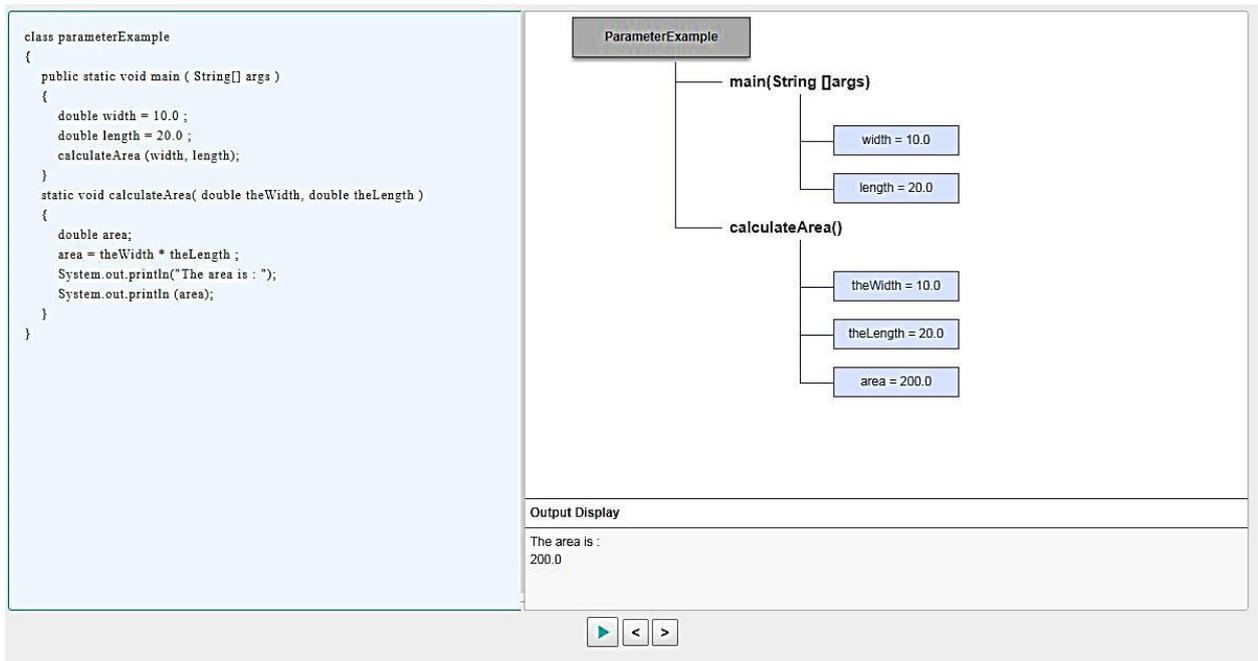
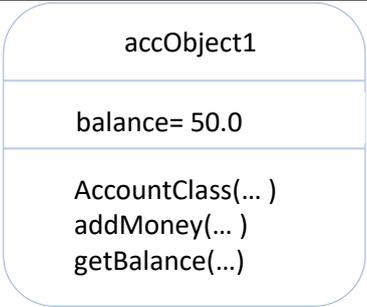
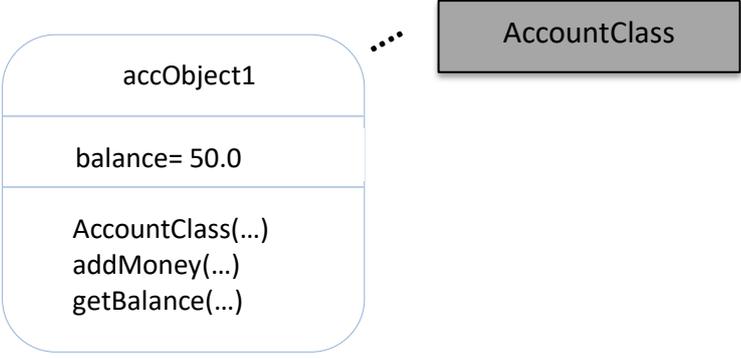
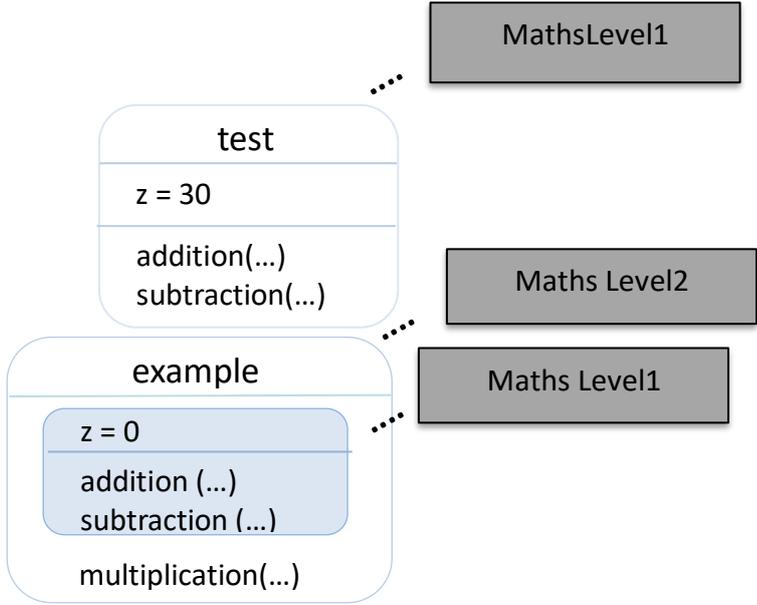


Figure 13. Visual Code Flow tool interface

Table 4 shows the Visual Code Flow tool and its components.

Table 4. Visual Code Flow tool components

Components	Description	Figures\examples
Control buttons	Control buttons consist of: <ul style="list-style-type: none"> • Play button, press to pause after playing • Backward and forward buttons 	
Methods	Text without box	calculateArea()
Variables	Light blue box	<div style="border: 1px solid black; background-color: #e6f2ff; padding: 2px; width: fit-content; margin-bottom: 5px;">width=10.0</div> <div style="border: 1px solid black; background-color: #e6f2ff; padding: 2px; width: fit-content;">length=20.0</div>
Classes	Dark grey box	<div style="border: 1px solid black; background-color: #a6a6a6; padding: 5px; width: fit-content; margin: 0 auto;">AccountClass</div>

<p>Objects</p>	<p>An ellipse with three parts: object name, local variables and local methods</p>	
<p>Class and object relationship</p>	<p>Refer the object to its class using a dotted line</p>	
<p>Inheritance</p>	<p>The example shows two classes of MathsLevel1 and MathsLevel2. Object 'example' inherits the attribute from class MathLevel1 and has its attribute from class MathsLevel2.</p>	
<p>Expression evaluation</p>	<p>In the example given, for the statement <i>area=theWidth*theLength</i>, it shows the value of each variable</p>	

		<pre> class parameterExample { public static void main (String[] args) { double width = 10.0 ; double length = 20.0 ; calculateArea (width , length); } static void calculateArea (double theWidth, double theLength) { double area; 10.0 20.0 area = theWidth * theLength ; System.out.println ("The area is : "); System.out.println (area); } } </pre>
--	--	--

Visual Code Flow tool features:

1- Control of the code's execution

The control of the execution will follow two modes:

- Animation
 - Where the user starts the animation and can pause it at any time and watch the sequence of the code execution. In the first row in Table 4, one can see the play button that can be used to start the animation. The button will change to pause when one starts the play in order to pause the animation at any time.
- Control button
 - Where the user can go through the code line by line by clicking the forward button or going back line by line using the backward button. In the first row in Table 4, one can see the backward and forward buttons that can be used to start the line-by-line control.

2- Interface/usability of the tool

Table 4 shows how the author used shapes to distinguish the shape of classes and objects. A normal grey box was used to represent the classes, and an ellipse was to represent the objects with three parts—object name, local variables and local methods. Dotted lines were used to link each object to

its class, as can be seen in the sixth row in Table 4. Text without any box represents the method, as can be seen in the second row in Table 4.

The Visual Code Flow tool also uses different colours to distinguish the different parts of the program. For instance, the light blue box represents the variables, and the dark grey box represents the classes. The inherited attributes in any object were embedded inside the inherited objects, as can be seen in the seventh row in Table 4.

3- Expression evaluation

The last row in Table 4 shows an example of how the author did the expression evaluation. When there is any arithmetic or logic operation, the values of each variable is written in the variable name in the expression to show its value before and after the execution of the statement.

4- Representation of class hierarchy

Representation of the class hierarchy was done sequentially. As noted earlier, this was the most preferred representation among the participants. As can be seen in Table 4 in the seventh row, the inherited classes appear from top to bottom in sequential order. The tool represents each class with its attributes (variable and methods) and links each class with dotted lines to illustrate the relationship between the other classes. For every subclass, a dotted line was used to refer to which superclass the subclass belongs. A light blue ellipse inside each subclass shows what variables and methods have been inherited from the superclass.

5- Programming problems

The Visual Code Flow tool presents three programming problems—calling methods, classes and objects and class inheritance. The three problems were selected because they are considered threshold concepts that students usually get stuck on in computing (Boustedt et al., 2007; Sanders et al., 2008; Bühlmann, 2011; Sanders et al., 2012; Sanders and McCartney, 2016; Kallia and Sentance, 2017).

The demonstration of the three problems at the end of the code execution is shown in Figures 14, 15 and 16, respectively.

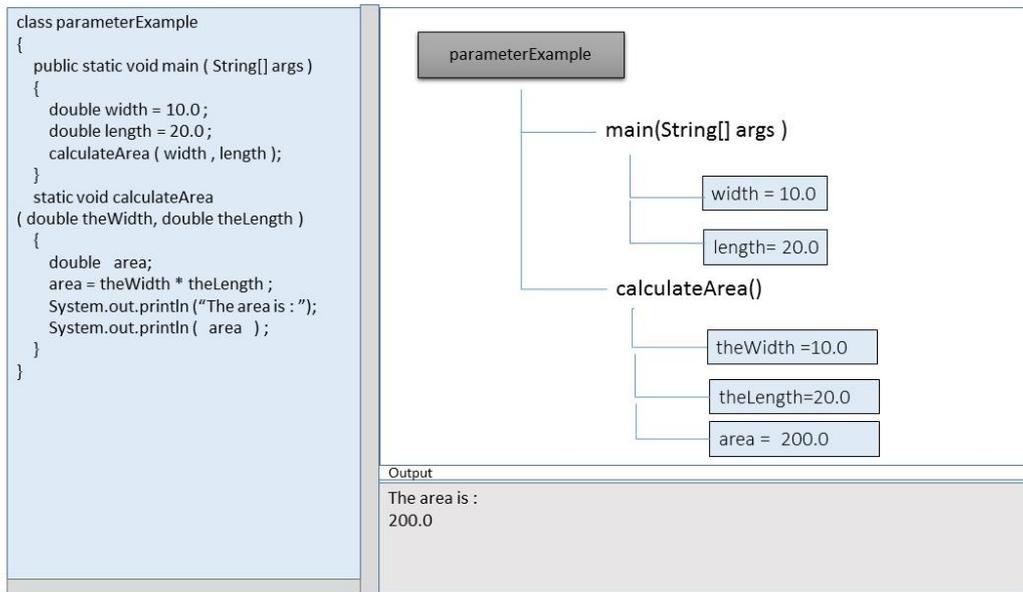


Figure 14. Visual code flow interface for the programming problem of calling method and passing parameters

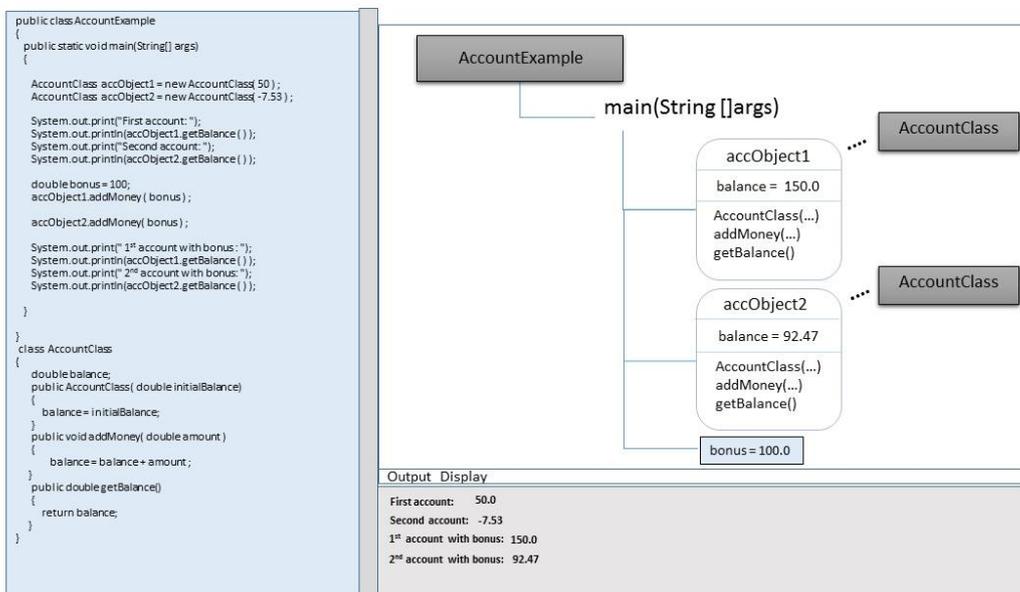


Figure 15. Visual code flow interface for the programming problem of defining and using classes and objects

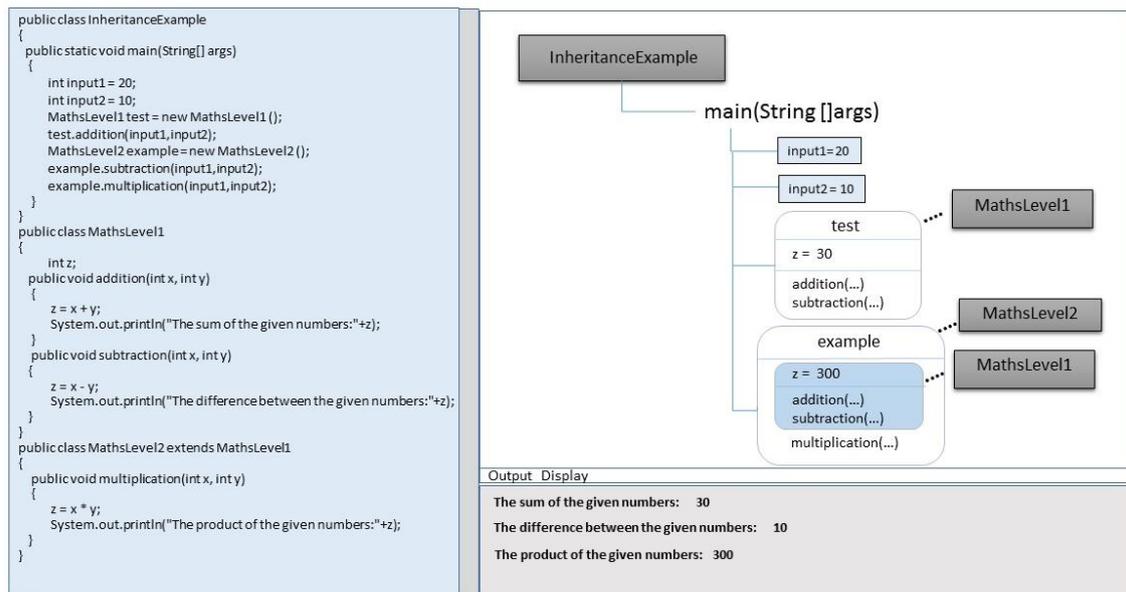


Figure 16. Visual code flow interface for the programming problem of class inheritance

6.7 Conclusion

This chapter presented a collection of data for the user requirements needed to develop a new framework. The study intended to show how visualisation tools and models are used to support the study of computer programming. The purpose was to establish whether visualisation could be exploited more fully or more effectively to support this learning. If that were the case, then this study would aim to define the requirements of novice programmers.

The results presented in this chapter show the categories of the visualisation tools and the students' preferences in using them. The students evaluated the following eight characteristics: tool availability, error explanation, expression evaluation, the interface, programming languages the tool support, animation, class hierarchy and saving the execution history.

The design component of the Visual Code Flow tool depended largely on the features found in the research study in this chapter regarding the role of visualisation in the study of computer programming and on the literature on the tools supporting programming education. This chapter presented how the Visual Code Flow was developed based on the research findings.

The categories of the Visual Code Flow tool that was developed and discussed in this chapter are:

1. Intuitive control of the execution of the program code. The tool enables users to play and pause the execution whenever they want, or they can add more control by navigating through the lines of the code line by line.
2. Availability of the tool (online or offline).
3. Because the tool is dedicated to novice programmers, it has a clear interface that allows ease of use.
4. The existence of expression evaluation, which means clearly showing the value of variables in any arithmetic or logic statement.
5. The tool provides a clear representation of the variables and methods.
6. The tool provides a clear, intuitive representation of the class hierarchy and uses different shapes and colours to represent classes and objects. It uses lines to show the relationships between classes and objects and represents class inheritance by referring each subclass to its superclass.
7. Highlighting the line currently being executed makes it easier to connect the code with the visualised representation.
8. Animation and transition for every component in the program from/to the code panel to the memory or output panel showing the change happening for each component if change exists.
9. Visual Code Flow simulates tracing in the manual method when using paper and pen.

The TLE evaluation that will be discussed in the next chapter used the Visual Code Flow tool as an instrument to measure the students' performance and confidence and to collect experts' perspectives through experiments that involved surveys, focus groups and interviews.

Chapter 7 EVALUATION OF THE USE OF VISUALISATION IN PROGRAMMING LEARNING

7.1 Introduction

The chapter presents an empirical study evaluating the application of visualisation when learning to program. Visualisation to trace the execution of a given program was used with novice programming students in a mixed-methods study. Pre- and post-test surveys, focus groups and interviews were used to gather information on student performance and confidence from both participants and a control group. The control group did not attend a visualisation session but only a standard teaching session.

The students' confidence in solving programming problems was a major aspect of the study. The study compared the students' confidence in the pre-test and the post-test, combined with data from focus groups with the students and interviews with programming teachers.

To evaluate the TLE framework, a visualisation model was developed to test its usability among novice programmers and instructors. The framework was evaluated using experiment and expert-based evaluation. The evaluation investigated the students' performance by measuring their scores.

There are some factors that could serve as implications for the study conducted in Chapter 6. First, the data collection period was limited, and because the research method involved semi-structured interviews the researcher had a limited period of time to gather students' opinions. To perform a more in-depth evaluation, the students would need to practice using the tools and become familiar with them. Second, the data collected from the study was qualitative data that does not measure the effect of visualisation on the students' performance, comprehension and confidence. An evaluation that measures the students' scores is needed to understand the impact of visualisation on novice programms.

7.2 Objectives

The purpose of this study was to evaluate the use of visualisation in learning to program from the perspectives of students and experts. The study aimed to discover the impact of using visualisation by tracing the program's instructions on the students' performance. Further, the study measured the students' level of confidence when solving programming problems before and after the visualisation. One of the factors investigated in the study was whether visualisation improves students' learning of programming concepts. The purpose was to evaluate whether visualisation plays a role in students' understanding of general programming concepts. Further, the study measured students' satisfaction with the visualisation tool. Finally, experts' opinions about the effectiveness of introducing visualisation in programming education were collected.

7.3 Evaluation Method

The evaluation phase sought to combine quantitative and qualitative approaches. The balance between quantitative and qualitative research helps fill the gaps between the two approaches. Relying on qualitative data may not provide a complete picture for evaluating an educational tool. For this evaluation, the quantitative data provided information about the students' performance and use of the tool, while the qualitative data gathered provided information on the opinions of the students and the perspectives of experts.

Therefore, a combination of surveys, focus groups with students and interviews with experts may draw additional findings that contribute to the research. Figure 17 shows the experiment's components and the flow of procedures that each group of participants went through.

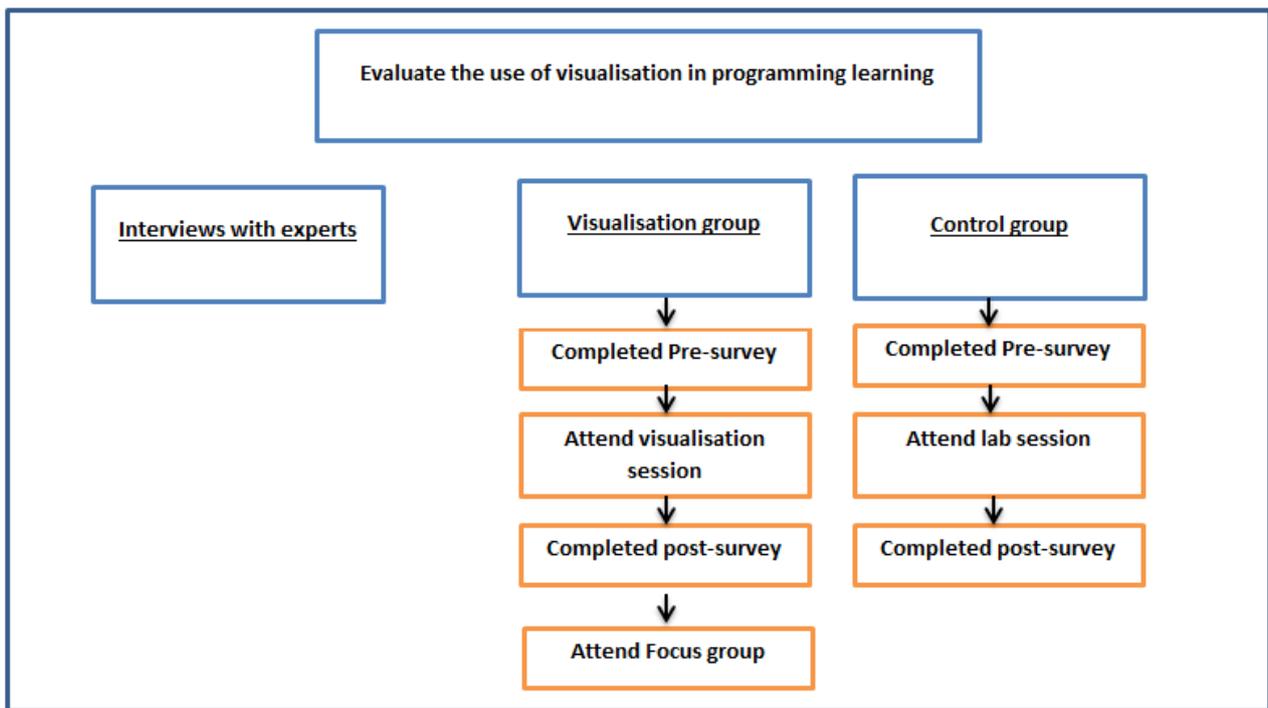


Figure 17. Experiment components

The experiment divided the participants into two groups; one group took part in the visualisation session and the other did not (control group). Therefore, the control group only experienced the lab activity and the visualisation group experienced a visualisation session. All participants took part in pre- and post- activity surveys. Each survey measured the participants' scores and confidence level in answering each question.

Conducting a class experiment that compares the performance of two groups (control and experimental) has been used in previous studies to evaluate visualisation in programming education. For instance, Mselle (1989) tested hypotheses concerning using memory diagrams in programming education to enhance students' ability to write code by conducting a class experiment involving 100 students divided into two groups (control and visualisation groups). The researcher then compared the test scores of the two groups in the final exam.

A questionnaire is a common approach used to collect quantitative data in evaluating the visualisation method. Researchers such as Dixon (2004b), Virtanen et al. (2005), Kasurinen et al. (2008) and Sun (2010) evaluated the use of their visualisation tools by administering questionnaires to determine their tools' effectiveness.

The evaluation in the present study also included interviews, which have been used effectively in the related research to collect qualitative data. Dixon (2004a) tested a visualisation tool by collecting feedback through interviews with the author/developer and lecturers.

7.3.1 Selection of population

The number of students in each group differed slightly, as it depended on the course classes and the laboratory (see Table 5).

Table 5. Number of students who participated in the experiment

	Visualisation group		Control group	
	Pre-test session	Post-test session	Pre-test session	Post-test session
Calling method	31	31	29	29
Classes and objects	30	30	24	24
Class inheritance	26	26	26	26

In the experiment and the focus group, the participants were students enrolled in Programming Language I and II courses. The programming curriculum included computer concepts and programming in Java. The Programming Language I course, a prerequisite for Programming Language II, consisted of classes, objects and methods and was taken in the first year in the first semester. The Programming Language II course included class inheritance and was taken in the second semester of the first year. As the target of the study was novice programmers, the experiments were conducted after participants took the courses to guarantee no other factors would affect the results. Table 6 shows the students' experience in terms of modules taken by students in each group.

Table 6. The experience in terms of modules taken by the students in each group and section

	Groups/Sections		
	Case A students	Case B students	Case C students
	Call method	Classes and objects	Class inheritance
Modules	Variables, data types, I/O operations, arithmetic operations, using constants and comments, debugging, Boolean expressions, relational operations, conditional statements (if, switch), loop (while, do while, for loop), break and continue, methods		
	Arrays, introduction to classes, constructors, Class inheritance		

The expert interviews were conducted with six lecturers and tutors, as shown in Table 7.

Table 7. Number of experts who participated in the experiment

Personal information	Exp1	Exp2	Exp3	Exp4	Exp5	Exp6
Gender	F	F	F	F	M	M
Degree	Master	PhD				
Occupation	Lecturer	Assistant professor	Associate professor	Assistant professor	Assistant professor	Associate professor

The lecturers and tutors who participated in the interviews were selected based on their experience in programming education. They had a minimum of seven years of teaching experience. The teaching experience was mainly in programming for beginners and at the advanced level for programming languages such as C, C++, C # and Java and web programming.

7.3.2 Instrument

The Visual Code Flow tool was used to evaluate the use of visualisation in programming education. It was developed based on the characteristics of the user requirements collected from semi-structured interviews in a qualitative study (Alhammad et al., 2016) (refer to Section 6.5). The researcher presented the Visual Code Flow to the visualisation group after the pre-test. The students were then given time to use the tool by themselves during the session.

7.3.3 Programming subjects

The programming problems embedded in the study were based on identified threshold concepts that students commonly find difficult, as discussed earlier in Sections 3.6 and 6.2.3. These were calling a method, classes and objects and class inheritance.

7.3.4 Quantitative method

Surveys were used to collect quantitative data in the experiment. Surveys are considered an effective tool to gather and analyse data from a large number of participants. As three programming topics were being covered, the survey questions were different depending on the topic. Therefore, for each problem, two types of surveys were distributed for each group: a pre-survey and a post-survey (see Appendix E). The surveys consisted mainly of three sections:

1. **Section one:** A questionnaire that measured the participants' confidence in their understanding of programming concepts. The first section in the questionnaire was provided for both surveys to compare the data before and after the experiment.

The questions in the first section for both groups and both surveys (pre- and post-survey) are basically the same but with a slight difference depending on the programming topic. The following questions appeared in the first section of the questionnaire:

Common questions that appeared in the survey for the three topics:

- I can program competently in at least one text-based programming language.
- I can use variables.
- I can use the relational operators.
- I can use the arithmetic operators.
- I can debug syntax errors.
- I can debug logical errors.

Questions that appeared in the survey for the call procedure topic

- I can use the procedures.
- I know how to call procedures
- I know how to pass a parameter(s).
- I know how procedures return value(s) to the program.

Questions that appeared in the survey for classes and objects topic

- I can use classes and objects.
- I know how to use constructors.

Questions that appeared in the survey for class inheritance topic

- I can use classes and objects.
- I can use the class inheritance.

2. **Section two:** This section consisted of two parts. The first part included a test programming problem that asked participants to trace the program and then answer some questions regarding the given program, such as to identify the values of variables or objects in some of the execution lines. The second part consisted of a questionnaire that measured the participants' confidence in solving each question for the given program.

Questions that appeared in the survey for the call procedure topic

- What is the purpose of the program?
- What is the output of the program?
- What does the statement in line 7 do?
- What is the value of the variable width after executing line 7?
- What is the value of the variable length after executing line 7?
- What is the value of the variable theWidth after executing line 9?
- What is the value of the variable theLength after executing line 9?
- What is the value of the variable area after executing line 12?
- What is displayed after executing line 13?
- What is displayed after executing line 14?
- Which line(s) of the code (if any) did you struggle with understanding?

Questions that appeared in the survey for classes and objects topic

- What is the purpose of the program?
- What is the output of the program?
- What is the value of variable balance in account1 after executing line 5?
- What is the value of variable balance in account2 after executing line 6?
- What is displayed after line 8 is executed?
- What is displayed after line 10 is executed?
- What is the value of variable balance in account1 after executing line 12?
- What is the value of variable balance in account2 after executing line 13?

- What is displayed after line 15 is executed?
- What is displayed after line 17 is executed?
- Which line(s) of the code (if any) did you struggle with understanding?

Questions that appeared in the survey for the class inheritance topic

- What is the purpose of the program?
- What is the output of the program?
- What does the statement in line 28 do?
- What are the attributes of object test after executing line 7?
- What is the value of variable z after executing line 8?
- What is displayed after line 20 is executed?
- What are the attributes of object example after executing line 9?
- What is the value of variable z after executing line 10?
- What is displayed after line 25 is executed?
- What is the value of variable z after executing line 11?
- What is displayed after line 33 is executed?
- Which line(s) of the code (if any) did you struggle with understanding?

3. **Section three:** The participants of the visualisation session were also asked to answer a third part in the survey, a questionnaire to measure the level of satisfaction in using the visualisation tool. This section was only included in the post-survey.

Questions that appeared in the survey for the call procedure topic

- Using the tool helped me understand the procedure call.
- I would prefer to use the tool to trace the code rather than manual approach in Section 2.
- I would prefer to use the manual approach in Section 2 to trace the code rather than the tool.
- The tool was useful for learning Java.
- Using the tool helped me understand the changes in the variable values.

- The tool is easy to use.
- I would use a tool like this one.

Questions that appeared in the survey for classes and objects topic

- Using the tool helped me understand classes.
- Using the tool helped me understand objects.
- Using the tool helped me understand the difference between classes and objects.
- Using the tool helped me understand the class constructor.
- Using the tool helped me understand how to set the variable value defined in the class.
- Using the tool helped me understand how to get the variable value defined in the class.
- I would prefer to use the tool to trace the code rather than manual approach in Section 2.
- I would prefer to use the manual approach in Section 2 to trace the code rather than the tool.
- The tool was useful for learning Java.
- Using the tool helped me understand the changes in the variable values.
- The tool is easy to use.
- I would use a tool like this one.

Questions that appeared in the survey for the class inheritance topic

- Using the tool helped me understand class inheritance.
- Using the tool helped me understand how to set the value of the variable that is inherited from the superclass.
- Using the tool helped me understand how to get the value of the variable that is inherited from the superclass.
- I would prefer to use the tool to trace the code rather than the manual approach in Section 2.
- I would prefer to use the manual approach in Section 2 to trace the code rather than the tool.
- The tool was useful for learning Java.

- The tool is easy to use.
- I would use a tool like this one.

7.3.5 Experiment preparation

As mentioned in Section 7.3.1, the participants were students at an academic institution, and the experiments were conducted on campus. The researcher made an agreement with the lecturer and tutor who taught the students for either spare time or class time or lab time to conduct the experiment. This agreement was reached either via email or in person. A total of six different sections participated in the experiment (Table 8) to arrange the students in the experiment groups. A section means a group of students who attended the lecture for a specific course at the same time and were taught by the same lecturer.

Table 8. Arrangement of the students in the experiment

Problem Case	Control Group	Visualisation Group
Calling method	Section 1	Section 2
Class and object	Section 3	Section 4
Class inheritance	Section 5	Section 6

For each section, the students spent approximately 40–50 minutes completing the pre-survey in the classroom and 50–60 minutes completing the post-survey in the computer lab. The researcher managed the date and time of the experiment, so the pre-survey came after the students were taught the lesson during the lecture time. The researcher managed the post-survey that came after the lab session for the control group and before the lab session for the visualisation group. Therefore, the researcher guaranteed that they compared the effect of the lab activity on the students' performance against the effect of visualisation.

7.3.6 Experiment procedures

Both groups followed the same procedure in completing the pre-survey. The following steps describe the procedure for the pre-survey:

- 1) The researcher asked the students if they were willing to participate in the experiment. If they were willing, then the researcher explained about withdrawal—that they could withdraw at any time from the experiment but not the learning course and could subsequently withdraw their data from the survey and the study and how to do so. The researcher explained that all personally identifiable information would be held separate from the core dataset.
- 2) Once the students were briefed and if they were willing to participate, the researcher asked them to sign the consent document before proceeding. An information sheet was also provided.
- 3) The researcher provided the students with an overview of the session activities and duration. The overview gave a breakdown of how long each step should take. This ensured that the students were fully informed of the process.
- 4) The students were given an introduction to the subject and the objectives of the study. The introduction provided the students with the background they needed to do the tasks.
- 5) The researcher explained the survey questions and how the students should fill out the answers.
- 6) After that, the students attempted to answer the survey. The students could interrupt the researcher to ask for clarification of any point.
- 7) Last, the papers were collected by the researcher.

Both groups followed the same procedure in completing the post-survey, except that for the visualisation group the step before answering the post-survey consisted of presenting the visualisation method using the Visual Code Flow tool and asking them to try it on their computers during the session. The visualisation demonstration lasted 20–30 minutes depending on the programming problem. The control group relied on the lecture notes and lab work to continue their part of the experiment.

7.3.7 Focus groups

After the experiment, the visualisation group was invited to a focus group to discuss their experience. They were asked to express their feedback and the aspects they liked or disliked in using the visualisation.

The purpose of conducting focus groups in this evaluation was to provide an atmosphere for the students where they would feel comfortable relating their ideas, concerns and experience. In the focus group discussion, students could influence each other and could share their opinions and thoughts, unlike with surveys that do not give participants the opportunity to develop new ideas. Researchers in the educational field, such as Williams and Katz (2001), have pointed out the advantages of conducting focus groups. According to their study, focus groups help in evaluating and developing learning tools. Conducting focus groups in the present study contributed to the evaluation of visualisation and to understanding the students' needs.

1. Focus group preparation

To gather the participants, the moderator arranged a timetable based on the free time (break time) available in the students' schedule. Students were distributed by their reference number across the focus group timetable. The timetable was organised according to each programming problem (Cases A, B and C) and the group number, as each case contains more than one group. For each problem, there were five or four groups. The visualisation group was distributed among these small groups of six to seven students to make the discussion easier to control (Table 9).

Table 9. Number of groups/participants in the focus group

Problem labels	Number of participants in each group				
A: Calling method Total of 31 participants	1 st group=7	2 nd group=6	3 rd group=6	4 th group=6	5 th group=6
B: Classes and objects Total of 30 participants	1 st group=6	2 nd group=6	3 rd group=6	4 th group=6	5 th group=6
C: Class inheritance Total of 26 participants	1 st group=7	2 nd group=7	3 rd group=6	4 th group=6	

As Table 9 shows, each problem was assigned a code, A, B or C, for the three problems and a group number from one to five. For instance, the participants in the Case A problem who belonged to the third group were assigned a reference (A.3.P2). The researcher emailed an invitation to the students who participated in the focus groups that contained a timetable specifying when and where the focus group would be conducted.

2. Focus group procedures

As discussed in Section 7.3.7, the visualisation group discussed their experience and provided their feedback and suggestions in the focus group sessions. The researcher herself moderated the focus groups, asked the questions and monitored and guided the discussion (see Appendix E for the focus groups questions). Each session lasted 45–50 minutes. The researcher gathered the students in a quiet and convenient meeting room that had a circular table so the students felt comfortable during the session. The researcher first introduced herself and presented the topics and the objectives. She then distributed a short questionnaire to determine the students' learning tool experience to prepare them for the in-depth discussion. As the students were already involved in the visualisation session and completed the pre- and post-surveys, the following step involved asking the students about their experience and feedback. The following were the suggested questions for the moderator to manage the discussion. The questions evolved based on the participants' answers.

Section one: Short questions

- Which level or course of programming do you study now?
- Which programming languages do you learn?
- What materials or websites do you use in addition to the lecture notes in studying programming?
- Have you ever used a visualisation tool to improve your understanding of programming?
(interviewer should explain the nature of visualisation tool)
- If so, what tool?
- How often do you use it?

- What aspects of the tool do you like/dislike?

Section two: Open-ended questions

Note that the discussion evolved depending on the topic and on the students' answers.

- What are the strength(s) and weakness(es) of the tool in general?
- How did you find the method representation? What things did you like and dislike?
- How did you find the variable representation? What things did you like and dislike?
- How did you find the expression of evaluation? What things did you like and dislike?
- How did you find the passing parameters and calling method representation? What things did you like and dislike?
- How did you find the class representation? What things did you like and dislike?
- How did you find the object representation? What things did you like and dislike?
- How did you find the class inheritance representation? What things did you like and dislike?
- How did you find the output representation? What things did you like and dislike?
- How did you find the animation? What things did you like and dislike?
- How did you find the control of execution? What things did you like and dislike?
- Would you use the tool if possible? Explain why or why not.
- Do you think it is beneficial and may improve programming learning?
- What improvements should be implemented in refining this tool?

Finally, the researcher concluded the focus group by asking the students to add any suggestions or voice any concerns they still had. All sessions of the focus group were audio recorded, and the researcher transcribed the discussion after the sessions. To understand the participants' comments, the transcripts were shortened to include only essential information. The transcripts were then cleaned and labelled with the group and participant numbers, so each comment appeared in a separate row labelled by the group and participant number. The answers for each question collected from the different groups were grouped and assigned codes and a participant ID. After all the

responses were entered, they were categorised according to some keywords and topics. The constant comparative analysis method was used to analyse the data, as recommended in the case of a focus group that consists of multiple groups in the same study (Kolb, 2012). See Appendix F, Section F-3 for the focus group transcript to see how the participants answered the discussion questions.

7.3.8 Interviews

Further evaluation of the visualisation was carried out through semi-structured interviews with experts in the field (lecturers and tutors). Semi-structured interviews were used in order not to limit respondents but allow an open conversation to gather the maximum amount of data.

1. Interview preparation

Before an interview took place, experts were invited via email or personally to participate in the interview. The participants had the right to choose the date and time that was suitable for them. Eventually, the researcher and the participants agreed on the date, time and place for the interviews. Participants were informed about ethical principles, such as confidentiality, in the invitation email. Interviews were conducted in a quiet office to ensure the participants' comfort.

2. Interview procedures

Six participants were interviewed separately, with each interview lasting 50–60 minutes.

- First, the researcher introduced the topic and the research objectives (see Appendix E for the experts' interview questions). The researcher asked the participants about their experience and background in programming education.

Section1: Background information

- Do you teach a programming language(s)?
If yes, proceed to the following questions.
- How long have you been teaching programming?
- Which level or course of programming do you teach?
- Which programming language do you teach?

- What is your experience (if any) using visualisation tools to improve your teaching of programming courses?(Interviewer should explain the nature of the visualisation tool.)

If yes, the following questions will be asked.

- What is the tool? Describe how it works and its method.
- How often do you use it?
- What are the tools' features?
- What aspects of the tool do you like/dislike?
- To what extent was it helpful?

Table 10 summarises the answers to each question regarding the interviewees' experience teaching programming.

Table 10. Expert interview background questions

Question	Answers					
	Expert 1	Expert 2	Expert 3	Expert 4	Expert5	Expert 6
Teaching experience	7 years	10 years	15 years	13 years	9 years	17 years
Course level	Programming I and II					
Programming languages	Java	Java,C,C++	Java, .NET	Java,C,C++, Pascal	Java, Pascal	Java,C,C++, Pascal, Python
Using visualisation tools	None	Jeliot	Jeliot	BlueJ	None	Python Tutor

- The researcher presented the visualisation method using the tool (Visual Java Code) to the participants and then asked for their feedback, comments and suggestions about using visualisation in studying programming. The questions were as follows.

Section 2: Evaluation questions for the Visual Java Code tool

- What are the strengthen(s) and weakness(es) of the tool?

- Will you use it if possible? Explain why or why not.
 - Do you think it is beneficial and may improve programming learning?
 - What improvements should be implemented in refining this tool?
 - How did you find the method representation? What things did you like and dislike?
 - How did you find the variable representation? What things did you like and dislike?
 - How did you find the expression of evaluation? What things did you like and dislike?
 - How did you find the passing parameters and calling method representation? What things did you like and dislike?
 - How did you find the class representation? What things did you like and dislike?
 - How did you find the object representation? What things did you like and dislike?
 - How did you find the class inheritance representation? What things did you like and dislike?
 - How did you find the output representation? What things did you like and dislike?
 - How did you find the animation? What things did you like and dislike?
 - How did you find the control of execution? What things did you like and dislike?
- Finally, the researcher asked the experts for any further comments or suggestions. See Appendix F, Section F-2 for the interview transcript to see how the participants answered the interview questions.

The researcher audio recorded the interviews and transcribed the data afterwards. The collected responses were categorised by participant code, and the answers were listed under each question. The answers for each expert were grouped under each question and assigned the participant reference for each answer (row). The cycle pattern coding method was used to analyse the data (Saldaña, 2016). The data analysis is discussed in Section 5.4.4.

7.4 Findings

7.4.1 Survey results

The quantitative analysis measured three factors for both groups in the pre- and post-tests, the students' scores on the test, their level of confidence while completing the test and their confidence about programming topics in general. Students' satisfaction was measured for the visualisation group only. After the results were analysed, the improvement achieved by both groups was measured.

The scores of both groups showed improvement in students' performance after attending either a visualisation session or a lab session. However, the visualisation group achieved better performance. The increment in the average scores for the visualisation group was more than that of the control group. Table 11 summarises the mean (M), standard deviation (SD) and skewness (SK) of the visualisation group's test scores.

Table 11. Statistical results for the visualisation group

	Pre-test			Post-test		
	M	SD	Sk	M	SD	Sk
Calling method	6.5	2.1	-0.5	7.6	3.1	-1.0
Classes and objects	4.1	2.8	0.2	5.3	3.0	-0
Class inheritance	4.7	2.4	-0.0	6.8	3.2	-0.2

The increase in the three means shows the visualisation method increased students' comprehension of the programming topics, as individuals had like responses to every question. Class inheritance had a high rate of increase in the average score of students, which means the visualisation of class inheritance had the most effect on understanding.

For the control group, the mean in every problem increased when lab sessions were carried out regularly, indicating those students' responses became more similar. However, the mean for the calling method did not show a significant difference between the two tests. Table 12 summarises the M, SD and SK of the control group's test scores.

Table 12. Statistical results for the control group

	Pre-test			Post-test		
	M	SD	SK	M	SD	SK
Calling method	7.0	2.6	-0.6	7.1	2.9	-0.5
Classes and objects	4.1	3.6	0.5	4.9	2.9	0.4
Class inheritance	5.2	3.1	0	6.0	3	-0.2

Table 13 summarises the comparison between the visualisation and control groups.

Table 13. Comparison between the visualisation and control groups

	Visualisation group			Control group		
	M	SD	SK	M	SD	SK
Calling method	7.6	3.1	-1	7.1	2.9	-0.5
Classes and objects	5.3	3.0	-0	4.9	2.9	0.4
Class inheritance	6.8	3.2	-0.2	6.1	3	-0.2

The mean of the three problems best compares the visualisation group and the control group. The factors calculated for the visualisation group for every issue are higher than those for the control group. The statistical differences reveal the visualisation group was more effective in improving their performance than the control group.

To confirm the findings regarding the students' scores, the researcher analysed whether the students' confidence improved while answering the test questions. The findings showed that visualisation plays a positive role in gaining confidence while answering programming questions. To better understand the transformation of the students' confidence while solving the programming problems, the researcher categorised the findings based on the three problems (Cases A, B and C). In all of Cases A groups, the participants reported their level of confidence (not confident, confident or very confident) for each question they solved. Only the level of confidence for the correct answers was counted.

Case A: calling a method

The questions given to the participants were classified into main topics. Table 14 shows the topics and gives examples of the kinds of questions.

Table 14. Case A - question topics

Question topics	Example
The output of the calling method problem	What is the output of the program?
Call a method	What does the statement <i>calculateArea (width, length)</i> do?
Value(s) of the passing parameter(s)	What is the value of the variable <i>width</i> after calling the method <i>calculateArea (width, length)?</i>
Value(s) of the variable(s) in the method header	What is the value of the variable <i>theWidth</i> after the method calling <i>static void calculateArea(double theWidth, double theLength)?</i>
Arithmetic expression	What is the value of variable <i>area</i> after the execution of sentence <i>area = theWidth * theLength ;?</i>

Table 15. Case A – students' level of confidence in solving the problem

The topics in Case A	Control group						Visualisation group					
	Pre-test			post-test			Pre-test			post-test		
	*x	Y	z	X	Y	Z	x	y	z	x	Y	z
The output of the calling method problem	11%	25.9%	62.9%	10.7%	25%	64.2%	4.17%	50%	45.8%	0	56%	44%
Call a method	21.4%	42.8%	35.7%	21.4%	35.7%	42.8%	30%	40%	30%	0	31.2%	68.7%
Value(s) of the passing parameter(s)	13.4%	30.7%	55.7%	9.26%	37%	53.7%	16.3%	59.1%	24.4%	17.6%	23.5%	58.8%
Value(s) of the variable(s) in the method header	20%	36%	44%	20%	33.3%	46.6%	56.2%	28.1%	15.6%	17.6%	23.5%	58.8%
Arithmetic operation	25.9%	25.9%	48.1%	15.3%	30.7%	53.8%	25%	50%	25%	13%	65.2%	21.7%

* x=not confident, y=confident, z=very confident

The confidence level of the participants for both groups was evaluated based on the topics in Table 14. The results presented in Table 15 show the percentage of respondents who identified their level of confidence in both groups for both tests.

In the visualisation group, the students had a positive shift in their level of confidence in solving the calling method problem (Case A) after attending the visualisation session. Most students were not confident in solving the question related to the value of the variables received by the method header after calling the method in the pre-test. In the post-test, most students became very confident in solving the same question. More than half of the students (58.8%) became very confident in solving the question, an increase from 15.63% before the visualisation. In addition, the percentage of very confident students in the passing parameter question increased significantly from 24.49% to 58.8%. However, the students in the control group were still struggling after attending the lab session. Although the number of students who became confident increased, the number of non-confident students remained the same for the question related to the value of the variables received by the method header after calling the method. There was an improvement in confidence for the control group students for the rest of the questions, but it was not as remarkable as for the visualisation group.

Case B: classes and objects

The researcher classified the questions that were given to the participants about the classes and objects problem into topics. Table 16 shows the topics and gives examples of the kinds of questions.

Table 16. Case B – question topics

Question topics	Example
The output of the program	What is the output of the program?
Class constructor/object creation	What is the value of variable <i>balance</i> in <i>account1</i> object after executing the following statement? <i>Account account1 = new Account (50)</i>
Get value(s) from the object before the set method	What is the value of variable <i>balance</i> in <i>account1</i> after executing <i>account1.getBalance ()</i> ?

Set value(s) to object	What is the value of variable <i>balance</i> in <i>account1</i> after executing <i>account1.addMoney (bonus)</i> ?
Get value(s) from the object after the set method	What is the value of variable <i>balance</i> in <i>account1</i> after executing <i>account1.getBalance ()</i> ?

The confidence level of the participants in both groups was evaluated based on the topics in Table 16. The results presented in Table 17 show the percentage of respondents in both groups who identified their level of confidence for both tests.

Table 17. Case B – students’ level of confidence in solving the problems

The topics in Case B	Control group						Visualisation group					
	Pre-test			Post-test			Pre-test			Post-test		
	*x	Y	Z	X	Y	Z	X	Y	z	x	Y	z
The output of the program	25%	62.5%	12.5%	25%	62.5%	12.5%	33%	66.6%	0	9.09%	27.2%	63.6%
Class constructor/object creation	36%	45.4%	18%	33%	41.6%	25 %	71.4%	22.8%	5.7%	25 %	35.7%	39.2%
Get value(s) from object before set method	57%	42.8%	0	60%	40%	0	76.4%	23.5%	0	30%	40 %	30%
Set value(s) to object	20%	60 %	20 %	20 %	60%	20%	55.5%	44.4%	0	16.6%	33.3%	50%
Get value(s) from object after set method	60%	40 %	0	60 %	40 %	0	83.3%	16.6%	0	25%	43.7%	31.2%

* x=not confident, y=confident, z=very confident

In the pre-test survey given to the visualisation group for Case B, many students were not confident in answering the questions presented in the program. The only issue related to creating an object, where two students (5.7%) were very optimistic and very confident about solving the question. After the experiment using visualisation was administered to the students, most responded positively about solving the object creation question and using the class constructor. The percentage of non-confident students who solved the question decreased from 71.4% to 25%. The second topic evaluated was getting values from an attribute in an object by asking about the value return from the method (get). The percentage of non-confident students who solved the question decreased from

76.4% to 30%, and the number of very confident students rose by 30%. The third topic was setting values in an object's attribute. For example, the students were asked to recognise the value of the variable (*balance*) that was defined in two objects (*account1*) and (*account2*) before and after adding a bounce to the balance existing in the two objects. The students' confidence in answering the question improved. The results showed 50% of the students became very confident in answering the question correctly, while none were very confident in the pre-test. The fourth topic asked the students about the set method, but this time after executing the set method, which changed the value of the variables. The percentage of non-confident students who solved the question decreased from 83.3% to 25%, and the number of very confident students rose to 31.2%. Overall, the students showed improved confidence in finding the program output after visualisation.

The control group showed a slight improvement on one topic—the class constructor—after attending the lab session. The number of very confident students who solved the question of the class constructor increased from 18% to 25%. However, the rest of the questions did not register any improvement in the students' confidence. The results showed that the lab session did not affect the confidence of students in finding the program output.

Case C: class inheritance

The researcher classified the questions given to the students in the class inheritance problem into topics. Table 18 shows the topics and gives examples of the kinds of questions.

Table 18. Case C – question topics

Question topics	Example
The output of the program	What is the output of the program?
Identify the class inheritance	What does the following statement do? <i>public class MathsLevel2 extends MathsLevel1</i>
Create an object from the superclass	What are the attributes of object <i>test</i> after executing <i>MathsLevel1 test = new MathsLevel1 ()?</i>
Operate a superclass	What is the value of variable <i>z</i> after executing <i>test.addition(input1,input2)</i> , where <i>z</i> is a local variable in the object <i>test</i> ?
Create an object from a subclass	What are the attributes of object <i>example</i> after executing <i>MathsLevel2 example = new MathsLevel2 ()?</i>

Operate the inherited method	What is the value of variable z after executing <i>example?subtraction(input1,input2)</i> , where subtraction is an inherited method?
Operate the non-inherited method	What is the value of variable z after executing <i>example?multiplication(input1,input2)</i> , where multiplication is a non-inherited method?

The confidence level of the participants in both groups was evaluated based on the topics in Table 18. The results presented in Table 19 show the percentage of respondents who identified their level of confidence in both groups for both tests.

Table 19. Case C- students' level of confidence in solving the problems

The topics in Case C	Control group						Visualisation group					
	Pre-test			post-test			Pre-test			post-test		
	*x	Y	Z	x	Y	z	x	y	z	X	Y	z
The output of the program	5.5%	66%	27.7%	7.69%	53.8%	38%	11.5%	42%	46%	4.7%	28.5%	66%
Identify the class inheritance	5%	65%	30%	23.5%	29%	47%	22.7%	45%	31.8%	16%	24%	60%
Create object from the superclass	33%	66%	0	0	60%	40%	100%	0	0	23%	53.8%	23%
Perform operation in a superclass	20%	32%	48%	8.3%	25%	66.6%	36%	45%	18%	14.2%	28.57%	57%
Create object from a subclass	25%	25%	50%	0	0	100%	66.6%	0	33%	41.6%	16.6%	41.67%
Perform operation in inherited method	13%	46%	40%	21%	25%	53.5%	36%	52%	12%	11.5%	23%	65%
Perform operation in non-inherited method	10%	51%	37.9%	8%	36%	56%	45%	45%	10%	29.6%	25.9%	44.4%

* x=not confident, y=confident, z=very confident

For the class inheritance problem (Case C), students in the visualisation group had a significant level of confidence regarding the questions. The degree of confidence increased when the post-test was undertaken. In general, the students became more confident when creating class inheritance and dealing with the inherited components. The results showed an increase in the students' confidence when answering the program output question in the post-test. However, the problem that registered

the highest decrement for the non-confident students was creating an object from the superclass. The number of negative responses about solving the question decreased from 100% to 23%. The number of students who became very confident increased from 0% to 23%. The results also showed an improvement in answering the question about performing operations in the inherited methods. The number of students who responded negatively to solving the question decreased from 36% to 11.5%. The number of students who became very confident rose from 12% to 65% of the total students, while the improvement achieved for the question about performing the operation in the non-inherited method rose from 10% to 44.4%. The control group had the same improvement, but with a lower ratio for the questions about how to create an object from a subclass and a superclass and how to operate the superclass. However, the rest of the questions, those about identifying class inheritance or performing operations in inherited or non-inherited methods, registered decreased confidence. For instance, the percentage of non-confident students who solved the identifying class inheritance question increased from 5% to 23.5%, and the number of confident students decreased from 65% to 29.41%.

Finally, the third aspect analysed was the transformation in students' confidence regarding programming topics. The survey revealed whether students' confidence in understanding programming concepts in general changed after administering each process. The questions in this part were slightly different depending on the programming concept that was focused on. As there were three concepts that covered programming problems presented in the pre- and post-tests, the discussion was categorised based on the three cases, A, B and C, for calling a method, classes and objects and class inheritance problems, respectively.

There were common questions in the pre- and post-surveys for the three cases, as follows.

- I can program competently in at least one text-based programming language.
- I can use variables.
- I can use the relational operators.
- I can use the arithmetic operators.

- I can debug syntax errors.
- I can debug logical errors.

Case A: calling a method

More questions were added to Case A, as follows.

- I can use the methods.
- I know how to call methods.
- I know how to pass a parameter(s) to a method.
- I know how methods return value(s) to the program.

Figure 18 shows the changes in the students’ responses in the pre- and post-tests for both groups.

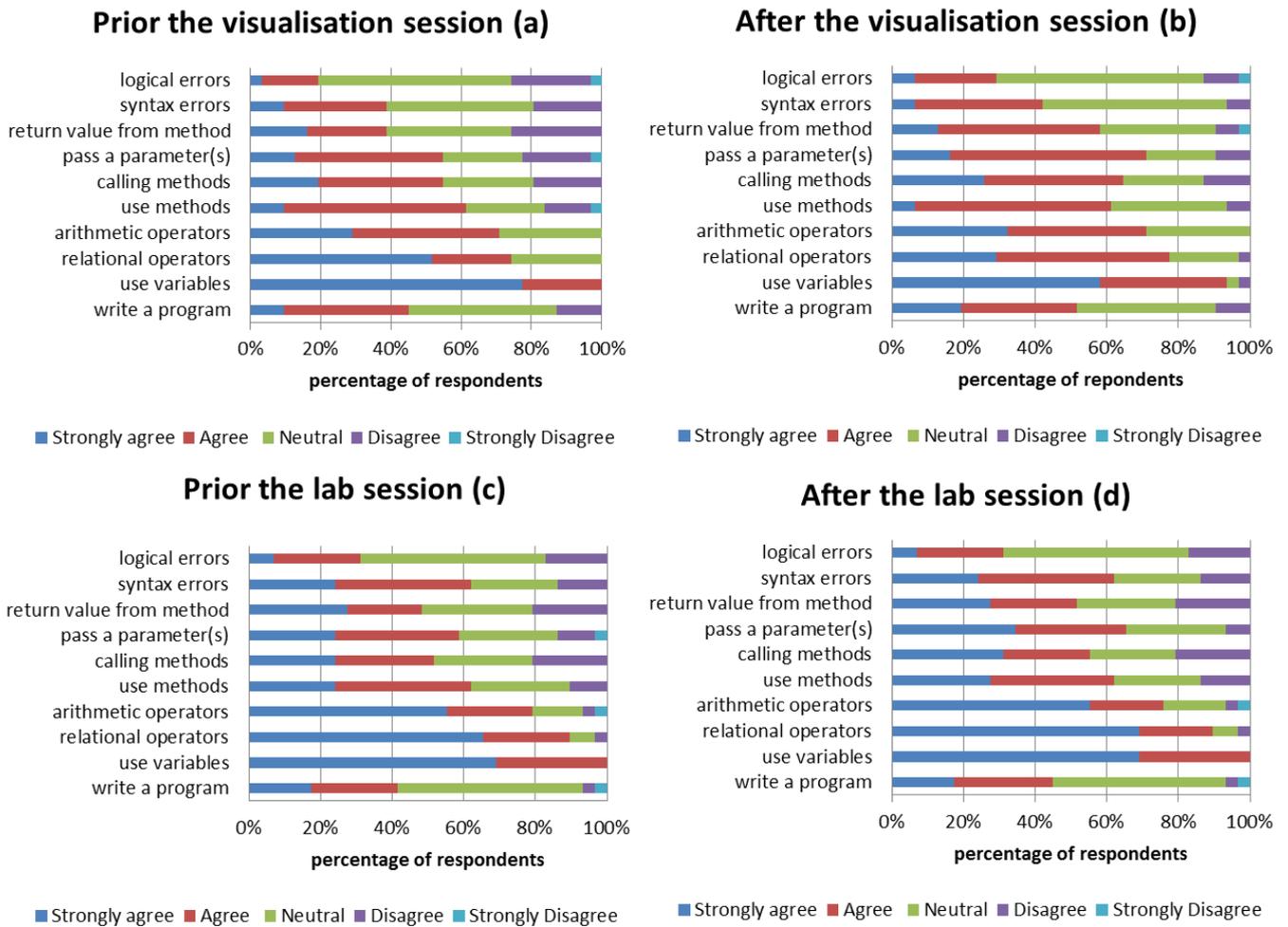


Figure 18. Case A - students’ confidence regarding general topics in programming

For the visualisation group that solved the calling method problem, in the first survey conducted before the experiment (Figure 18 (a)) the number of students who were confident using variables in a program was the highest, while the number of students confident in their ability to debug logical and syntax errors, pass a parameter and use the methods was the lowest. This means that most of the students could use variables before the experiment, while only a few students strongly agreed they were confident in applying programming methods with passing parameters and debugging logical and syntax errors.

The findings indicate that visualisation can make a positive difference in students' confidence regarding the Case A problem in general. The results showed that after the visualisation session (Figure 18 (b)) approximately 58% of the respondents strongly agreed they could apply variables; it was the question that recorded the highest number of respondents. Most of the students agreed more that they could use the procedure/methods, return values from the method and pass a parameter to the method. In addition, a substantial number of students were neutral regarding their ability to debug syntax errors. There was no remarkable improvement in the students' confidence in debugging errors, as the visualisation tool did not provide debugging errors. The other question that most students felt neutral about was their ability to write a program. However, students' confidence in writing programs increased after the visualisation session.

The results for the control group were somewhat the same before attending the lab session (Figure 18 (c)). After attending regular lab sessions (Figure 18 (d)), students became confident in how to call a method and pass parameters. The confidence achieved in understanding the methods was similar to the confidence achieved in the visualisation group. For instance, the increasing ration of the number of students from the control group who strongly agreed on how to call a method was 6.89% of the percentage of the students, which is approximately the same increasing ration achieved by the visualisation group. However, the visualisation group became more confidence in how to pass parameters to the method. The total percentage of students who agreed or strongly agreed on understanding how to pass parameters to a method was 58.62% and 54.84% for the

control and visualisation groups, respectively. However, this percentage rose to 65.5% and 70.9% for the control and visualisation groups, respectively.

Moreover, the visualisation group achieved more confidence in how to return values to a method. The percentage of students from the visualisation group who agreed or strongly agreed they understood how to return values to a method rose from 38.7% to 58%, while the percentage in the control group went from 48% to 51.7%. The results showed that the lab session did not affect the students' confidence in debugging errors and the use of variables. However, it reduced the confidence in performing arithmetic operations but improved the confidence in the rest of the topics, such as performing relational operations and how to write a program but to a lesser extent than the visualisation sessions.

Case B: classes and objects

The higher-level students studying more advanced programming were asked to solve the classes and objects problem. In the first part of the survey, more focused questions relating to the classes and objects were added to Case B, as follows.

- I can use classes and objects.
- I know how to use constructors.

In Case B, the students' confidence in general programming topics before the experiments varied across the participants (Figure 17).

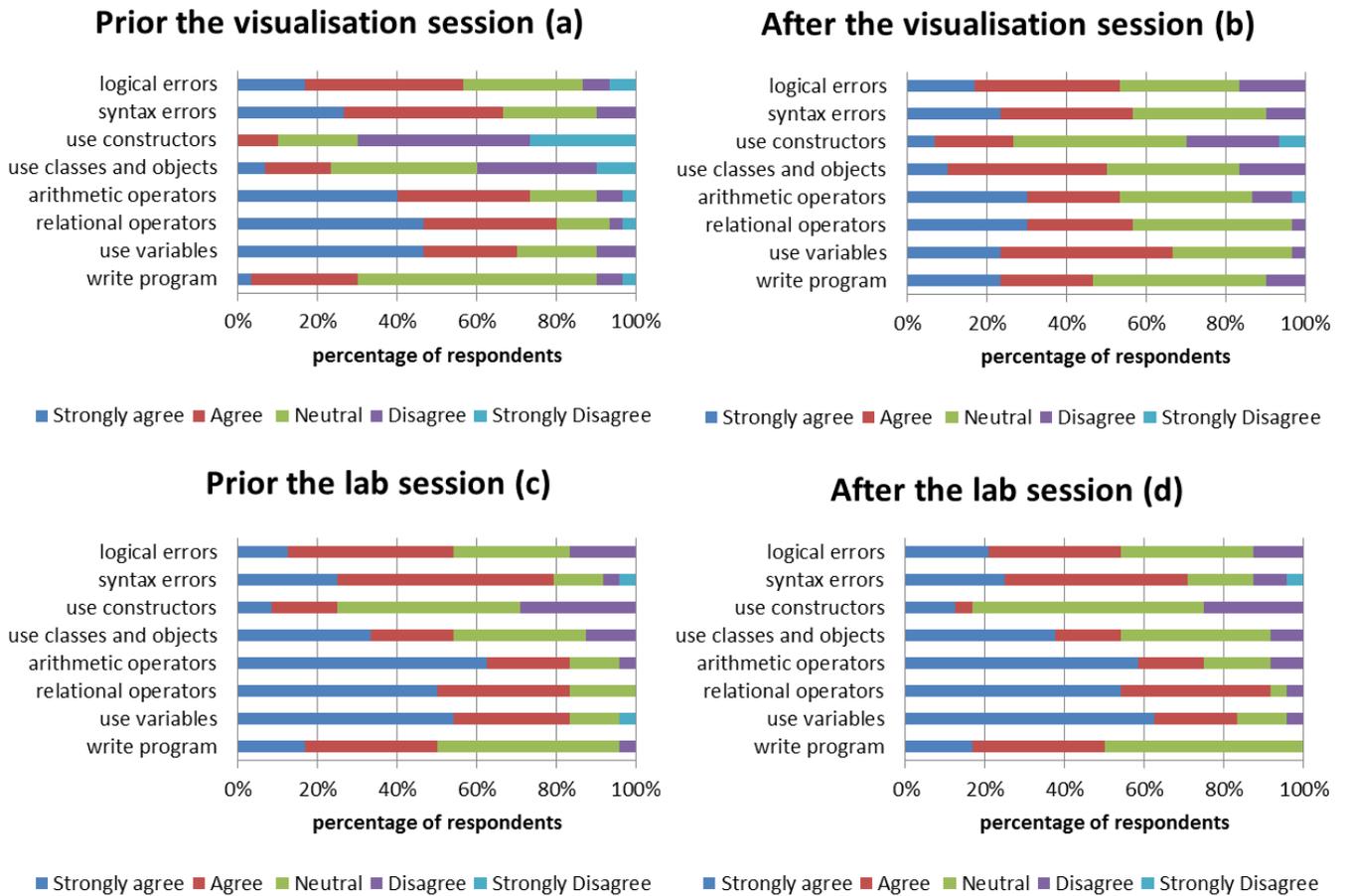


Figure 19. Case B – students’ confidence regarding general topics in programming

Most students in both groups were neutral regarding their ability to write a program in the pre-test (Figures 19 (a) and (c)). Most students strongly agreed they were able to use variables in relation to arithmetic and relational operators. However, they disagreed that they were able to use constructors. In the pre-test, 70% of the students in the visualisation group either disagreed or strongly disagreed that they were able to use constructors. After the visualisation session, their confidence level increased (Figure 19 (b)). It was observed that most were becoming neutral regarding their ability to use constructors, and the percentage who disagreed decreased to 30%. Before the experiment, the students had little confidence in their ability to use classes and objects (Figure 19 (a)); only 23% agreed they could use classes and objects. However, this percentage rose to 50% after attending the visualisation session (Figure 19 (b)).

The control group achieved limited confidence in understanding classes, objects and class constructors after attending regular lab sessions (Figure 19- c,d). An increment of one student strongly agreed regarding understanding the class constructor, while in the visualisation group there was an increment of two students for the same concept. Most of the students remain neutral toward the classes and objects topic.

Case C: class inheritance

The third group that was asked to solve the class inheritance was also asked about their ability to use class inheritance. Figure 20 shows the statistical analysis for the Case C group.

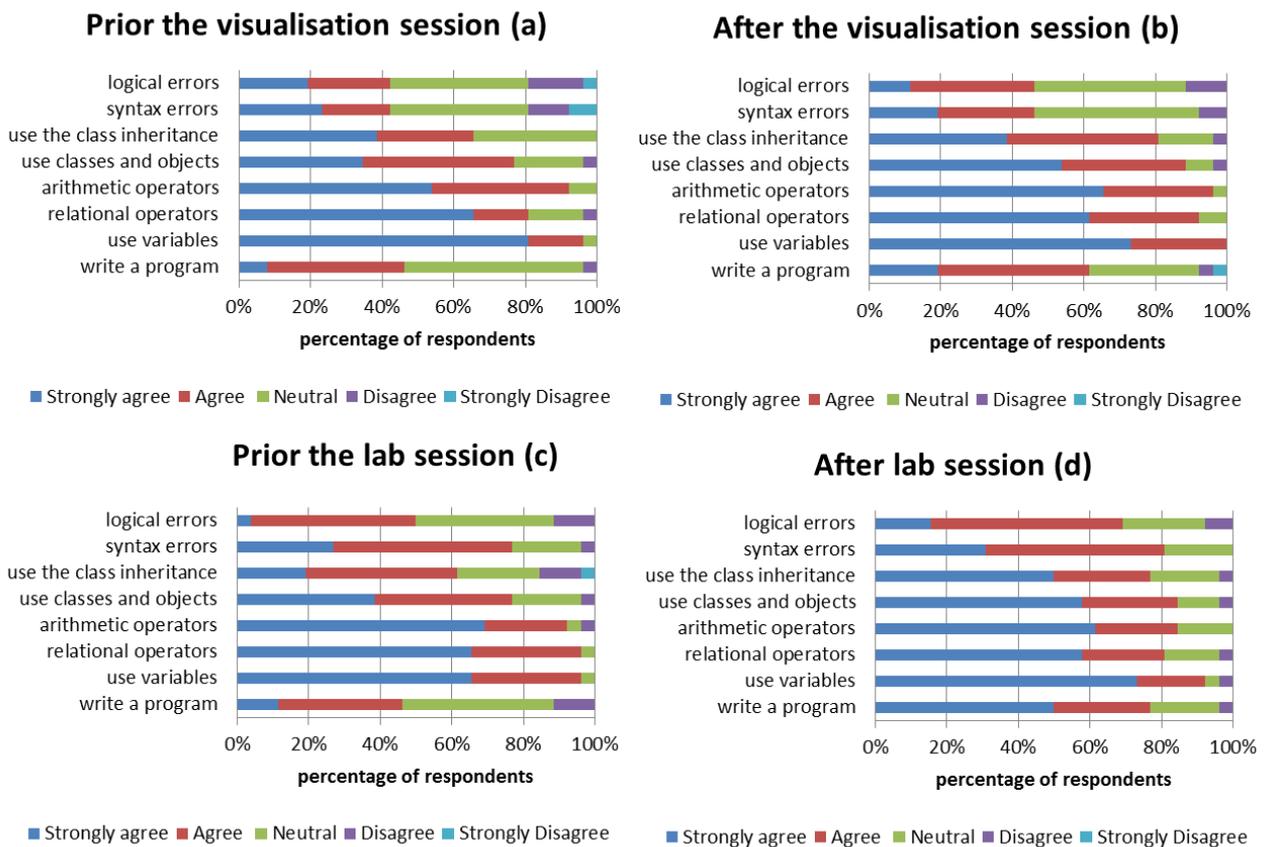


Figure 20. Case C - students' confidence regarding general topics in programming

In the pre-test for both groups (Figures 20 (a) and (c)), the students showed a high level of confidence in the general topic, such as identifying variables, performing arithmetic and relational operations, debugging errors and writing programs. This result was expected, as the students studied

the more advanced level of programming. After administering the visualisation session, the number of respondents who stated they were confident in their ability to use class inheritance increased. Before the session, 27% agreed they were confident in understanding class inheritance, but after the session the percentage rose to 42%. The control group showed more improvement after undertaking regular lab sessions, as the percentage of students who stated they were confident in understanding class inheritance increased from 19.23% to 50%. Therefore, the lab sessions contributed to a better understanding of the meaning of class inheritance.

1. Students' difficulties

The survey for both tests (pre- and post-test) had a question asking participants about the statements in the code for the given program they had difficulty understanding or solving. In the calling method problem, most of the respondents pointed to the calling method statement and the method header. For the visualisation group, the percentage of students who had difficulty with the calling methods decreased from 26% to 13%, while the percentage of students who struggled in tracing the method header decreased from 22.5% to 13%. For the control group, in the post-test the percentage of students who said they had difficulty in tracing the method calls decreased, while the percentage of students who were still having difficulty solving the method header question remained the same. Ten percent of the students were struggling with both statements in the pre-test, but that percentage decreased to 3.4% on the post-test.

In the classes and object problem, the statement the visualisation group highlighted as difficult was the statement `(account1.getBalance ())`. The percentage of students who were struggling with the statement decreased from 26% to 10% in the post-test. The control group referred to the call a method statement that is defined in objects such as `(account1.addMoney())`. However, the percentage of students who pointed out the problem decreased in the post-test from 16% to 12.5%.

Last, for the class inheritance problem both groups had difficulty with two statements: `(test.addition(input1,input2))` and `(example.subtraction(input1,input2))`; 30.7% of the control group students and 38% of the visualisation group students struggled with the statements. In the post-test,

the number of struggling control group students decreased to 3.8% and 11.5% for statement one and statement two, respectively, while the number of struggling visualisation group students decreased to 15% and 23% for statement one and statement two, respectively.

2. *Students' satisfaction*

Students in the visualisation group were asked to give their feedback after going through the visualisation of the three problems— Cases A, B and C. For the calling method (Case A), the students responded to the following statements:

- Using the tool helped me understand the procedure call .
- I would prefer to use the tool to trace the code rather than the manual approach.
- I would prefer to use the manual approach to trace the code rather than the tool.
- The tool was useful for learning Java.
- Using the tool helped me understand the changes in the variable values.
- The tool was easy to use.
- I would use a tool like this one.

Figure 21 shows the students' satisfaction regarding the statements above by indicating their level of agreement.

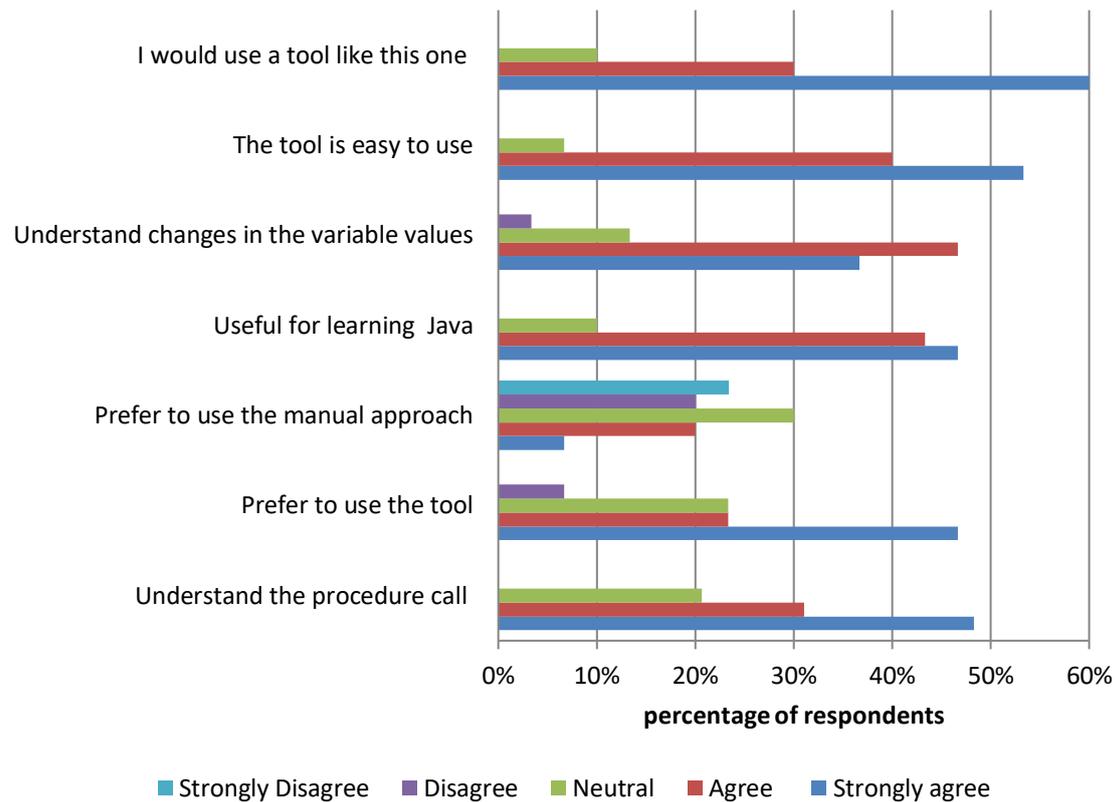


Figure 21. Satisfaction level of the participants regarding the use of the Visual Code Flow tool in the Case A problem

Many students strongly agreed they were satisfied with the experiment that helped them to perform the method calling questions and with how they passed parameters to the method. There was less satisfaction regarding preferring to use the manual approach to trace the program compared to using the Visual Code Flow tool. The students agreed the tool was easy to use, useful for learning Java and helped them to understand how the variables changed.

The Case B students were asked to give their opinion about the following statements:

- Using the tool helped me understand classes.
- Using the tool helped me understand objects.
- Using the tool helped me understand the difference between classes and objects.
- Using the tool helped me understand the class constructor.
- Using the tool helped me understand how to set the variable value defined in the class.
- Using the tool helped me understand how to get the variable value defined in the class.

- I would prefer to use the tool to trace the code rather than the manual approach.
- I would prefer to use the manual approach to trace the code rather than the tool.
- The tool was useful for learning Java.
- Using the tool helped me understand the changes in the variable values.
- The tool was easy to use.
- I would use a tool like this one.

A significant number of participants who solved the classes and objects problem said they were satisfied with the experiment with the visualisation method (Figure 22).

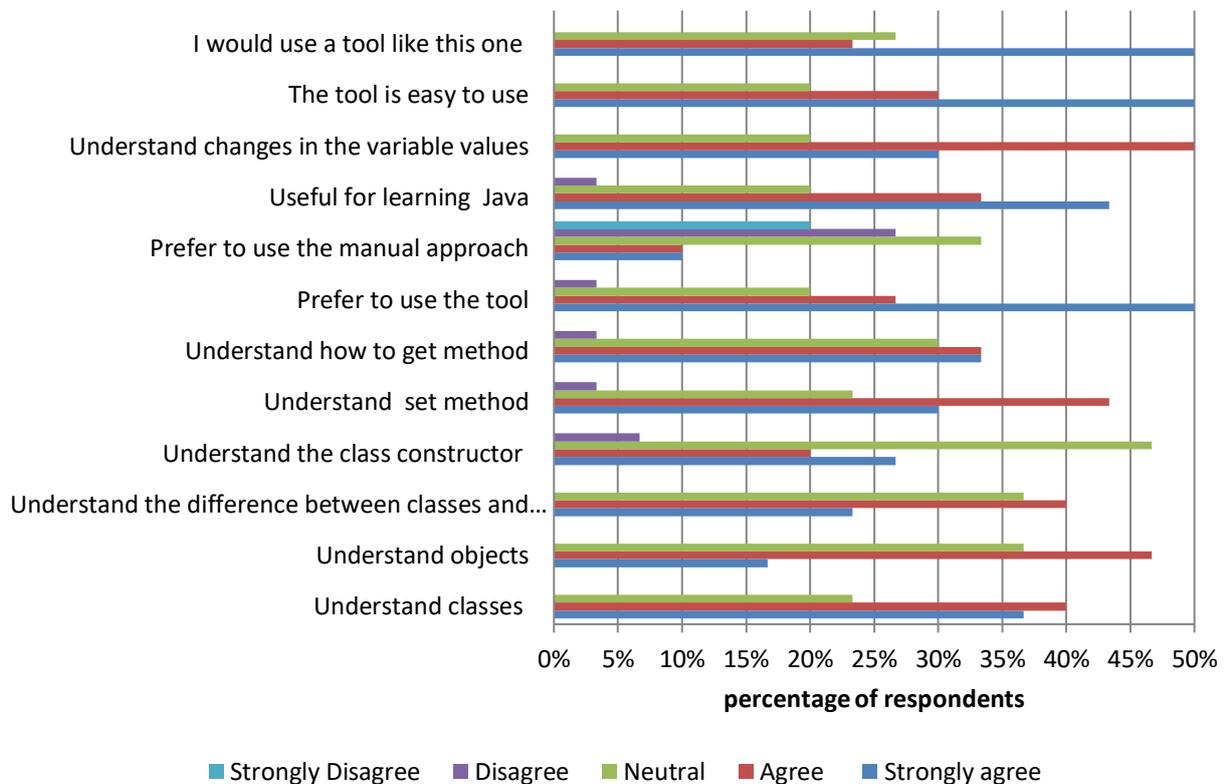


Figure 22. Satisfaction level of the participants regarding the use of the Visual Code Flow tool in the Case B problem

Most responses agreed on handling the differences between classes and objects. Students mostly agreed that visualisation helped them to understand, set and get variables’ values defined in a class. When asked about whether the visualisation helped them understand the class constructor, the majority of the responses were neutral.

Finally, the Case C students were asked to give their opinion regarding the following statements:

- Using the tool helped me understand class inheritance.
- Using the tool helped me understand how to set the value of the variable inherited from the superclass.
- Using the tool helped me understand how to get the value of the variable inherited from the superclass.
- I would prefer to use the tool to trace the code rather than the manual approach.
- I would prefer to use the manual approach to trace the code rather than the tool.
- The tool was useful for learning Java.
- The tool was easy to use.
- I would use a tool like this one.

Figure 23 shows the students' satisfaction regarding the statements above by indicating their level of agreement.

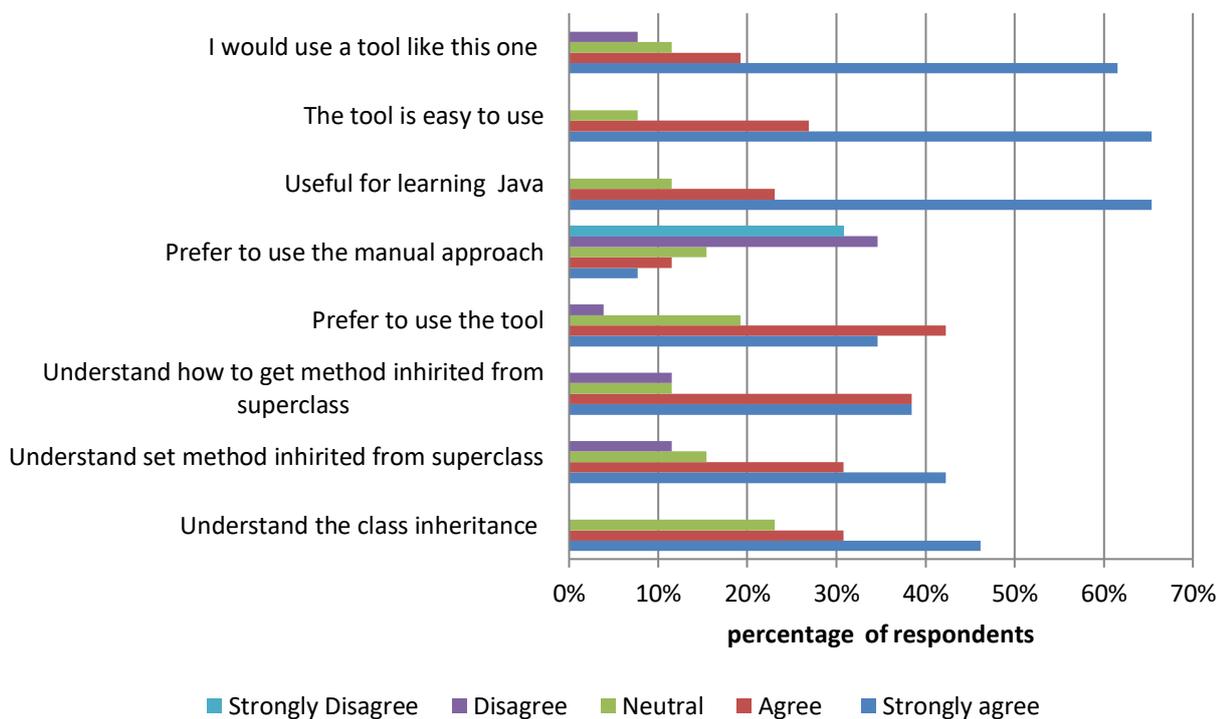


Figure 23. Satisfaction level of the participants regarding the use of the Visual Code Flow tool in the Case C problem

Students working on class inheritance strongly agreed they were satisfied with the visualisation. The majority strongly agreed that the tool helped them understand class inheritance and how to get

and set the inherited methods. For all three problems, the students who experienced the visualisation were satisfied and preferred to use the Visual Code Flow tool as an alternative to the manual approach. They agreed on the usefulness of the tool and its ease of use.

7.4.2 Qualitative findings

To better understand the quantitative findings, the researcher examined the results by conducting a focus group with the students and by interviewing experts. All the keywords determined during the interviews and the focus group were features in the presented visual tool (see Sections 5.4.1 and 5.4.4). In the focus group, the students described what they liked and disliked. Therefore, the researcher used those features as keywords for the analysis.

1. Focus group data

A constant comparison analysis was used to analyse the focus group data in this experiment. Three major stages characterise this type of analysis (Corbin and Strauss, 1990). In the first stage, the data were shortened into small units. A code was then assigned for each unit. In the second stage, the codes were grouped into categories. Finally, one or more themes that expressed the content of each of the groups were determined (Bishop-Clark et al., 2007).

The comments made in the focus group discussion were categorised into general comments and specific comments relating to the understanding of the three problems. General comments were about the design of the interface and usefulness, including the control of execution. Table 20 gives a summary of the general comments of the focus group regarding tool usage.

Table 20. Summary of content - general comments of the focus group regarding tool usage

Category	No. of participants	Sample comment (s)
Interface	A.2.1	<i>'The portions of the window was clear.'</i>
	C.3.4	<i>'The position of each window has to make sense.'</i>
	C.1.7	<i>'Colours were not distracting me.'</i>
Usefulness	B.1.1	<i>'Better than using the manual trace.'</i>
	C.1.6	<i>'Prefer the tracing that shows the steps of code flow.'</i>
	C.1.2	<i>'Great for tracing and having an application for the tool would be easier for students to use rather than a website.'</i>
	B.5.6	<i>'I would like to use this method after finishing each chapter.'</i>
	A.3.2	<i>'Very useful.'</i>
	A.5.1	<i>'I like the idea of the tool, and I think it will help me to learn Java.'</i>
	B.1.5	<i>'I may use the manual trace and then refer to the tool to make sure that my answer is correct.'</i>
	A.5.2	<i>'Make easier to understand and suggest to add visual examples of how the method can be used to program certain things like video games.'</i>
	B.3.3	<i>'Control buttons were clear and easy to understand.'</i>
Control of execution	C.1.4	<i>'The motion of going step by step was suitable.'</i>
	B.2.6	<i>'Prefer to control my tracing.'</i>

Regarding interviews conducted with the sample study, the researcher summarised their answers according to the following questions:

What are the strength(s) and weakness (es) of the tool in general?

When asking the students about the strengths and weaknesses of the tool, most of the discussion was about the tool's usefulness. The students mentioned the tracing feature and described it as useful, saying they may use it to improve the manual tracing that they usually use.

For instance, Participant C.1.2 said 'Great for tracing'. Participant B.1.5 preferred using it after using the manual trace to confirm his answers. The participants mentioned how the visualisation helped them to understand the three problems in some specific comments. The researcher categorised the specific comments relating to the problems.

How did you find the output representation? What things did you like and dislike?

When asking the students about the output representation, most of the discussion was that changing the values of a method's parameters inside the method would affect the values of variables in the main method. Moreover, the variable of output representation passed to the method has the same memory location as the method's parameters themselves. From the students' point of view, the variable representation similar to what was shown in class helped them to identify the value of the area and helped them to understand the calculated balance.

Participant A.4.5: *'I thought that the variables that have been passed to the method have the same memory location for the method's parameters themselves.'*

How did you find the animation? What things did you like and dislike?

When asking the students about animation, from their point of view changing the values of the method's parameters inside the method affects the values of variables in the main method. Further, the variables passed to the method have the same memory location as the method's parameters themselves.

Participant A.1.2: *'Before, I was thinking that changing the values of method's parameters inside the method will take effect on the values of variables in the main method.'*

How did you find the control of execution? What things did you like and dislike?

When asking the students about control of execution, they said that it made them understand the concept of classes and objects. Some preferred the play mode, and others liked having complete control over execution to trace line by line. Further, it helped define the concept of the class as abstract and to understand how it works. Control of execution also helped students to know the difference between class and object and to transfer the inherited object to the memory location. It facilitated self-learning and practicing after the lecture.

Participant B.3.3: *‘Control buttons were clear and easy to understand.’*

Participant B.2.6: *‘Prefer to control my tracing.’*

Participant C.1.4: *‘The motion of going step by step was suitable.’*

Case A: calling method

Students who solved the calling method problem described the animation and expression evaluation as helpful for understanding the passed parameter to the method. Table 21 gives a summary of the specific focus group comments for Case A students.

Table 21. Case A - summary of content-specific focus group comments

Category	No. of participants	Sample comment (s)
Animation	A.1.2	<i>‘Before, I was thinking that changing the values of method’s parameters inside the method will take effect on the values of variables in the main method.’</i>
	A.4.5	<i>‘I thought that the variables that have been passed to the method have the same memory location for the method’s parameters themselves.’</i>
Expression evaluation	A.3.4	<i>‘Helps me to identify the value of the area.’</i>

The animation helped the students resolve misunderstandings regarding the memory location for the parameters, which contributed to understanding the pass parameters. Of the students, 80.4% who solved the pass parameter problem mentioned the animation was helpful and that it improved their understanding of how the parameters transfer from the method call to the destination in the method header.

Participant A.1.2: *'Before, I was thinking that changing the values of the method's parameters inside the method will take effect on the values of variables in the main method.'*

Participants A.4.5: *'I thought that the variables that have been passed to the method have the same memory location for the method's parameters themselves.'*

The call method problem contains arithmetic operations that calculate the area. The participants noted the expression evaluation helped them to 'see' how the values of the variables on the right-hand side of the expression were recalled, replaced and calculated.

Participant A.3.4: *'Helps me to identify the value of area.'*

Case B: classes and objects

In the discussion with the students who solved the classes and objects problem, they mentioned how they had misunderstood the concept of class and object. They stated that visualisation helped them correct their misunderstandings regarding the class constructor and the representation of classes and object and to understand the expression evaluation. Table 22 gives a summary of the specific focus group comments of Case B students.

Table 22. Case B - summary of content-specific focus group comments

Category	No. of participants	Sample comment(s)
Animation	B.4.1	<i>‘It makes me understand the concept of classes and objects because I misunderstood them.’</i>
	B.4.4	<i>‘I knew the concept of the class as abstract, but now I understand how it works. Moreover, I understand how we can define more than one object from the same class, but each object has different memory locations so the changing in one attribute for one object will not affect the same attribute in the other objects.’</i>
Class constructor	B.2.3	<i>‘I was thinking that class constructor is a method that should be invoked same as any other method, the step by step tracing for the code makes me understand that class constructor is automatically invoked when creating the object.’</i>
Representation of classes and object	B.3.1	<i>‘Similar to UML representation.’</i>
	B.4.6	<i>‘Helps me know the difference between class and object.’</i>
	B.3.2	<i>‘Like the representation of data inside object.’</i>
	B.2.1	<i>‘Like the referring exists that relate the class to its objects.’</i>
Expression evaluation	B.4.5	<i>‘Understand the calculated balance.’</i>

The representation of the classes and objects used in the tool was the aspect most mentioned by the participants. The representation helped to distinguish between the class and the object.

The animation also helped in understanding the difference between the class and the object. Some of the participants mentioned they had difficulty understanding how the different objects derives from the same class located in memory.

Participant B.4.4: *‘I knew the concept of the class as abstract, but now I understand how it works. Moreover, I understand how we can define more than one object from the same class, but each object has different memory locations so the changing in one attribute for one object will not affect the same attribute in the other objects.’*

The participants mentioned how the visualisation and tracing the code helped them to understand the class constructor and when it should be invoked and take effect.

Participant B.2.3: *‘I was thinking that class constructor is a method that should be invoked same as any other method, the step by step tracing for the code makes me understand that class constructor is automatically invoked when creating the object.’*

Case C: class inheritance

In the discussion with the class inheritance students, they stated that they understood the scope of the subclass using the representation in the visualisation. Table 23 gives a summary of the specific focus group comments for Case C students.

Table 23. Case C - summary of focus group content-specific comments

Category	No. of participants	Sample comment (s)
Animation	C.2.1	<i>‘Like the transfer of the inherited object to the memory location.’</i>
Representation of inheritance	C.4.3	<i>‘Using shapes to represent inheritance with (box) inside (box) makes me understand what attributes are should be in the subclass.’</i>
	C.1.6	<i>‘Before, I thought that any change happens inside any inherited variable or method should be done to every copy exist in any other object for the same variable or method.’</i>

The representation of the inheritance contributed to improving the students' comprehension. The participants said using the boxes helped them to know the scope of the superclass and the subclass. It helped in identifying where any change took place.

2. Interview data

The purpose of the expert interviews was to gather experts' opinions on using visualisation during lectures or lab activities and whether it could help improve performance. This qualitative data analysed the first coding process through initial and final coding. Text analysis was used to conduct the first cycle of coding to determine keywords and phrases that were common amongst interviewees. Keywords and phrases appeared as word clouds, which were then analysed and encoded with suitable category labels. The second cycle of coding then compared the keywords to find similarities and differences. The second cycle pattern coding method was used to recognise similarly coded data and summarise it into sub-categories or to consolidate it (Saldaña, 2016).

The researcher examined the data from the expert interviews and categorised it into groups. The main comments were about five fundamental issues regarding how visualisation will lead to better understanding. Table 24 summarises the categories of the issues raised and gives sample comments.

Table 24. Summary of content comments from the experts' interviews

Category	Interviewee	Sample comments
Design of the visualisation	Expert 1	<i>'Using different shapes and styles to represent variables, objects, and classes will help the students to distinguish the difference between them.'</i>
	Expert 2	<i>'The structure and branching that have been used in the memory frame will help in understand which item is belong to what method or class , for example the student can recognise that variable (width) and variable (length) belongs to the method (main) which belongs to class (parameterExample) also the objects (test) is an object from class (MathsLevel1) where object (example) is an object from class (MathsLevel2).'</i>
	Expert 3	<i>'I like the highlighting of every single statement before, and during the execution, that will show the flow of the code and give enough time for the student to understand what happens during execution.'</i>
Usefulness	Expert 2	<i>'I would like to use the visualisation as a supplement in lectures or even before starting the lab activity to improve the student's comprehension.'</i>
	Expert 6	<i>'It is easy to use and doesn't need any training sessions so the students can use it as self-learning'</i>
	Expert 4	<i>'During lectures, I keep on drawing shapes and arrows to show the variables and their changes, this tool will consume my effort and time.'</i>
Calling methods	Expert 5	<i>'Showing how the parameters travel from where it is calling to memory and finally to their place in the header.'</i>
	Expert 1	<i>'Helps to understand that sending and receiving parameters have different memory location.'</i>
Classes and objects	Expert 2	<i>'The difference between the classes and objects are one of the challenges in programming teaching; I think using visualisation to present the class and object with different shapes and format will be very helpful.'</i>
	Expert 5	<i>'Students now become able to know the difference between object and class.'</i>
Class inheritance	Expert 6	<i>'Showing how the attributes inherited from the superclass by using drawing and animation will add support for me when teaching the class inheritance lesson since it always confuses the students.'</i>

	Expert 3	<i>‘Drawing the inherited object inside the super-class object was very clear and meaningful.’</i>
--	----------	--

All six interviewees commented about how the design of the visualisation and animation may help improve understanding.

Expert 1: *‘Using different shapes and styles to represent variables, objects, and classes will help the students to distinguish the difference between them.’*

Expert 2: *‘The structure and branching ...’*

Expert 3: *‘I like the highlighting of every single statement ...’*

Some of the interviewees suggested using the visualisation as a supplement to support their lectures.

Expert 2: *‘I would like to use the visualisation as a supplement in lectures or even before starting the lab activity to improve the student’s comprehension.’*

The interviewees agreed on the three selected problems with which the students struggled. All the experts liked the tracing that showed how the parameters transferred and how they changed. For instance:

Expert 5: *‘Showing how the parameters travel from where it is calling to memory and finally to their place in the header.’*

They liked the representation of the classes and objects. However, one expert suggested using a different shape for the inherited class.

Expert 2: *‘Why you use the same shape to represent the object test and object example.’*

Moreover, they liked the representation of class inheritance. However, two of them suggested using the same class hierarchy in the UML diagram, which uses the branching from the superclass.

Expert 1: *‘The students familiar with the class hierarchy in the UML, so I suggested to use it in the class inheritance representation.’*

The interviewees suggested other concepts, such as loops, array and recursion.

Expert 2: *'I suggested to develop an example about loops and recursions; this will help the students to keep track of the loop counter and show how the condition test should be made.'*

Expert 4: *'I prefer to use visualisation with arrays to represent the rang and contents of each element of the array.'*

7.5 Conclusion

Despite the availability of visualisation in programming learning, continuous evaluation for better development is required. The research in this study aimed to evaluate visualisation from the perspectives of novices and experts to improve the visualisation tool. Improvement of the tool can help students learn how to program and learn fundamental programming concepts. Based on the results, the researcher can propose visualisation as an effective method that can be considered as a teaching approach. The major finding of this study is that visualisation positively affected students' programming comprehension and helped in assessing the students' confidence regarding programming concepts and solving programming problems.

The study suggests expanding the scope of the presented problems in the Visual Code Flow tool. The experts in the field suggest including other threshold concepts, such as the use of a loop, an array and recursions.

There are certain limitations to the study. The students only used visualisation for one hour (during lecture time), which was too short; they needed more time to try the tool. Another limitation is that the scope of the tool covered three examples that were limited to three programming concepts. Further studies can investigate and evaluate various problems, such as control statements and arrays.

Chapter 8 CONCLUSIONS AND FUTURE WORK

8.1 Introduction

The following sections discuss the importance of visualisation in programming learning and the necessity for continued improvement to meet novices' needs. The main objective of this research was to define an approach to increase novices' comprehension when learning programming and thereby improve their performance. An additional aim was to expand and improve the model. The research investigated the obstacles and challenges novice programmers face in understanding programming concepts generally and in existing programming learning systems. The proposed approach is based on two main concepts—pedagogy theory (threshold concepts) and the benefits of using visualisation in programming learning. The proposed approach was evaluated based on such aspects as novices' performance and confidence in solving programming problems and in understanding programming concepts. Expert evaluation was provided for further improvement. The objective was achieved by defining the gap between the threshold concepts and novice programmers' needs after carefully reviewing the most appropriate approaches. Thereafter, a comprehensive visualisation tool was designed and was then evaluated by students and experts.

8.2 Contributions and Achievement of the Research

Overall, the research achieved all the objectives set out in Chapter 1 using a series of experiments in the first and second phases and studies to examine the learning environment. The main achievements of this research are as follows.

- It established a current understanding of strengths and weaknesses in programming learning systems, specifically visualisation systems and some existing tools that support the teaching of programming.
- It defined the attributes of current visualisation methods and assessed the necessary attributes required for the new framework.

- It explored pedagogy theories related to the threshold concepts to discover the challenges for novice programmers.
- It developed and implemented the proposed framework based on the existing tools and novice programmers' needs.
- It evaluated the proposed framework by obtaining novices' feedback and experts' opinions to ensure it is appropriate for novice programmers' requirements.

Overall, the study contributed to finding what programming learners need in a visualisation tool. The study defined the features and characteristics of a visualisation tool in order to refine it based on the learners' perspectives. The study also contributed to identifying some of the threshold concepts in programming that visualisation can help learners to understand better. It also evaluated the visualisation tool to determine the impact of visualisation on learners' performance, confidence while solving programming problems and development in terms of understanding general programming concepts.

8.3 The Research Questions

The achievements in terms of answering the research questions can be summarised as follows.

1. What is the impact of using program visualisation on students' performance while tracing the program?

The major finding of this study was that visualisation positively affected students' programming comprehension. However, a significant difference was observed between the performance of the control group and that of the visualisation group. The improvement in the visualisation group was more remarkable than that in the control group. The reasons for this can be explained from different perspectives.

According to Mselle and Twaakyondo (2012), MTL permits novices to illustrate code execution as the machine does it. Students feel the machine can be viewed as a human that can respond and anticipate what the code is doing. The study findings support this idea. This can explain the improvement in the students' scores. Most of the students in the visualisation group

indicated that visualisation and animation helped them understand object-oriented programming. Therefore, the present study has proven what was discussed by researchers such as Hagan and Markham (2000), Holliday and Luginbuhl (2004) and Sun (2010).

2. What is the impact of using program visualisation on students' comprehension of general programming concepts?

Assessing students' comprehension of general programming concepts is one of the aims of this study. Therefore, the researcher measured the comprehension of the students before and after using visualisation by asking them general questions about their understanding of some of the programming topics. The results indicated that students' comprehension increased after attending the visualisation session. This can be seen in the students' answers to the general questions at the beginning of the survey.

3. What is the impact of using program visualisation on students' confidence while they are solving programming problems?

Another aim of this study was to assess students' confidence in solving programming problems. There is a strong relationship between committed errors in programming and students' confidence. The study by Mselle and Twaakyondo (2012) showed how visualisation reduced the number of errors students committed. The present study showed how visualisation increased students' confidence and consequently reduced mistakes. The survey asked the students about their confidence when answering each step in the programming problem. In comparing the responses about confidence before and after the visualisation, an improvement in the students' confidence while solving the programming problems can be seen.

8.4 Limitations of the Research

Although the objectives of the research program have been met, a number of decisions had to be made that imposed limitations on the work. These decisions were typically a result of either practical constraints or time restrictions. The key limitations of the research are summarised below.

- 1- The number of programming problems considered in the investigation and in the visual code flow tool implementation was limited to three. The researcher believes that expanding the scope of the programming problems and investigating other threshold concepts in programming would have provided more data and allowed a more in-depth exploration of more difficulties faced by programming learners.
- 2- In both experiments, the participants did not have the chance to practise using the visualisation tool for longer than the time allotted for the interview or experiments. Allowing them to use the visualisation tool for a longer period would allow for a better evaluation of the tool.
- 3- The research population in the study was not generalisable across the worldwide population, as the participants were from one area. Moreover, the sample size used in conducting qualitative research in the interviews and focus groups was limited. Because of the nature of the study, it was difficult to gather a large sample for the interviews, as that would have consumed significant time and effort. The kind and size of the research population could affect the reliability of the results, which would increase the variability and may lead to bias.

8.5 Future Research

There are a number of areas that require future work, both specifically related to this research and more generally to visualisation.

- 1- Other programming problems that students usually struggle with, such as arrays and recursion, should be investigated.

- 2- Empirical research to compare the approach in this study with approaches in related works could be conducted. Such a comparison would uncover more strengths and/or weaknesses of all the various approaches that benefit visualisation in programming learning.
- 3- The Visual Code Flow tool needs to be improved in terms of its capacity to find and debug errors. Error debugging is a potentially valuable measure of students' performance and has an impact on their comprehension.
- 4- Other methodologies should be used to gather quantitative data about performance, task duration and tool usage, such as automatic recording or video analysis. The researcher suggests quantitative data, such as counting the number of errors committed while writing a program or the time needed to solve or trace a programming question. This kind of data may provide more evidence on performance and confirm the study findings.
- 5- The researcher suggests extending the study to cover various international sites and expanding the sample size for both quantitative and qualitative research. The larger the sample, the more valid the research results will be.

8.6 The Importance Of Visualisation In Programming Learning

It has become clear that programming is a challenge to most students and that mastering the basics of a programming language is a huge obstacle for many. Traditional teaching methods are often passive and thus do not engage students in active learning with dynamic program execution. Programming is dynamic by nature; however, traditional learning materials have a static format. Using visualisation in programming learning showed that it contributes to a better understanding and improvement in programming performance among students. Based on the literature reviewed and on the study findings, the researcher proved that a significant percentage of students achieved better results when they were using a software visualisation tool. The researcher believes that visualisation should be continuously improved to keep up with both students' needs and the evolution of programming languages.

The future of visualisation implementations will have to consider diverse programming issues, such as side effects in expression evaluations and data dependencies. Students should be able to learn from their mistakes by representing them during dynamic execution. The continued growth of visualisation tools and research into their usability and ability to improve students' comprehension is both necessary and expected.

8.7 Research Dissemination

While there are many studies to prove the advantages of using visualisation in programming learning, the present study emphasises the need to introduce visualisation as an official aspect at academic institutions. Visualisation in programming learning is widely accepted; however, it is not recognised by many programming learners, as was shown in the interviews conducted with the programming students in Chapter 6. Therefore, efforts to include visualisation in programming learning should continue in order to disseminate visualisation among programming learners. Academic institutions can further this cause by introducing and using visualisation in lectures and computer labs. Continuing to have conferences and publications about the importance of using visualisation in programming learning can facilitate this. The research conducted has considered the need for visualisation and has proposed means that can contribute to its improvement.

The impact of this study is that it encourages educational institutions to use visualisation as a method to facilitate the learning of programming. This is particularly important in Saudi Arabia, where the visualisation method is little known. This study will hopefully help introduce the use of visualisation to Saudi programming learners. The study explains the benefits of using visualisation and how it will positively affect programming learners' performance and programming knowledge.

References

- Adam, A., and Laurent, J.-P. (1980). LAURA, a system to debug student programs. *Artificial Intelligence*, 15, pp. 75-122. [http://doi.org/10.1016/0004-3702\(80\)90023-5](http://doi.org/10.1016/0004-3702(80)90023-5).
- Al-Gahtani, S. S. (2004). Computer technology acceptance success factors in Saudi Arabia: an exploratory study. *Journal of Global Information Technology Management*, 7(1), pp. 5-29.
- Aktunc, O. (2013). A Teaching Methodology for Introductory Programming Courses using Alice, 3(1),pp. 350-353.
- Al-imamy, S., Alizadeh, J., and Nour, M. a. (2006). On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process. *Journal of Information Technology Education*, 5, pp. 271-283. Retrieved from <http://www.editlib.org.ezproxy.psz.utm.my/p/111545/>.
- Al-Othman, A. and Almawash, F., (2020). The Impact of Teaching Programming by using Scratch on Self-motivation towards Learning Programming for Primary School Students in Riyadh. *Journal of Educational and Psychological Studies [JEPS]*, 14(1), pp.54.
- Amineh, R. J., and Asl, H. D. (2015). Review of constructivism and social constructivism. *Journal of Social Sciences, Literature and Languages*, 1(1), pp. 9-16.
- Alakeel, A. M. (2015). Investigating Difficulties of Learning Computer Programming in Saudi Arabia. *Universal Journal of Educational Research*, 3(9), pp. 567-577.
- Ala-Mutka, K. M. (2004). Problems in learning and teaching programming-a literature study for developing visualizations in the Codewitz-Minerva project. *Codewitz Needs Analysis*,pp. 1-13.
- Anderson, J. R., and Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Commun. ACM*, 29(9),pp. 842-849.
- Aparicio, M., Bacao, F., and Oliveira, T. (2016). An e-learning theoretical framework. *An e-learning theoretical framework*(1), pp. 292-307.
- Arango-Muñoz, S. (2015). Joëlle Proust: The Philosophy of Metacognition: Mental Agency and Self-Awareness. In: Springer.
- Alhammad, S., Atkinson, S., and Stuart, L. (2016). The role of Visualisation in the study of Computer Programming. In *27th Annual Workshop of the Psychology of Programming Interest Group - PPIG 2016* ,pp. 5-16. Cambridge,UK.
- Allen, E., Cartwright, R., and Stoler, B. (2002). DrJava: A lightweight pedagogic environment for Java. Paper presented at the SIGCSE.
- Alshenqeeti, H. (2014). Interviewing as a Data Collection Method: A Critical Review. *English Linguistics Research*, 3(1),pp. 39-45. <http://doi.org/10.5430/elr.v3n1p39>.
- Aytekin, M. C. (2019). Construction and visualization of concept prerequisite graphs for e-learning (Doctoral dissertation).
- Back, G.G. and Dietrich, G.S., 2017. Market Trends. In *Handbook of Incineration of Hazardous Wastes (1991)* ,pp. 59-66. CRC Press.
- Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., and Slominski, A. (2017). Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing* ,pp. 1-20: Springer.
- Barnes, D. J., Kölling, M., and Gosling, J. (2006). *Objects First with Java: A practical introduction using BlueJ*: Pearson Prentice Hall London.
- Bode, R., and Hahn, N. (2015). *StratEx Environmental Testing*.
- Bennedsen, J., and Schulte, C. (2010). BlueJ Visual Debugger for Learning the Execution of Object-Oriented Programs? *ACM Transactions on Computing Education*, 10(2),

- pp.1-22. <http://doi.org/10.1145/1789934.1789938>.
- Beatty, K. (2013). *Teaching and researching: Computer-assisted language learning*. Routledge.
- Behera, A. K., Verbert, J., Lauwers, B., and Duflou, J. R. (2013). Tool path compensation strategies for single point incremental sheet forming using multivariate adaptive regression splines. *Computer-Aided Design*, 45(3), pp. 575-590.
- Bertram, D. (2006). *Likert Scales: CPSC 681—Topic Report*. Poincare, pp.1-11. <http://doi.org/10.1002/9780470479216.corpsy0508>
- Bishop-Clark, C., Courte, J., Evans, D., and Howard, E. (2007). A Quantitative and Qualitative Investigation of Using Alice Programming to Improve Confidence, Enjoyment and Achievement Among Non-Majors. *Journal of Educational Computing Research*, 37(2), pp.193-207. <http://doi.org/10.2190/J8W3-74U6-Q064-12J5>
- Bitzer, D. L., Hicks, B. L., Johnson, R. L., and Lyman, E. R. (1967). The Plato System: Current Research and Developments. *IEEE Transactions on Human Factors in Electronics*, HFE-8(2), pp.64-70. <http://doi.org/10.1109/THFE.1967.233313>
- Blandford, A. (2013). Semi-Structured Qualitative Studies. *The Encyclopedia of Human-Computer Interaction*, 2, pp.53.
- Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. Paper presented at the Proceedings of the 1st international conference on learning analytics and knowledge.
- Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., and Zander, C. (2007). Threshold concepts in computer science. *ACM SIGCSE Bulletin*, 39(1), pp. 504. <http://doi.org/10.1145/1227504.1227482>.
- Bryman A (2012) *Social Research Methods*. Fourth edition. Oxford University Press, Oxford.
- Bryman, A. (2017). *Social Research Methods*. Oxford University Press (Vol. 91).
- Brinkmann, S. (2014). Doing without data. *Qualitative Inquiry*, 20(6), pp.720-725.
- Brusilovsky, P. (1994). The Construction and Application of Student Models in Intelligent Tutoring Systems. *The Construction and Application of Student Models in Intelligent Tutoring Systems*, 32(1), pp. 70-89.
- Bühlmann, M. (2011). The Quality of Democracy; C Rises and Success Stories, 49(1), 123-128.
- Campbell, A., Catto, G., and Hansen, E. (2003). Language-independent interactive data visualization. *ACM SIGCSE Bulletin*, pp.215-219. Retrieved from <http://dl.acm.org/citation.cfm?id=611972>.
- Capper, J. (2001). E-Learning Growth and Promise For the Developing World. *TechKnowLogia*, May/June, 7-10. Retrieved from www.TechKnowLogia.org.
- Charmaz, K., and Belgrave, L. L. (2007). Grounded theory. *The Blackwell encyclopedia of sociology*.
- Conti, S., Peruginelli, G., and Francesconi, E. (2019). The e-learning approach and visualisation techniques in the judicial area. *J. Open Access L.*
- Corbin, J., and Strauss, A. (1990). Grounded Theory Research: Procedures, Canons and Evaluative Criteria. *Zeitschrift Fur Soziologie*, 19(6), pp.418-427. <http://doi.org/10.1007/BF00988593>.
- Corbin, J., and Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1), pp.3-21.
- Corbin, J., and Strauss, A. (2008). *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, Inc., pp.1-23. <http://doi.org/http://dx.doi.org/10.4135/9781452230153>.
- Crotty, M. (1998). *The foundations of social research*. London: Sage.

- Creswell, J. (2014). *Research Design: qualitative, quantitative, and mixed methods approaches*. SAGE (4th ed., Vol. 91).
- Creswell, J. W. (1994). *Research design: Qualitative and quantitative approaches*. Thousand Oaks, Calif: Sage Publications.
- Creswell, J. W. (1998). *Qualitative inquiry and research design: Choosing among five traditions*. Thousand Oaks, Calif: Sage Publications.
- Daniel, E. (2016). The Usefulness of Qualitative and Quantitative Approaches and Methods in Researching Problem-Solving Ability in Science Education Curriculum, 7(15), pp. 91-100.
- Davies, P. (2003). Threshold Concepts: how can we recognise them? European Association in Learning and Instruction Conference EARLI, 44, pp.1-21. <http://doi.org/10.4324/9780203966273>
- Dekson, D. E., Suresh, E. S. M., and Ponnusamy, R. (2009). Intelligent system to teach programming languages. 2009 International Conference on Intelligent Agent and Multi-Agent Systems, IAMA 2009. <http://doi.org/10.1109/IAMA.2009.5228014>
- Dixon, M. (2004a). Code-Memory Diagram Animation Software Tool: Towards on-Line Use. Proceeding of the IASTED International Conference WEB-BASED EDUCATION, (February 16-18, 2004, Innsbruck, Austria), pp. 601-603.
- Dixon, M. (2004b). Generic Code-Memory Diagram Animation Authoring Tool: Impact on Learning and Teaching Object-Oriented Programming.
- Durzo, J., (1978) . Basic considerations for implementing instructional development programs in higher education: Some suggestions from the literature. *Journal of Instructional Development*, 1(2), pp.30-35.
- Ebrahimi, A., Geranzeli, S., Shokouhi, T., and Tee, E. R. (2013). Programming for children: "Alice and Scratch analysis". Paper presented at the 3rd International Conference on Emerging Trends of Computer and Information Technology (ICETCIT), Singapore.
- Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., and Zander, C. (2006). Putting threshold concepts into context in computer science education. *ACM SIGCSE Bulletin*, 38(3), pp.103. <http://doi.org/10.1145/1140123.1140154>
- Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K., Thomas, L., and Zander, C. (2007). From Limen to Lumen: Computing students in liminal spaces. *Proceedings of the Third International Workshop on Computing Education Research - ICER '07*, pp.123-132. <http://doi.org/10.1145/1288580.1288597>.
- Eliot and Associates (2007). (2005). Guidelines for conducting a focus group.
- Etheredge, J. (2004). CMeRun. *ACM SIGCSE Bulletin*, 36(1), pp.22. <http://doi.org/10.1145/1028174.971311>
- Evaluation Research Team. (2008). Data Collection Methods for Program Evaluation: Focus Groups. *Evaluation Briefs*, (13), Retrieved from <http://www.cdc.gov/healthyyouth/evaluation/index.htm>.
- Evangelidis, G., Dagdilelis, V., Satratzemi, M., and Efopoulos, V. (2001). X-compiler: Yet another integrated novice programming environment. *Proceedings - IEEE International Conference on Advanced Learning Technologies, ICALT 2001*, pp.166-169. <http://doi.org/10.1109/ICALT.2001.943890>.
- Franzoni, A. L., Assar, S., Defude, B., and Rojas, J. (2008, July). Student learning styles adaptation method based on teaching strategies and electronic media. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*, pp. 778-782. IEEE.
- Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, 18(2), pp.93-116. <http://doi.org/10.1080/08993400802114508>
- Fouh, E., Akbar, M., Shaffer, C. A., and Tech, V. (2012). The Role of Visualization in

Computer Science Education.

- Georgantaki, S., and Retalis, S. (2007). Using educational tools for teaching object oriented design and programming. *Journal of Information Technology Impact*, 7(2), pp. 111-130.
- Ghirardini, B. (2011). E-learning methodologies: A guide for designing and developing e-learning courses. Food and Agriculture Organization of the United Nations (FAO). <http://doi.org/12516E/1/11.11>.
- Gill, P., Stewart, K., Treasure, E., and Chadwick, B. (2008). Methods of data collection in qualitative research: Interviews and focus groups. *British Dental Journal*, 204(6), pp. 291-295. <http://doi.org/10.1038/bdj.2008.192>.
- Glaser, B., and Strauss, A. (1967). *The discovery of grounded theory: Strategies for qualitative research*. London, UK: Weidenfeld and Nicholson.
- Gliner, B. E., Wyler, A., Fowler, B., Sheffield, W. D., Kuntz, R., Leyde, K., and Sloan, L. R. (2009). U.S. Patent No. 7,483,747. Washington, DC: U.S. Patent and Trademark Office.
- Gomes, A., and Mendes, A. J. (2007). An environment to improve programming education. *Proceedings of the 2007 International Conference on Computer Systems and Technologies*. ACM, 1. <http://doi.org/10.1145/1330598.1330691>.
- Gouyon, F., and Dixon, S. (2005). A review of automatic rhythm description systems. *Computer music journal*, 29(1), 34-54.
- Gray, K., and Flatt, M. (2003). ProfessorJ: a gradual introduction to Java through language levels. *Companion of the 18th Annual ACM SIGPLAN*, 170-177. <http://doi.org/10.1145/949344.949394>.
- Gries, P., Mnih, V., Taylor, J., Wilson, G., and Zamparo, L. (2005). Memview: A Pedagogically-Motivated Visual Debugger. *Proceedings Frontiers in Education 35th Annual Conference*, S1J-11-S1J-16. <http://doi.org/10.1109/FIE.2005.1612204>.
- Guba, E. G. (1990). *The paradigm dialog*. In *Alternative Paradigms Conference*, Mar, 1989, Indiana U, School of Education, San Francisco, CA, US. Sage Publications, Inc.
- Gudmundsen, D., Olivieri, L., and Sarawagi, N. (2011). Learn how to use executable flowcharts to enhance learning in general education, CS0, and CS1 courses: tutorial presentation. *Journal of Computing Sciences in Colleges*, 26(6), pp.107-109.
- Guo, P. J. (2013, March). Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pp. 579-584. ACM.
- Harris, J., Mishra, P., and Koehler, M. (2009). Teachers' technological pedagogical content knowledge and learning activity types: Curriculum-based technology integration reframed. *Journal of research on technology in education*, 41(4), pp.393-416.
- Hagan, D., and Markham, S. (2000). Teaching Java with the BlueJ environment. *Proceedings of Australasian Society for Computers in Learning in Tertiary Education Conference ASCILITE 2000*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.1985andrep=rep1andtype=pdf>.
- Hatziapostolou, T., and Paraskakis, I. (2010). Enhancing the impact of formative feedback on student learning through an online feedback system. *Electronic Journal of e-Learning*, 8(2), pp. 111-122.
- Holliday, M. a., and Luginbuhl, D. (2004). CS1 assessment using memory diagrams. *ACM SIGCSE Bulletin*, 36(1), 200. <http://doi.org/10.1145/1028174.971373>
- Holliday, M. A., and Luginbuhl, D. (2003). Using Memory Diagrams When Teaching a Java-Based CS1.
- Hongwarittorn, N., and Krairit, D. (2010). Effects of Program Visualization (Jeliot3) on Students' Performance and Attitudes towards Java Programming. *The Spring 8th International Conference on Computing, Communication and Control Technologies*, (August), pp. 6-9.

- Hsia, J. I., Simpson, E., Smith, D., and Cartwright, R. (2005). Taming Java for the classroom. *ACM SIGCSE Bulletin*, 37(1), pp. 327-331.
- Husain, M., Tarannum, N., and Patil, N. (2013). Teaching programming course elective: A new teaching and learning experience. *IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, pp. 275-279. <http://doi.org/10.1109/MITE.2013.6756349>.
- Ishizue, R., Sakamoto, K., Washizaki, H., and Fukazawa, Y. (2018). PVC: Visualizing C programs on Web browsers for novices. Paper presented at the Proceedings of the 49th ACM Technical Symposium on Computer Science Education.
- Jamieson, S. (2004). Likert scales: how to (ab) use them. *Medical education*, 38(12), pp. 1217-1218.
- Kallia, M., and Sentance, S. (2017). Computing Teachers' Perspectives on Threshold Concepts. In *Proceedings of the 12th Workshop in Primary and Secondary Computing Education*, pp. 15-24. <http://doi.org/10.1145/3137065.3137085>
- Karnalim, O., and Ayub, M. (2018). The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor. *CommIT (Communication and Information Technology) Journal*, 11(2), pp. 67. <http://doi.org/10.21512/commit.v11i2.3704>
- Kasurinen, J., Purmonen, M., and Nikula, U. (2008). A Study of Visualization in Introductory Programming. *Ppig '08, (Winslow 1996)*, pp.181-194.
- Khalife, J. T. (2006). Threshold for the introduction of programming: providing learners with a simple computer model. *28th International Conference on Information Technology Interfaces, 2006.*, 71-76. <http://doi.org/10.1109/ITI.2006.1708454>.
- Kent, A., Hutcheon, I., Ryerson, F., and Phinney, D. (2001). The temperature of formation of carbonate in Martian meteorite ALH84001: Constraints from cation diffusion. *Geochimica et Cosmochimica Acta*, 65(2), p. 311-321.
- Khan, S. N. (2014). Qualitative research method: Grounded theory. *International Journal of Business and Management*, 9(11), 224-233.
- Kolb, S. M. (2012). Grounded Theory and the Constant Comparative Method: Valid Research Strategies for Educators. *Journal of Emerging Trends in Educational Research and Policy Studies*, 3(1), pp. 83-86. <http://doi.org/10.1.1.301.9451andrep=rep1andtype=pdf>
- Kölling, M. (2018). Blue, BlueJ, Greenfoot: Designing Educational Programming Environments. In S. Goschnick (Ed.), *Innovative Methods, User-Friendly Tools, Coding, and Design Approaches in People-Oriented Programming*, pp. 42-87. Hershey, PA: IGI Global. doi:10.4018/978-1-5225-5969-6.ch002
- Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Computer Science Education*, 13(4), pp.1-12. <http://doi.org/10.1076/csed.13.4.249.17496>.
- Kerlinger, F.N. (1979). *Behavioral research: A conceptual approach*. NewYork:Holt, Rinehart, and Winston.
- Krosnick, J. a. (1999). Survey research. *Annual Review of Psychology*, 50, pp. 537-567. <http://doi.org/10.1146/annurev.psych.50.1.537>
- Land, R., Cousin, G., Meyer, J. H. F., and Davies, P. (2005). Threshold concepts and troublesome knowledge (3): implications for course design and evaluation. *Improving Student Learning Diversity and Inclusivity*, 49(3), 53-64. Retrieved from http://owwww.brookes.ac.uk/services/ocslid/isl/isl2004/abstracts/conceptual_papers/ISL_04-pp53-64-Land-et-al.pdf.
- Law, K. M. Y., Lee, V. C. S., and Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers and Education*, 55(1), pp. 218-228. <http://doi.org/10.1016/j.compedu.2010.01.007>.
- Le May, M. R., So, D. Y., Dionne, R., Glover, C. A., Froeschl, M. P., Wells, G. A., ... and O'Brien, E. R. (2008). A citywide protocol for primary PCI in ST-segment elevation

- myocardial infarction. *New England Journal of Medicine*, 358(3), pp. 231-240.
- Likert, R. (1932). A technique for measurement of attitudes. *Archives of Psychology*.
- Lune, H., and Berg, B. L. (2016). *Qualitative research methods for the social sciences*: Pearson Higher Ed.
- Ma, W., Adesope, O. O., Nesbit, J. C., and Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of educational psychology*, 106(4), 901.
- Martín-Blas, T., and Serrano-Fernández, A. (2009). The role of new technologies in the learning process: Moodle as a teaching tool in Physics. *Computers and Education*, 52(1), pp. 35-44.
- Martin, P. Y., and Turner, B. A. (1986). Grounded theory and organizational research. *The journal of applied behavioral science*, 22(2), pp. 141-157.
- Marshall, C., and Rossman, G. B. (2014). *Designing qualitative research*: Sage publications.
- Masadeh, M. a. (2012). Focus Group: Reviews and Practices. *International Journal of Applied Science and Technology*, 2(10), pp. 63-68. Retrieved from http://www.ijastnet.com/journals/Vol_2_No_10_December_2012/9.pdf.
- Mayer, R. E. (2017). Using multimedia for e-learning. *Journal of Computer Assisted Learning*, 33(5), pp. 403-423.
- Mayer, R. E. (2013). *Teaching and learning computer programming: Multiple research perspectives*. Routledge.
- Mather, R. (2015). Multivariate Gradient Analysis for Evaluating and Visualizing a Learning System Platform for Computer Programming. *Journal, The lafor iii, Education Volume, III(I)*, pp.17-30.
- McCartney, R., Eckerdal, A., Mostrom, J. E., Sanders, K., and Zander, C. (2007). Successful students' strategies for getting unstuck. *ACM SIGCSE Bulletin*, 39(3), pp. 156. <http://doi.org/10.1145/1269900.1268831>.
- Meyer, J. L., Durairaj, S., and Hakhinian, M. (2016). U.S. Patent No. 9,251,360. Washington, DC: U.S. Patent and Trademark Office.
- Meyer, J., and Land, R. (2003). Threshold concepts and Troublesome knowledge: linkages to ways of thinking and practising within the disciplines.
- Meyer, B. (1993). Logic and the Structure of Space-Towards a Visual Logic for Spatial Reasoning. Paper presented at the ILPS.
- Meyerovich, L. A., and Rabkin, A. (2013). *Empirical Analysis of Programming Language Adoption*.
- Mertens, T., Kautz, J., and Van Reeth, F. (2009, March). Exposure fusion: A simple and practical alternative to high dynamic range photography. In *Computer graphics forum* (Vol. 28, No. 1, pp. 161-171). Oxford, UK: Blackwell Publishing Ltd.
- Milne, I., and Rowe, G. (2002). Difficulties in Learning and Teaching Programming – Views of Students and Tutors. *Education and Information Technologies*, 7, 55-66. <http://doi.org/10.1023/A:1015362608943>
- Mohorovičić, S., and Strčić, V. (2011). An Overview of Computer Programming Teaching Methods. *Central European Conference on Information* Retrieved from http://www.cecis.foi.hr/app/public/conferences/1/archive2011/EIS_3.pdf.
- Moreno, A., Myller, N., and Bednarik, R. (2005). Jeliot 3, an extensible tool for program visualization.
- Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '04*, 373. <http://doi.org/10.1145/989863.989928>
- Moons, J., and De Backer, C. (2009). Rationale Behind the Design of the EduVisor Software Visualization Component. *Electronic Notes in Theoretical Computer Science*, 224(C), 57-65. <http://doi.org/10.1016/j.entcs.2008.12.049>
- Moström, J. E., Boustedt, J., Eckerdal, A., McCartney, R., Sanders, K., Thomas, L., and

- Zander, C. (2008). Concrete Examples of Abstraction as Manifested in Students' Transformative Experiences. *Science Education*, 125-135. <http://doi.org/10.1145/1404520.1404533>
- Moström, J. E., Boustedt, J., Eckerdal, A., McCartney, R., Sanders, K., Thomas, L., and Zander, C. (2009). Computer science student transformations. *ACM SIGCSE Bulletin*, 41(3), 181. <http://doi.org/10.1145/1595496.1562935>
- Moussa, W. E., Almalki, R. M., Alamoudi, M. A., and Allinjawi, A. (2016). Proposing a 3d interactive visualization tool for learning oop concepts. 2016 13th Learning and Technology Conference, L and T 2016, 26-32. <http://doi.org/10.1109/LT.2016.7562861>
- Mouzelis, N. P. (2016). *Back to sociological theory: the construction of social orders*. Springer.
- Mselle, L. J. (1989). Enhancing Comprehension by Using Random Access Memory (RAM) Diagrams in Teaching Programming : Class Experiment. *Science And Technology*.
- Mselle, L. J., and Twaakyondo, H. (2012). The impact of Memory Transfer Language (MTL) on reducing misconceptions in teaching programming to novices. *International Journal of Machine Learning and Applications*, 1, 1-6. <http://doi.org/10.4102/ijmla.v1i1.3>
- Murphy, C., Kim, E., Kaiser, G., and Cannon, A. (2008). Backstop: a tool for debugging runtime errors. *ACM SIGCSE Bulletin*, 40(1), pp.173. <http://doi.org/10.1145/1352322.1352193>.
- Noble, H., and Mitchell, G. (2016). What is grounded theory? *Evidence-based nursing*, 19(2), pp. 34-35.
- Nyamawe, A. S. (2014). A Proposed Framework for Development of a Visualizer Based on Memory Transfer Language (MTL). arXiv preprint arXiv:1408.2564.
- Padilla, S. A., Straus, J., Leidich, J., and Hahn, N. (2015). StratEx Mission Overview.
- Parker, K. R., Ottaway, T. A., and Chao, J. T. (2014). Criteria for the selection of a programming language for introductory courses, (May). <http://doi.org/10.1504/IJKL.2006.009683>.
- Phillips, D. C. and N. C. Burbules. (2000). *Postpositivism and educational research*. Lanham, MA: Rowman and Littlefield.
- Pinsonneault, A., and Kraemer, K. L. (1993). Survey Research Methodology in Management Information Systems: An Assessment. *Journal of Management Information System*, 10(2), 75-105. <http://doi.org/10.1016/j.breastdis.2014.01.018>.
- Priest, H., Roberts, P., and Woods, L. (2002). An overview of three different approaches to the interpretation of qualitative data. Part 1: Theoretical issues. *Nurse Researcher*, 10(1), pp. 30.
- Resnik, D. B., Elliott, K. C., and Miller, A. K. (2015). A framework for addressing ethical issues in citizen science. *Environmental Science and Policy*, 54, 475-481.
- Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., ... Silver, J. (2009). Scratch: Programming for All. *Communications of the ACM*, 52(11), 60. <http://doi.org/10.1145/1592761.1592779>.
- Richard, T. (2013). Qualitative versus quantitative methods: Understanding why qualitative methods are superior for criminology and criminal justice.
- Robins, A., Rountree, J. and Rountree, N. (2003). A review and discussion. *Computer Science Education*. <http://doi.org/10.1076/csed.13.2.137.14200>
- Rößling, G. (2010). A family of tools for supporting the learning of programming. *Algorithms*, 3, 168-182. <http://doi.org/10.3390/a3020168>.
- Rößling, G., Schüer, M., and Freisleben, B. (2000). The ANIMAL algorithm animation tool. Paper presented at the ACM SIGCSE Bulletin.
- Rodrigues, H., Almeida, F., Figueiredo, V. and Lopes, S., (2019). Tracking e-learning through published papers: A systematic review. *Computers and Education*, 136, pp.87-98.
- Rountree, J., and Rountree, N. (2009). Issues regarding threshold concepts in computer

- science. *Conferences in Research and Practice in Information Technology Series*, 95, pp. 139-145.
- Salakoski, T. (2006). Koli Calling 2005 Conference on Computer Science Education. on Computing Education Research.(Koli, Finland, 2005 ..., (November). Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.9111andrep=rep1andtype=pdf#page=10%5Cnhttp://scholar.google.com/scholar?hl=enandbtnG=Searchandq=intitle:Koli+Calling+2005+Conference+on+Computer+Science+Education#2>
- Salcedo, S. L., and Idrobo, A. M. O. (2011). New tools and methodologies for programming languages learning using the scribbler robot and Alice. *Proceedings - Frontiers in Education Conference, FIE*, pp. 1-6. <http://doi.org/10.1109/FIE.2011.6142923>
- Saldaña, J. (2016). The coding manual for qualitative researchers. *The Coding Manual for Qualitative Researchers*, 339. <http://doi.org/10.1109/TEST.2002.1041893>
- Samy, M., and Robertson, F. (2017). From positivism to social constructivism: an emerging trend for CSR researchers. In *Handbook of Research Methods in Corporate Social Responsibility*. Edward Elgar Publishing.
- Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Thomas, L., and Zander, C. (2008). Student understanding of object-oriented programming as expressed in concept maps. *ACM SIGCSE Bulletin*, 40(1),pp.332. <http://doi.org/10.1145/1352322.1352251>.
- Sanders, K., Boustedt, J., Eckerdal, A., Moström, J. E., Mccartney, R., Thomas, L., and Zander, C. (2012). Threshold Concepts and Threshold Skills in Computing Categories and Subject Descriptors. *Icer'12*, 23-30. <http://doi.org/10.1145/2361276.2361283>
- Sanders, K., and Mccartney, R. (2016). Threshold Concepts in Computing: Past , Present , and Future. In the 16th Koli Calling International Conference ,pp. 91-100.
- Satratzemi, M., Dagdilelis, V., and Evagelidis, G. (2001). A system for program visualization and problem-solving path assessment of novice programmers. *ACM SIGCSE Bulletin*, 33(3), pp.137-140. <http://doi.org/10.1145/507758.377667>.
- Schroeder, A., Minocha, S., and Schneider, C. (2010). The strengths, weaknesses, opportunities and threats of using social software in higher and further education teaching and learning. *Journal of Computer Assisted Learning*, 26(3), pp.159-174.
- Shatri, K., and Buza, K. (2017). The Use of Visualization in Teaching and Learning Process for Developing Critical Thinking of Students. *European Journal of Social Science Education and Research*, 4(1), 71-74.
- Shinners-Kennedy, D., and Fincher, S. a. (2013). Identifying threshold concepts. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research ICER*, pp.9. <http://doi.org/10.1145/2493394.2493396>.
- Shinners-Kennedy, D. (2016). How NOT to identify threshold concepts. In *Threshold concepts in practice* ,pp. 253-267. SensePublishers, Rotterdam.
- Siti Rosminah, M. D., and Ahmad Zamzuri, M. A. (2012). Difficulties in learning programming: Views of students. *1st International Conference on Current Issues in Education*, pp. 74-79. <http://doi.org/10.13140/2.1.1055.7441>
- Skinner, B. F. (1958). Teaching machines. *Science*, 128(3330), pp. 969-977.
- Sorva, J. (2010). Reflections on threshold concepts in computer programming and beyond. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10*, 21-30. <http://doi.org/10.1145/1930464.1930467>
- STRATEX.pdf. (1997).
- Stankov, S., Glavinic, V., and Rosie, M. (2011). Design, Implementation and Evaluation.
- Starks, H., and Trinidad, S. B. (2007). Choose your method: A comparison of phenomenology, discourse analysis, and grounded theory. *Qualitative Health Research*, 17(10), pp. 1372- 1380.
- Strauss, A., and Corbin, J. (1994). Grounded theory methodology. In N. Denzin and Y. Lincoln (Eds.), *Handbook of qualitative research*,pp. 273-285. Thousand Oaks, CA: Sage.

- Su, J. M., and Hsu, F. Y. (2017). Building a Visualized Learning Tool to Facilitate the Concept Learning of Object-Oriented Programming. *Proceedings - 2017 6th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2017*, pp. 516-520. <http://doi.org/10.1109/IIAI-AAI.2017.180>.
- Suddaby, R. (2006). From the editors: What grounded theory is not. *Academy of Management Journal ARCHIVE*, 49(4),pp. 633-642.
- Sun, B. (2010). Java teaching based on BlueJ platform. *2nd International Conference on Information Engineering and Computer Science - Proceedings, ICIECS 2010*, 2-5. <http://doi.org/10.1109/ICIECS.2010.5677726>.
- Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research*, 36(2),pp. 223-244.
- Tadajewski, M. (2004). The philosophy of marketing theory: Historical and future directions. *The Marketing Review*, 4(3), pp. 307-340.
- Taherdoost, T., (2019) .What Is the Best Response Scale for Survey and Questionnaire Design; Review of Different Lengths of Rating Scale / Attitude, Scale / Likert Scale, *International Journal of Academic Research in Management*, Vol. 8, No. 1, 2019, pp. 1-10.
- Tamim, R. M., Lowerison, G., Schmid, R. F., Bernard, R. M., and Abrami, P. C. (2011). A multi-year investigation of the relationship between pedagogy, computer use and course effectiveness in postsecondary education. *Journal of Computing in Higher Education*, 23(1), pp.1-14.
- Vagianou, E. (2006). Program Working Storage: A Beginner ' s Model, 69-76.
- Vall Castelló, B. (2016). Bridging constructivism and social constructionism: The journey from narrative to dialogical approaches and towards synchrony. *Journal of Psychotherapy Integration*, 26(2), pp. 129.
- Valverde-Berrocoso, J., Garrido-Arroyo, M., Burgos-Videla, C. and Morales-Cevallos, M., (2020). Trends in Educational Research about e-Learning: A Systematic Literature Review (2009-2018). *Sustainability*, 12(12), pp.5153.
- Virtanen, A. T., Lahtinen, E., and Jarvinen, H.-M. (2005). VIP, a Visual Interpreter for Learning Introductory Programming with C++. *Koli Calling '05*.
- Welsh, E. T., Wanberg, C. R., Brown, K. G., and Simmering, M. J. (2003). E-learning: Emerging uses, emirical results and future directions. *International Journal of Training and Development*, 7(4),pp. 245-258. <http://doi.org/10.1046/j.1360-3736.2003.00184.x>
- Williams, A., and Katz, L. (2001). The use of focus group methodology in education: Some theoretical and practical considerations. *International Journal for Leadership in Learning*, 5(3).
- Yang, F.-J. (2010). The ideology of intelligent tutoring systems. *ACM Inroads*. <http://doi.org/10.1145/1869746.1869765>
- Yang, J., Lee, Y., and Chang, K. H. (2018). The Journal of Systems and Software Evaluations of JaguarCode: A web-based object-oriented programming environment with static and dynamic visualization. *The Journal of Systems and Software*, 145(December 2017), pp.147-163. <http://doi.org/10.1016/j.jss.2018.07.037>.
- Yeomans, L., Zschaler, S., and Coate, K. (2019). Transformative and Troublesome? Students' and Professional Programmers' Perspectives on Difficult Concepts in Programming. *ACM Transactions on Computing Education*, 19(3), pp.1-27. <http://doi.org/10.1145/3283071>
- Young, R. A., and Collin, A. (2004). Introduction: Constructivism and social constructionism in the career field. *Journal of vocational behavior*, 64(3),pp.373-388.
- Zander Carol Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., and Sanders, K. (2008). Threshold Concepts In Computer Science: A Multi-National

- Empirical Investigation. Threshold Concepts within the Disciplines, pp.105-118.
- Zelhart, F., and Wallingford, E. (1994). A survey of intelligent tutoring systems and the methods used for effective tutoring. Intelligent Systems Laboratory, pp.1-35. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.1280&rep=rep1&type=pdf>
- Zhang, Y., Surisetty, S., and Scaffidi, C. (2013). Assisting comprehension of animation programs through interactive code visualization. *Journal of Visual Languages and Computing*, 24(5), pp.313-326. <http://doi.org/10.1016/j.jvlc.2013.07.001>.
- Zheng, D., Shi, M., Wang, Y., Eseye, A., and Zhang, J. (2017). Day-ahead wind power forecasting using a two-stage hybrid modeling approach based on scada and meteorological information, and evaluating the impact of input-data dependency on forecasting accuracy. *Energies*, pp. 10(12).

Appendices

Appendix A- Ethical approval letter and form- Consent form- Information sheet (Data collection)

Ethical approval

**RESEARCH
WITH
PLYMOUTH
UNIVERSITY**

3 September 2015

CONFIDENTIAL

Sarah Alhammad
School of Computing, Electronics and Mathematics

Dear Sarah

Ethical Approval Application

Thank you for submitting the ethical approval form and details concerning your project:

Tracing Learning Environment in JAVA Programming Language

I am pleased to inform you that this has been approved.

Kind regards



Paula Simson
Secretary to Faculty Research Ethics Committee

Cc. Dr Shirley Atkinson

PLYMOUTH UNIVERSITY FACULTY OF SCIENCE AND ENGINEERING

Research Ethics Committee

APPLICATION FOR ETHICAL APPROVAL OF RESEARCH INVOLVING
HUMAN PARTICIPANTS

All applicants should read the guidelines which are available via the following link:
<https://staff.plymouth.ac.uk/scienv/humanethics/intranet.htm>

This is a WORD document. Please complete in WORD and extend space where necessary.
*All applications must be word processed. Handwritten applications **will** be returned.*

Please submit with interview schedules and/or questionnaires appropriately.

Postgraduate and Staff must submit a signed copy to SciEngHumanEthics@plymouth.ac.uk

Undergraduate students should contact their School Representative of the Science and Engineering Research Ethics Committee or dissertation advisor prior to completing this form to confirm the process within their School.

School of Computing, Electronics and Mathematics undergraduate students – please submit to SciEngHumanEthics@plymouth.ac.uk with your project supervisor copied in.

1. TYPE OF PROJECT

1.1 What is the type of project? (Put an X next to one only)

STAFF should put an X next to one of the three options below:

Specific project

Thematic programme of research

Practical / Laboratory Class

1.2 Put an X next to one only

POSTGRADUATE STUDENTS should put an X next to one of the options below:

Taught Masters Project

M.Phil / PhD by research

UNDERGRADUATE STUDENTS should put an X next to one of the options below:

Student research project

Practical / Laboratory class where you are acting as the experimenter

2. APPLICATION

2.1 TITLE of Research project
Tracing Learning Environment in JAVA programming Language

2.2 General summary of the proposed research for which ethical clearance is sought, briefly outlining the aims and objectives and providing details of interventions/procedures involving participants (no jargon)	
<p>A plenty of visualization software have emerged recently with the aim of support the learning how to program for novices programmer. Each tool has its features that may or may not be useful for better understanding. The software that considered in this research is the one that based on using memory-referencing and visualizing what happen in the individual execution for program statement. Clearly, the effectiveness of the existing educational software is an important part of collecting students requirements and needs for further improvements.</p> <p>The objectives of the research is to explore the problem(s) do students have with memory-reference tools and what are the technical hurdles remain to students being able to learn programming using the existing tools. The research investigates on the program visualization tools and compares them for the purpose of collecting user requirements needed to implement the Ph.D. framework.</p> <p>The evaluation will incorporate the semi-structured interviews method that the students will be asked open-ended questions based on issues of interest.</p> <p>In detail, the interviews will cover the following aspects:</p> <ul style="list-style-type: none"> ▪ The students will be asked about their experience and background in using visualization tools for the purpose of learning programming. ▪ Three tools will be presented to the students as a sample of using visualization in the programming learning, and then the students will be discussed the tools, and its usability determining the strengthen and weakness on them ▪ Students will be asked to compare the tools among each other on solving some programming problems and requesting their opinion on the best/worst tool. This will help to identify the areas where difficulties understanding and the weakness area of the tools. <p>Interviewers will request the student's permission to record these sessions so that the process is not interrupted by note taking.</p>	
2.3 Physical site(s) where research will be carried out	
Plymouth University, Drake Circus, Plymouth, PL4 8AA Princess Norah Bint Abdurrahman University (PNU) , Riyadh, Saudi Arabia	
2.4 External Institutions involved in the research (e.g. other university, hospital, prison etc.)	
Our research may include participants form other universities in Saudi Arabia .	
2.5 Name, telephone number, e-mail address and position of lead person for this project (plus full details of Project Supervisor if applicable)	
PhD Student: Sarah Alhammad: +966554427159 : sarah.alhammad@plymouth.ac.uk PhD Supervisor: Dr. Shirley Atkinson : 01752 586209 : shirley.atkinson@plymouth.ac.uk	
2.6 Start and end date for research for which ethical clearance is sought (NB maximum period is 3 years)	
Start date:01-10-2015	End date: 01-10-2018
2.7 Has this same project received ethical approval from another Ethics Committee?	
Delete as applicable:	No <input checked="" type="checkbox"/> Yes <input type="checkbox"/>
2.8 If yes, do you want Chairman's action?	

Delete as applicable:	No	Yes
<i>If yes, please include other application and approval letter and STOP HERE. If no, please continue</i>		

3. PROCEDURE

3.1 Describe procedures that participants will engage in, Please do not use jargon

The proposed evaluation procedure has four steps. Each is described in detail:

Step 1: The students will be welcomed as they arrive (5 minutes)

- The interviewer will ask the students if they are willing to participate in the session.
- If they are, then the interviewer will ask the users permission to audio record the session.
- If they are still willing to participate then the interviewer will explain all about withdrawal - the fact that they can withdraw at any time and they can also subsequently, withdraw their data from the study and how to do so.
- The interviewer will also explain that all personally identifiable information will be held separate from the core dataset. It will only be used if the student wishes to withdraw their consent.
- Once the student has been briefed and if they are willing to participate the interviewer will ask them to sign the consent document before proceeding.

The interviewer will provide the user with an overview of the session activities and duration. This will give a breakdown of how long each step should take. This will ensure that the student is fully informed about the process.

Step 2 – student is given an introduction (5 minutes)

- Initially, the student will be introduced to the subject in general and the objectives of the study. This will give the students the background they need to do the tasks.

Step 3 – User attempts to do task 1, 2 and 3 (45 minutes for all)

- The tasks are not specified in exact detail here, as they will vary slightly as the research evolves. However, they will all follow the same basic outline:
 - First: The students background on using any tool for the purpose of tracing and observing the program behaviour during their programming study. The discussion expanded to their experience and whether they benefit from them. Moreover, the features and drawbacks existing on the tools from the students perspective. This session will be designed so that it lasts no more than 10 minutes.
 - Second: The second part of the discussion is evaluating of three selected tools. The tools have a common methodology that is the program visualization method. However, each tool has its features.

The evaluation is done by presenting three activities that usually students stuck on. This session will be designed so that it lasts no more than 25 minutes.

- Third: comparison discussion is the third aspect. The comparison between the tools on solving the three kinds of activities. This session will be designed so that it lasts no more than 10 minutes.

Step 4 – Final feedback and thanks (5 minutes)

- After all the tasks are completed, the student will be asked to comment more generally about the tools. For example, they will be asked their opinion on the method that these tools follow and whether they find it useful and effective.
- Since we want our students to be happy with this process, we plan to reaffirm their

rights to withdraw their data.

- It is also important that we give students the right to ask questions about the research and the process. We want the students to feel valued as an important part of this process.
- Finally, we will thank the subjects (handing over the thank you card with their code on it) and ask them if they would be willing to participate again at a later stage.

Ethical approval for recording of these sessions is specifically sought so that note taking does not interfere with the process. In particular, the discursive process when users are describing the problems they see/have experienced when using the tools.

3.2 How long will the procedures take? Give details

All evaluation sessions will last approximately 55-60 minutes.

- Step 1 – As the student arrives they will be welcomed (5 minutes). This includes an overview of sessions and its duration.
- Step 2 – Interviewer will give an introduction (5 minutes). This includes an overview of the subject and the study objectives
- Step 3– students attempt to do task 1, 2 and 3 (45 minutes divided over three interlocutors, 10 min, 25 min, 10 min)
- Step 4 – Final feedback and thanks (5 minutes)

3.3 Does your research involve deception?

Delete as applicable:

No

Yes

3.4 If yes, please explain why the following conditions apply to your research:

a) Deception is completely unavoidable if the purpose of the research is to be met

b) The research objective has strong scientific merit

c) Any potential harm arising from the proposed deception can be effectively neutralised or reversed by the proposed debriefing procedures (see section below)

3.5 Describe how you will debrief your participants

The participants will be debriefed as follows

- The students will be thanked for their participation.
- Then, to maintain anonymity, each student will be assigned their own “participants code”.
- They will be issued with this “participants code” in writing and reminded of the process/contact details for withdrawing their consent/data from the study. Specifically
 - Participants can withdraw their consent/data from this study, up to 6 weeks after the date of the usability session took place
 - To withdraw, participants simply contact Mrs Sarah Alhammad stating their unique “participants code”. Contact details include postal as well as electronic contact information.
 - Subsequently, the Participants data will be permanently deleted, and the participant will receive a confirmation email/letter (as appropriate) to that effect. It is important that participants who withdraw are kept well informed and are reassured that their data has been permanently deleted as

requested.

3.6 Are there any ethical issues (e.g. sensitive material)?

Delete as applicable:

No

Yes

3.7 If yes, please explain. You may be asked to provide ethically sensitive material. See also section 11

4. BREAKDOWN OF PARTICIPANTS

4.1 Summary of participants

Type of participant	Number of participants
Non-vulnerable Adults	20
Minors (< 16 years)	0
Minors (16-18 years)	0
Vulnerable Participants (other than by virtue of being a minor)	0
Other (please specify)	0
TOTAL	20

4.2 How were the sample sizes determined?
It is estimated that the average number for effective semi-structured interviews is about 15 to 20 participants. Participants will be recruited using email as well as personal contact.
4.3 How will subjects be recruited?
Participants will be recruited using email as well as personal contact.
4.4 Will subjects be financially rewarded? If yes, please give details.
Yes, subjects will be paid £8 per hour. This rate is based on the current rates used by Plymouth University for similar experiments. The students in PNU will volunteer to participate in the study.

5. NON-VULNERABLE ADULTS

5.1 Are some or all of the participants non-vulnerable adults?
Delete as applicable: No <input type="checkbox"/> Yes <input checked="" type="checkbox"/>
5.2 Inclusion / exclusion criteria
There are no specific inclusion/exclusion criteria
5.3 How will participants give informed consent?
At the beginning, embedded in Step 1 consent is sought for both participation and audio recording individually. Again, embedded in the final step, the process for consent withdrawal is reiterated and the subject is given a written "thank card" which describes how and the time limitations on the withdrawal of consent.

As mentioned in section 3.1 – repeated here for clarity

Step 1: The students will be welcomed as they arrive (5 minutes)

- The interviewer will ask the students if they are willing to participate in the session.
- If they are, then the interviewer will ask the users permission to audio record the session.
- If they are still willing to participate then the interviewer will explain all about withdrawal - the fact that they can withdraw at any time and they can also subsequently, withdraw their data from the study and how to do so.
- The interviewer will also explain that all personally identifiable information will be held separate from the core dataset. It will only be used if the student wishes to withdraw their consent.
- Once the student has been briefed and if they are willing to participate the interviewer will ask them to sign the consent document before proceeding.

The interviewer will provide the user with an overview of the session activities and duration. This will give a breakdown of how long each step should take. This will ensure that the student is fully informed about the process.

Step 4 – Final feedback and thanks (5 minutes)

- After all the tasks are completed, the student will be asked to comment more generally about the tools. For example, they will be asked their opinion on the method that these tools follow and whether they find it useful and effective.
- Since we want our students to be happy with this process, we plan to reaffirm their rights to withdraw their data.
- It is also important that we give students the right to ask questions about the research and the process. We want the students to feel valued as an important part of this process.
- Finally, we will thank the subjects (handing over the thank you card with their code on it) and ask them if they would be willing to participate again at a later stage.

Ethical approval for recording of these sessions is specifically sought so that note taking does not interfere with the process. In particular, the discursive process when users are describing the problems they see/have experienced when using the tools.

5.4 Consent form(s) attached

Delete as applicable: No Yes

If no, why not?

5.5 Information sheet(s) attached

Delete as applicable: No Yes

If no, why not?

5.6 How will participants be made aware of their right to withdraw at any time?

Ensuring that participants clearly understand their right to withdraw is embedded in the study. During Step 1, the participator is welcomed. After the welcome, withdrawal of participation/data and permission to audio record is discussed in detail and written consent is sought before continuing the interview.

Delete as applicable:	No	Yes
<i>If no, why not?</i>		
6.9 How will minors be made aware of their right to withdraw at any time?		
6.10 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?		

7. MINORS 16-18 YEARS OLD

7.1 Are some or all of the participants between the ages of 16 and 18?		
Delete as applicable:	No <input checked="" type="checkbox"/>	Yes <input type="checkbox"/>
<i>If yes, please consult special guidelines for working with minors. If no, please continue.</i>		
7.2 Inclusion / exclusion criteria		
7.3 How will minors give informed consent? (See guidelines)		
7.4 Consent form(s) for minor attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
7.5 Information sheet(s) for minor attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
7.6 Consent form(s) for parent / legal guardian attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
7.7 Information sheet(s) for parent / legal guardian attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
7.8 How will minors be made aware of their right to withdraw at any time?		
7.9 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?		

8. VULNERABLE GROUPS

8.1 Are some or all of the participants vulnerable? (See guidelines)

Delete as applicable:	No <input checked="" type="checkbox"/>	Yes <input type="checkbox"/>
<i>If yes, please consult special guidelines for working with vulnerable groups. If no, please continue.</i>		
8.2 Describe vulnerability (apart from possibly being a minor)		
8.3 Inclusion / exclusion criteria		
8.4 How will participants give informed consent?		
8.5 Consent form(s) for vulnerable person attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
8.6 Information sheet(s) for vulnerable person attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
8.7 Consent form(s) for parent / legal guardian attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
8.8 Information sheet(s) for parent / legal guardian attached		
Delete as applicable:	No	Yes
<i>If no, why not?</i>		
8.9 How will participants be made aware of their right to withdraw at any time?		
8.10 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?		

9. EXTERNAL CLEARANCES

Investigators working with children and vulnerable adults legally require clearance from the Disclosure and Barring Service (DBS)

9.1 Do ALL experimenters in contact with children and vulnerable adults have <u>current</u> DBS clearance? Please include photocopies.			
Delete as applicable:	No <input type="checkbox"/>	Yes <input type="checkbox"/>	N/A <input checked="" type="checkbox"/>
<i>If no, explain</i>			
9.2 If your research involves external institutions (school, social service, prison, hospital etc) please provide cover letter(s) from institutional heads permitting you to carry out research on their clients, and where applicable, on their site(s). Are these included?			

Delete as applicable:	No	Yes	N/A
<i>If not, why not?</i>			

10. PHYSICAL RISK ASSESSMENT

<i>10.1 Will participants be at risk of physical harm (e.g. from electrodes, other equipment)? (See guidelines)</i>			
Delete as applicable:	No	<input checked="" type="checkbox"/>	Yes <input type="checkbox"/>
<i>10.2 If yes, please describe</i>			
<i>10.3 What measures have been taken to minimise risk? Include risk assessment proformas which has been signed by the Head of Department</i>			
The session will be supervised at all times and equipment that is being used has been tested for electrical safety. No procedures or content will be used that might be expected to cause physical or psychological harm.			
<i>10.4 How will you handle participants who appear to have been harmed?</i>			

11. PSYCHOLOGICAL RISK ASSESSMENT

<i>11.1 Will participants be at risk of psychological harm (e.g. viewing explicit or emotionally sensitive material, being stressed, recounting traumatic events)? (See guidelines)</i>			
Delete as applicable:	No	<input checked="" type="checkbox"/>	Yes <input type="checkbox"/>
<i>11.2 If yes, please describe</i>			
<i>11.3 What measures have been taken to minimise risk?</i>			
The session will not expose the user to any psychological risks.			
<i>11.4 How will you handle participants who appear to have been harmed?</i>			

12. RESEARCH OVER THE INTERNET

<i>12.1 Will research be carried out over the internet?</i>			
Delete as applicable:	No	<input checked="" type="checkbox"/>	Yes <input type="checkbox"/>
<i>12.2 If yes, please explain protocol in detail, explaining how informed consent will be given, right to withdraw maintained, and confidentiality maintained. Give details of how you will guard against abuse by participants or others (see guidelines)</i>			

13. CONFLICTS OF INTEREST and THIRD PARTY INTERESTS

13.1 Do any of the experimenters have a conflict of interest? (See guidelines)	
Delete as applicable:	No <input checked="" type="checkbox"/> Yes <input type="checkbox"/>
13.2 If yes, please describe	
13.3 Are there any third parties involved? (See guidelines)	
Delete as applicable:	No <input checked="" type="checkbox"/> Yes <input type="checkbox"/>
13.4 If yes, please describe	
13.5 Do any of the third parties have a conflict of interest?	
Delete as applicable:	No <input checked="" type="checkbox"/> Yes <input type="checkbox"/>
13.6 If yes, please describe	

14. ADDITIONAL INFORMATION

14.1 [Optional] Give details of any professional bodies whose ethical policies apply to this research
14.2 [Optional] Please give any additional information that you wish to be considered in this application

15. ETHICAL PROTOCOL and DECLARATION

To the best of our knowledge and belief, this research conforms to the ethical principles laid down by the University of Plymouth and by any professional body specified in section 14 above.

This research conforms to the University's Ethical Principles for Research Involving Human Participants with regard to openness and honesty, protection from harm, right to withdraw, debriefing, confidentiality, and informed consent

Sign below where appropriate:

STAFF / RESEARCH POSTGRADUATES

	Print Name	Signature	Date
Principal Investigator:	Sarah Alhammad	_____	12-Aug-2015
Other researchers:	Dr. Shirley Atkinson	_____	12-Aug-2015

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND ENGINEERING
Human Ethics Committee Sample Consent Form

Project :visualization tools evaluation
Consent Form

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language

Brief statement of purpose of work

The purpose of this research is to assess the effectiveness of using visualization tools in programming learning and collecting the students' needs and the kind of improvements that could be implemented in future work.

The objectives of this research have been explained to me.

I understand that I am free to withdraw from the research at any stage, and ask for my data to be destroyed if I wish.

I understand that my anonymity is guaranteed, unless I expressly state otherwise.
I gave my permission for audio recording .

I understand that the Principal Investigator of this work will have attempted, as far as possible, to avoid any risks, and that safety and health risks will have been separately assessed by appropriate authorities (e.g. under COSHH regulations)

Under these circumstances, I agree to participate in the research.

Name:

Signature:

Date:

FACULTY OF SCIENCE AND ENGINEERING

Tracing Learning Environment in JAVA programming Language Research Information Sheet

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language: **evaluate visualization tools used in learning programming**

You have been invited to take part in this tool evaluation study that funded by Plymouth University. So, if you are interested in participating in this study, please take some time to read the following information carefully and make sure the study and its procedure are clear before signing the consent form.

Aim of research

A plenty of visualization software have emerged recently with the aim of support the learning how to program for novices programmer. Each tool has its features that may or may not be useful for better understanding. The software that considered in this research is the one that based on using memory-referencing and visualizing what happen in the individual execution for program statement. Clearly, the effectiveness of the existing educational software is an important part of collecting students requirements and needs for further improvements.

The objectives of the research is to explore the problem(s) do students have with memory-reference tools and what are the technical hurdles remain to students being able to learn programming using the existing tools. The research investigates on the program visualization tools and compares them for the purpose of collecting user requirements needed to implement the Ph.D. framework.

Description of the overall procedure

The study will last 60 minutes maximum, so this provided three methods to collect the data, general discussion, presenting the tasks, and collecting feedback through open-ended discussion. Therefore, the session will be audio recording.

An overview of the timetable is given below:

1. Welcome (5 minutes)
interviewer provides the subject with a written overview of the session with timings
2. introduction: introduced to the subject in general and the objectives of the study (5 minutes)
3. Task1 (10 minutes)
4. Task2 (25 minutes)
5. Task3 (10 minutes)
6. Debrief (5 minutes)

Description of risks

There are no procedures or content in this study that could be expected to cause physical or psychological harm.

Benefits of proposed research

This research will provide several important benefits. Firstly, it is an opportunity to know the student's experience in using visualization tools during their programming learning courses. Secondly, to assess the effectiveness of using visualization tools And identify the drawbacks that require further design/development. Finally, it is an opportunity to gather suggestions and opinions about the subject so that new framework can be developed to increase the usefulness of visualization method in the educational field.

Right to withdraw

If you note that to take part in this research, please keep this information sheet. Also, you have the freedom to withdraw your participation/data at any time **up to 6 weeks** after the date of the evaluation session. To withdraw, your participation in this study simply contact Mrs. Sarah Alhammad, stating your unique "participators code" that is this code given on the day.

Confidentially

None of the results reported from the study will include information that allows identification of named individuals. No reference will be made to individual persons or participants, either by name or type of data supplied. However, a coding scheme will be used. Each participator will be issued with a "code" which will be used to store their individual data. This code will be required should the user wish to withdraw their consent/data from the study.

All data will be stored on University equipment. The information will be collected and stored on a University computer. This computer is password protected and exists behind a University firewall. The computer is located in a secure building.

Please don't hesitate to contact me if you now wish to withdraw or if you have any questions about the project. To do this, please contact Mrs. Sarah Alhammad, email: sarah.alhammad@plymouth.ac.uk

Thank you for taking time to read the information sheet.

Date: / /

If you are dissatisfied with the way the research is conducted, please contact the principal investigator in the first instance: telephone number 01752 586209. If you feel the problem has not been resolved please contact the secretary to the Faculty of Science and Technology Human Ethics Committee: Mrs Paula Simson 01752 584503

Appendix B- Questions for Semi-structured interviews - students and novices programmers

Semi-structured interview

List of open-ended questions to evaluate visualization tools used in learning programming

Participant code : _____

Dear participant

Thank you for taking the time to have this discussion. The discussion is evaluation of using visualization tools that assist the leaning of programming and gathering the students opinion and suggestions.

Regards

Sarah Alhammad

Personal information			
Name (optional)			
Age	19-21	22-24	25-27
Gender	<input type="radio"/> Male <input type="radio"/> Female		
Occupation	<input type="radio"/> Undergraduate student <input type="radio"/> Post graduate student without job <input type="radio"/> Post graduate student with job <input type="radio"/> Post graduate student		

TASK1: Background information (10 minutes)	
Programming language(s) you have learned	
When was the last time you wrote a program ? what was the purpose?	Few days - few months – a year – more than year Reason:

<p>What are your experiences (if any) on using visualization tools to improve your understanding to programming? (interviewer should explain the nature of visualization tool)</p> <p>If yes, the following questions will be asked:</p> <ul style="list-style-type: none"> ○ What is the tool? (describe how it works and its method) ○ How often you use it? ○ What are the tool features? ○ What the aspects you like/dislike on the tool? ○ To what extent it was helpful?
<p>What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p>
<p>What technical hurdles remain to you being able to learn programming? Suggestion: lack of (e-learning system, software(s), example databases, program tracing tools)</p>

<p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Name of the tool:</p>
<p>What are the strengthen(s) and weakness(es) of the tool ?</p>
<p>Will you use it if possible ? explain why or why not</p>
<p>Do you think it is beneficial and may improve programming learning?</p>
<p>What other types of improvements should be implemented to build on this tool?</p>

Task3: Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes)

The discussion will be held for each activity on the effectiveness of the tool

Which tool do you prefer for solving the LOOP activity ? Describe why?

Which tool do you prefer for solving the OBJECT activity ? Describe why?

Which tool do you prefer for solving the PASSING PARAMETER activity ? Describe why?

The following is the code for the activities that will be used to evaluate the tool

The code is in JAVA language and C++

	JAVA	C++
LOOPS	<pre> public class CalculateArrayAverageExample { public static void main(String[] args) { int[] numbers = new int[] {10,20,15,25,16,60,100}; int sum = 0; for(int i=0; i < numbers.length ; i++) sum = sum + numbers[i]; double average = sum / numbers.length System.out.println("Average value of array elements is : " + average); } } </pre>	<pre> #include <iostream> using namespace std; #include <iostream> int main() { int n, count; float x, sum, avg; sum = 0; cout << "How many numbers? "; cin >> n; int size= n; int array[size]; for (count=1; count<=n; count++){ cout << "Enter Number: "; cin >> array[n]; } } </pre>

		<pre> sum = sum + array[n]; } cout << "The sum is " << sum << endl; avg = sum / n; cout << "The average is " << avg << endl; system("pause"); return 0; } </pre>
Objects	<pre> public class Polygon { int sides; Polygon() { } Polygon(int s) { sides = s; } public int getSides() { return sides; } } public class Rectangle extends Polygon { int width, height; Rectangle() { super(4); width = 0; height = 0; } Rectangle(int w, int h) { super(4); width = w; height = h; } public int getArea() { return width * height; } } public class Square extends Rectangle { int side; Square() { side = 0; } Square(int s) { </pre>	<pre> #include <iostream> using namespace std; class Rectangle { int width, height; public: void set_values (int,int); int area() {return width*height;} }; void Rectangle::set_values (int x, int y) { width = x; height = y; } int main () { Rectangle rect; rect.set_values (3,4); cout << "area: " << rect.area(); return 0; } </pre>

	<pre> super(s, s); side = s; } } public class MyClass { public static void main() { Square square; square = new Square(3); int area = square.getArea(); System.out.println("The area of the square is " + area); } } </pre>	
<p>Passing parameter</p>	<pre> public class PrimitiveParameter { public void changeValue (int parameter) {parameter = 20;} public static void main (String args[]) { int parameter = 10; PrimitiveParameter test = new PrimitiveParameter(); test.changeValue(parameter); } } </pre>	<pre> class PrimitiveParameter { public: virtual void changeValue(int parameter) { parameter = 20; } static void main(std::wstring args[]) { int parameter = 10; PrimitiveParameter *test = new PrimitiveParameter(); test->changeValue(parameter); } }; </pre>

Appendix C- Visual Code Flow

<http://visualcodeflow.com>

Open in Google Chrome

Appendix D- Ethical approval letter and form- Consent form- Information sheet
(Evaluation of Visualisation)

Ethical approval

**RESEARCH
WITH
PLYMOUTH
UNIVERSITY**

21 November 2017

CONFIDENTIAL

Sarah Alhammad
School of Computing, Electronics and Mathematics

Dear Sarah

Ethical Approval Application

Thank you for submitting the ethical approval form and details concerning your project:

Tracing Learning Environment in JAVA programming Language

I am pleased to inform you this has been approved.

Kind regards



Paula Simson
Secretary to Faculty Research Ethics Committee

Cc. Dr Shirley Atkinson
Dr Liz Stuart

PLYMOUTH UNIVERSITY FACULTY OF SCIENCE AND ENGINEERING

Research Ethics Committee

APPLICATION FOR ETHICAL APPROVAL OF RESEARCH INVOLVING HUMAN PARTICIPANTS

All applicants should read the guidelines which are available via the following link:

<https://staff.plymouth.ac.uk/scienv/humanethics/intranet.htm>

This is a WORD document. Please complete in WORD and extend space where necessary.
All applications must be word processed. Handwritten applications **will** be returned.

Postgraduate and Staff must submit a signed copy to SciEngHumanEthics@plymouth.ac.uk

Undergraduate students should contact their School Representative of the Science and Engineering Research Ethics Committee or dissertation advisor prior to completing this form to confirm the process within their School.

School of Computing, Electronics and Mathematics undergraduate students – please submit to SciEngHumanEthics@plymouth.ac.uk with your project supervisor copied in.

4. TYPE OF PROJECT

1.1 What is the type of project?

Applicant	Type	Put X in 1 only
STAFF	Specific project	
	Thematic programme of research	
	Practical / Laboratory Class	
POSTGRADUATE STUDENTS	Taught Masters Project	
	M.Phil / PhD by research	X
UNDERGRADUATE STUDENTS	Student research project	
	Practical / Laboratory class where you are acting as the experimenter	

5. APPLICATION

2.1 TITLE of Research project
Tracing Learning Environment in JAVA programming Language
2.2 Name, telephone number, e-mail address and position of applicant for this project (plus full details of Project Supervisor for postgraduate and undergraduate students)
Ph.D. Student: Sarah Alhammad: +966554427159 / +447450181279 sarah.alhammad@plymouth.ac.uk
Ph.D. Supervisor: Dr. Shirley Atkinson: 01752 586209 shirley.atkinson@plymouth.ac.uk
2.3 General summary of the proposed research for which ethical clearance is sought, briefly

outlining the aims and objectives (no more than 200 words)	
<p>A plenty of visualization software has emerged recently with the aim of support the learning how to program for novices programmer. Each tool has its features that may or may not be useful for better understanding. The software that considered in this research is the one that based on using memory-referencing and visualizing what happens in the individual execution for program statement. The effectiveness of the existing educational software is an important part of collecting students requirements and needs for further improvements.</p> <p>The purpose of this research is to evaluate the use of visualization in programming learning by measure the students' performance before and after using the method. Furthermore, the research will seek to gathered experts' feedback about the method.</p> <ul style="list-style-type: none"> ➤ The evaluation will incorporate experiment using task that the participants will be asked to solve programming problems and completing surveys regarding their experiment ➤ Expert evaluation <p>Interviews will be conducted for expert participants (computer programming tutors and lecturers)</p>	
2.4 Physical site(s) where research will be carried out	
Princess Norah Bint Abdurrahman University (PNU) , Riyadh, Saudi Arabia	
2.5 Does your research involve external institutions (e.g. other university, hospital, prison etc. see guidelines)	
Delete as applicable: yes	
2.5a If yes, please give details:	
<p>The study will be conducted on the students and faculties of Princess Norah Bint Abdurrahman University (PNU), Riyadh, Saudi Arabia</p> <ul style="list-style-type: none"> ➤ The study will incorporate experiment using tasks that the PNU programming students will be asked to solve programming problems and completing surveys regarding their experiment ➤ The students will be invited to discuss their experiment in focus groups ➤ Interviews will be conducted for expert participants (computer programming tutors and lecturers who work at PNU) 	
2.5b If yes, you must provide letter(s) from institutional heads permitting you to carry out research on their clients, and where applicable, on their sites(s). Are they included?	
Delete as applicable: yes	
If not, why not?	
2.6 Start and end date for research for which ethical clearance is sought (NB maximum period is 3 years)	
Start date: Date of approval	End date: 01-10-2019
2.7 Has this same project received ethical approval from another Ethics Committee?	
Delete as applicable: No	
2.7a If yes, do you want Chair's action?	
Delete as applicable: No Yes	
If yes, please include other application and approval letter and STOP HERE. If no, please continue	

6. PROCEDURE

3.1 Describe (a) the procedures that participants will engage in, and (b) the methods used for data collection and recording

The proposed evaluation procedure will consist of three methodologies (experiment, focus group and interviews)

The participants will engage in the three activities as voluntary, and it is not part of their course, the study will be conducted in their free time.

The researcher will deliver the whole tasks without any involvement from other parties.

The following steps will give details about each methodology and their sessions

Experiment and Focus group :

The evaluation will incorporate experiment using tasks that the participants(students) will be asked to solve programming problems and completing surveys regarding their experiment

The participants will expertise the following aspects:

- The participants will be divided into two groups: the control group and the experiment group
- Both groups will be asked to solve programming problems (pre-test) and answer a giving questionnaire which measures the participants' confidence in solving the problems
- In the second step, the researcher will present the visualization method using the tool (Visual Java Code) to the experiment group and ask them to try it on their computers during the session.

The control group will rely on the lecture notes and lab work to continue their experiment

- In the final step, both groups will be asked to solve a programming problem (post-test) and answer a giving questionnaire which measures the participants' confidence in solving the problems.

The experiment group will be asked to fill a given questionnaire to measure the level of satisfaction on using the tool.

- In a focus group, the experiment group will discuss their experience on using the tool determining the strength and weakness of the visualization method.

The following are the procedure steps for each group:

➤ Experiment group :

The proposed experiment's procedure for the experiment group has five steps including the focus group discussion. Each is described in detail:

Step 1: The participants will be welcomed as they arrive (5 minutes)

- The researcher will ask the participants if they are willing to participate in the session.
- If they are willing to participate then the researcher will explain all about withdrawal - the fact that they can withdraw at any time from the experiment but not the learning course and they can also subsequently, withdraw their data in the survey from the study and how to do so.
- And they can also subsequently, withdraw their data from the study and how to do so.
- The researcher will also explain that all personally identifiable information will be held separate from the core dataset. It will only be used if the participant wishes to withdraw their consent.
- Once the participant has been briefed and if they are willing to participate the

researcher will ask them to sign the consent document before proceeding.

The researcher will provide the participant with an overview of the session activities and duration. The overview will give a breakdown of how long each step should take. It will ensure that the participant is fully informed about the process.

Step 2 – participant is given an introduction (5 minutes)

Initially, the participants will be introduced to the subject in general and the objectives of the study. The introduction will provide the participant the background they need to do the tasks.

Step 3 – participant attempts to do task 1, 2 and 3 (50 minutes for all)

The tasks are not specified in exact detail here, as they will vary slightly as the research evolves. However, they will all follow the same basic outline:

- Task1: The participant will be asked to answer a questionnaire that includes two parts:
 - First part is a questionnaire about their background in the programming field.
 - The second part is a given programming problem(paper-based) that they asked to solve it (pre-test).

After that, the papers will be collected by the researcher. This session will be designed so that it lasts no more than 15 minutes.

- Task 2: The interviewer will present the visualization method, describe the features of the method and how its work.
This session will be designed so that it lasts no more than 20 minutes.
- Task 3: The participant will be asked to answer a questionnaire that includes two parts:
 - First part is a given programming problem(paper-based)that they asked to solve it (post-test).
 - The second part is a questionnaire about their feedback on the tool that has been presented.

After that, the papers will be collected by the researcher. This session will be designed so that it lasts no more than 15 minutes.

Step 4 – Focus group discussion (45 minutes)

- The researcher again will ask the participants if they are willing to participate in the focus group session.
- If they are, then the researcher will ask the participants' permission to audio record the session.
- The researcher will explain all about withdrawal - the fact that they can withdraw at any time from the focus group but not the learning course and they can also subsequently, withdraw their data from the study and how to do so.
- The researcher will ask the participants to discuss their experience on using the tool as group

This session will be designed so that it lasts no more than 45 minutes.

Step 5 – Final feedback and thanks (5 minutes)

- Since we want our participants to be happy with this process, we plan to reaffirm their rights to withdraw their data.
- It is also essential that we give participants the right to ask questions about the research and the process. We want the participants to feel valued as an essential part of this process.
- Finally, we will thank the subjects (handing over the thank you card with their code

on it) and ask them if they would be willing to participate again at a later stage.

Ethical approval for the recording of these sessions is specifically sought so that note-taking does not interfere with the process. In particular, the discursive process when users are describing the problems they see/have experienced when using the tools.

➤ Control group :

The proposed experiment's procedure for the control group has four steps and it will not involve any focus group discussion. Each step is described in detail:

Step 1: The participants will be welcomed as they arrive (5 minutes)

- The researcher will ask the participants if they are willing to participate in the session.
- If they are willing to participate then the researcher will explain all about withdrawal - the fact that they can withdraw at any time from the experiment but not the learning course and they can also subsequently, withdraw their data in the survey from the study and how to do so.
- And they can also subsequently, withdraw their data from the study and how to do so.
- The researcher will also explain that all personally identifiable information will be held separate from the core dataset. It will only be used if the participant wishes to withdraw their consent.
- Once the participant has been briefed and if they are willing to participate the researcher will ask them to sign the consent document before proceeding.

The researcher will provide the participant with an overview of the session activities and duration. The overview will give a breakdown of how long each step should take. It will ensure that the participant is fully informed about the process.

Step 2 – participant is given an introduction (5 minutes)

Initially, the participants will be introduced to the subject in general and the objectives of the study. The introduction will provide the participant the background they need to do the tasks.

Step 3 – participant attempts to do task 1, and 2 (30 minutes for all)

The tasks are not specified in exact detail here, as they will vary slightly as the research evolves. However, they will all follow the same basic outline:

Task1 and Task2 will be conducted in two separate days , Task 1 is the pre-test which should be conducted before computer-lab practice that is part of the course, while Task 2 is the post-test which should be conducted after the students have the computer-lab practice.

- Task1: The participant will be asked to answer a questionnaire that includes two parts:
 - First part is a questionnaire about their background in the programming field.
 - The second part is a given programming problem(paper-based) that they asked to solve it (pre-test).

After that, the papers will be collected by the researcher. This session will be designed so that it lasts no more than 15 minutes.

- Task 2: The participant will be asked to answer a questionnaire that includes programming problem(paper-based)that they asked to solve it (post-test).
After that, the papers will be collected by the researcher. This session will be

designed so that it lasts no more than 15 minutes.

Step 4 – Final feedback and thanks (5 minutes)

- Since we want our participants to be happy with this process, we plan to reaffirm their rights to withdraw their data.
- It is also essential that we give participants the right to ask questions about the research and the process. We want the participants to feel valued as an essential part of this process.
- Finally, we will thank the subjects (handing over the thank you card with their code on it) and ask them if they would be willing to participate again at a later stage.

Interviews:

Interviews will be conducted for expert participants (computer programming tutors and lecturers)

The interview will discuss the following aspects

- The participants will be asked about their experience and background in teaching programming.
- The researcher will present the visualization method using the tool (Visual Java Code) to the participants, and then the participants will give their feedback, comments, and suggestions.

The interviewer will request the participant's permission to record these sessions so that the process is not interrupted by note-taking.

The proposed interviews' procedure has four steps. Each is described in detail:

Step 1: The participants will be welcomed as they arrive (5 minutes)

- The interviewer will ask the participants if they are willing to participate in the session.
- If they are, then the interviewer will ask the users permission to audio record the session.
- If they are still willing to participate then the interviewer will explain all about withdrawal - the fact that they can withdraw at any time and they can also subsequently, withdraw their data from the study and how to do so.
- The interviewer will also explain that all personally identifiable information will be held separate from the core dataset. It will only be used if the participant wishes to withdraw their consent.
- Once the participant has been briefed and if they are willing to participate the interviewer will ask them to sign the consent document before proceeding.

The interviewer will provide the user with an overview of the session activities and duration. The overview will give a breakdown of how long each step should take. It will ensure that the participant is fully informed about the process.

Step 2 – participant is given an introduction (5 minutes)

Initially, the participants will be introduced to the subject in general and the objectives of the study. The introduction will provide the participant the background they need to do the tasks.

Step 3 – User attempts to do three sessions of interview(60 minutes for all)

- First: The participant background on using any tool for teaching programming. The discussion expanded to their experience and whether

they benefit from them. Moreover, the features and drawbacks existing on the tools from the participant's perspective. This session will be designed so that it lasts no more than 10 minutes.

Second: The interviewer will present the visualization method, describe the features of the method and how its work. This session will be designed so that it lasts no more than 20 minutes.

- The third part of the discussion is evaluating of the given method. The participants will be asked about their comments and suggestions. The evaluation is done by presenting three activities that usually students stuck on. This session will be designed so that it lasts no more than 30 minutes.

Step 4 – Final feedback and thanks (5 minutes)

- After all the sessions are completed, the participant will be asked to comment more generally about the tool. For example, they will be asked their opinion on the method and whether they find it useful and practical.
- Since we want our participants to be happy with this process, we plan to reaffirm their rights to withdraw their data.
- It is also essential that we give participants the right to ask questions about the research and the process. We want the participants to feel valued as an essential part of this process.
- Finally, we will thank the subjects (handing over the thank you card with their code on it) and ask them if they would be willing to participate again at a later stage.

Ethical approval for the recording of these sessions is specifically sought so that note-taking does not interfere with the process. In particular, the discursive process when users are describing the problems they see/have experienced when using the tool.

3.1a If surveying or interviewing, you must include your questionnaire(s) and interview schedule(s).

Are these attached:

Delete as applicable:

Yes

3.2 How long will the procedures take? Give details

As mentioned in section 3.1 – repeated here for clarity

The experiment and the focus group for experiment group

All experiment sessions will last approximately 1 hour, 50 minutes including the focus group discussion

The proposed experiment's procedure has five steps including the focus group discussion.

- Step 1 – As the participants arrive, they will be welcomed (5 minutes). This includes an overview of sessions and its duration.
- Step 2 – The researcher will give an introduction (5 minutes). This includes an overview of the subject and the study objectives
- Step 3– students attempt to do task 1, 2 and 3 (50 minutes divided over three interlocutors, 15 min, 20 min, 15 min)
- Step 4 – Focus group discussion (45 minutes)
- Step 5 – Final feedback and thanks (5 minutes)

The survey for control group

Pre-survey and post-survey will last approximately 30 minutes for each survey consists of 4 steps

- Step 1 – As the participants arrive, they will be welcomed (5 minutes). This includes an overview of sessions and its duration.
- Step 2 – The researcher will give an introduction (5 minutes). This includes an overview of the subject and the study objectives

4.2 How were the sample sizes determined?
It is estimated that the average number for effective experiment is about 60 participants divided into two groups of 30 in each, the experimental group will distribute into five groups (5 to 6 participants in each). The experimental group distributed into small groups for better control during the computer-lab activity. Moreover, 5 participants will engage in interviews for expert evaluation
4.3 How will subjects be recruited?
Participants will be recruited using email as well as personal contact.
4.4 Will subjects be financially rewarded? If yes, please give details.
The students and faculties in Princess Nora University will volunteer to participate in the study.

5. NON-VULNERABLE ADULTS

5.1 Are some or all of the participants non-vulnerable adults?
Delete as applicable: Yes
5.2 Inclusion / exclusion criteria
There are no specific inclusion/exclusion criteria
5.3 How will participants give informed consent?
In the beginning, embedded in Step 1 consent is sought for both participation and audio recording individually. Again, embedded in the final step, the process for consent withdrawal is reiterated, and the subject is given a written “thank card” which describes how and the time limitations on the withdrawal of consent. <i>As mentioned in section 3.1 – repeated here for clarity</i> Step 1: The participants will be welcomed as they arrive (5 minutes) <ul style="list-style-type: none"> • The researcher/interviewer will ask the participants if they are willing to participate in the session. • If they are, then the researcher/interviewer will ask the participants permission to audio record the session. • If they are still willing to participate then the researcher/interviewer will explain all about withdrawal - the fact that they can withdraw at any time and they can also subsequently, withdraw their data from the study and how to do so. • The interviewer will also explain that all personally identifiable information will be held separate from the core dataset. It will only be used if the student wishes to withdraw their consent. • Once the student has been briefed and if they are willing to participate the researcher/interviewer will ask them to sign the consent document before proceeding. <p>The researcher/interviewer will provide the user with an <u>overview</u> of the session activities and duration. The overview will give a breakdown of how long each step should take. It will ensure that the participants are fully informed about the process.</p> <p>Step 4 – Final feedback and thanks (5 minutes)</p> <ul style="list-style-type: none"> • After all the tasks are completed, the participants will be asked to comment more generally about the tool. For example, they will be asked their opinion on the

<p>method that the tool follows and whether they find it useful and practical.</p> <ul style="list-style-type: none"> • Since we want our participants to be happy with this process, we plan to reaffirm their rights to withdraw their data. • It is also essential that we give participants the right to ask questions about the research and the process. We want the participants to feel valued as an essential part of this process. • Finally, we will thank the subjects (handing over the thank you card with their code on it) and ask them if they would be willing to participate again at a later stage. <p>Ethical approval for the recording of these sessions is specifically sought so that note-taking does not interfere with the process. In particular, the discursive process when participants are describing the problems they see/have experienced when using the tool.</p>
5.4 Consent form(s) attached
Delete as applicable: Yes
If no, why not?
5.5 Information sheet(s) attached
Delete as applicable: Yes
If no, why not?
5.6 How will participants be made aware of their right to withdraw at any time?
<p>Ensuring that participants understand their right to withdraw is embedded in the study. During Step 1, the participator is welcomed. After the welcome, withdrawal of participation/data and permission to audio record is discussed in detail, and written consent is sought before continuing the session.</p> <p>Towards the end of the session, the participant's freedom to withdraw their participation/data is reiterated. Furthermore, the process of withdrawing consent is explained in detail at this stage. The process for withdrawing is described in section 3.1</p>
5.7 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?
<p>None of the results reported from the study will include information that allows identification of named individuals. No reference will be made to individual persons or participants, either by name or type of data supplied. However, a coding scheme will be used. Each participator will be issued with a "code" which will be used to store their data. This code will be required should the user wish to withdraw their consent/data from the study.</p> <p>All data will be stored on University equipment. The information will be collected and stored on a University computer. This computer is password protected and exists behind a University firewall. The computer is located in a secure building.</p>

6. VULNERABLE PARTICIPANTS (Minors <18 years, and Vulnerable Adults)

6.1 Are some or all of the participants:	(Delete as applicable)
Under the age of 16?	No

Between the ages of 16 and 18?	No
Vulnerable adults? (See guidelines)	No
If no to all, please proceed to section 7.	
If yes, please continue and consult guidelines for working with minors and/or vulnerable groups.	

6.2 Describe the vulnerability (for minors give age ranges)		
6.3 Inclusion / exclusion criteria		
6.4 How will minors and vulnerable adults give informed consent?		
Please delete as applicable and explain below (See guidelines)		
For minors < 16 only:	Opt-in	Opt-out
If opt-out, why?		
6.5a Consent form(s) for minor/vulnerable adult attached		
Delete as applicable:	No	Yes
If no, why not?		
6.5b Information sheet(s) for minor/vulnerable adult attached		
Delete as applicable:	No	Yes
If no, why not?		
6.6a Consent form(s) for parent / legal guardian attached		
Delete as applicable:	No	Yes
If no, why not?		
6.6b Information sheet(s) for parent / legal guardian attached		
Delete as applicable:	No	Yes
If no, why not?		
6.7 How will parent/legal guardians, minors and/or vulnerable adults be made aware of their right to withdraw at any time?		
6.8 How will confidentiality be maintained, including archiving / destruction of primary data where appropriate, and how will the security of the data be maintained?		
Investigators working with children and vulnerable adults legally require clearance from the Disclosure and Barring Service (DBS)		
6.9 Do ALL experimenters in contact with children and vulnerable adults have <u>current</u> DBS clearance? Please include photocopies.		
Delete as applicable:	No	Yes
If no, explain		

Delete as applicable:	No	Yes
<i>If yes, please describe</i>		
10.1b Do any of the third parties have a conflict of interest?		
Delete as applicable:	No	Yes
<i>If yes, please describe</i>		

11. ADDITIONAL INFORMATION

11.1 Give details of any professional bodies whose ethical policies apply to this research
NON
11.2 Please give any additional information that you wish to be considered in this application
NON

12. ETHICAL PROTOCOL and DECLARATION

To the best of our knowledge and belief, this research conforms to the ethical principles laid down by the University of Plymouth and by any professional body specified in section 10 above.

This research conforms to the University's Ethical Principles for Research Involving Human Participants with regard to openness and honesty, protection from harm, right to withdraw, debriefing, confidentiality, and informed consent.

Sign below where appropriate:

STAFF / RESEARCH POSTGRADUATES

	Print Name	Signature	Date
Principal Investigator:	<u>Sarah Alhammad</u>		<u>30/7/2017</u>
Other researchers:	<u>Dr. Shirley Atkinson</u>		<u>31/7/2017</u>

Consent Form (experiment's consent form)

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND ENGINEERING

Human Ethics Committee

Project: Evaluation of visualization method in programming learning
Consent Form (experiment's consent form)

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language

Brief statement of purpose of work

The purpose of this research is to evaluate the use of visualization in programming learning by measure the students' performance before and after using the proposed visualisation method.

The objectives of this research have been explained to me.

I understand that I am free to withdraw from the research at any stage and up to 6 weeks after the date of the evaluation session, and ask for my data to be destroyed if I wish.

I understand that my anonymity is guaranteed unless I expressly state otherwise. I gave my permission for audio recording the focus group discussion.

I understand that the Principal Investigator of this work will have attempted, as far as possible, to avoid any risks.

Under these circumstances, I agree to participate in the research.

Name:

Signature:

Date:

Consent Form-(Expert interviews' consent form)

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND ENGINEERING

Human Ethics Committee

Project: Evaluation of visualization method in programming learning
Consent Form-(expert interviews' consent form)

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language

Brief statement of purpose of work

The purpose of this research is to evaluate the use of visualization in programming learning from the experts' perspective, the research will seek to gathered experts' feedback about the method .

The objectives of this research have been explained to me.

I understand that I am free to withdraw from the research at any stage and up to 6 weeks after the date of the interview session, and ask for my data to be destroyed if I wish.

I understand that my anonymity is guaranteed unless I expressly state otherwise.
I gave my permission for audio recording.

I understand that the Principal Investigator of this work will have attempted, as far as possible, to avoid any risks.

Under these circumstances, I agree to participate in the research.

Name:

Signature:

Date:

Research Information Sheet- student's experiment (Visualisation group)

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND TECHNOLOGY

Tracing Learning Environment in JAVA programming Language Research Information Sheet- student's experiment Visualisation group

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language: Evaluation of visualisation method in programming learning

You have been invited to take part in this tool evaluation study which is a voluntary study. So, if you are interested in participating in this study, please take some time to read the following information carefully and make sure the study and its procedure are clear before signing the consent form.

The study involves two groups: experiment group and control group.

To participate in the study, you have been selected to be a member of the experiment group. This sheet is to describe the overall procedure that members of the experiment group will experience.

Aim of research

A plenty of visualization software has emerged recently with the aim of support the learning how to program for novices programmer. Each tool has its features that may or may not be useful for better understanding. The software that considered in this research is the one that based on using memory-referencing and visualizing what happens in the single execution of program statement. The effectiveness of the existing educational software is an essential part of gathering experts' feedback about the method for further improvements.

The purpose of this research is to evaluate the use of visualization in programming learning by comparing student's comprehension who is a member of the experiment group before and after using the visualization method. After that, The survey of experiment group will be compared with the survey of the control group who rely on lectures notes and lab practice in their comprehension.

Description of the overall procedure

The study will incorporate experiment with the students using tasks that the participants will be asked to solve programming problems and completing surveys regarding their experiment

All experiment sessions will last approximately 1 hour, 50 minutes including the focus group discussion.

Focus group discussion will be audio recording. Ethical approval for the recording of these sessions is sought explicitly so that note-taking does not interfere with the process. In particular, the discursive process when users are describing the problems they see/have experienced when using the tools.

An overview of the timetable is given below:

1. Welcome (5 minutes). This includes an overview of sessions and its duration.
2. An introduction (5 minutes). This includes an overview of the subject and the study objectives
3. Task 1, 2 and 3 (50 minutes divided over three interlocutors, 15 min, 20 min, 15 min)
4. Focus group discussion (45 minutes)
5. Debrief (5 minutes)

Description of risks

There are no procedures or content in this study that could be expected to cause physical or psychological harm.

Benefits of proposed research

This research will provide several significant benefits. Firstly, it is an opportunity to know the student's experience and background obtained during their programming learning courses. Secondly, to measure students' performance before and after using visualization method and identify the drawbacks that require further design/development. Finally, it is an opportunity to gather suggestions and opinions about the subject so that framework can be updated to increase the usefulness of visualization method in the educational field.

Right to withdraw

If you note that to take part in this research, please keep this information sheet. Also, you have the freedom to withdraw your participation/data at any time **up to 6 weeks** after the date of the evaluation session. To withdraw, your participation in this study merely contact Mrs. Sarah Alhammad, stating your unique "participators code" that is this code given on the day.

Confidentially

None of the results reported from the study will include information that allows identification of named individuals. No reference will be made to individual persons or participants, either by name or type of data supplied. However, a coding scheme will be used. Each participator will be issued with a "code" which will be used to store their data. This code will be required should the user wish to withdraw their consent/data from the study.

The researcher will use an audio recorder to record the focus group discussion and store it on the researcher's computer, and it will be deleted from the recorder directly. All data collected from the questionnaire papers and audio from the focus group discussion's will be stored on the researcher computer which is a computer belongs to Princess Norah Bint Abdulrahman University. This computer is password protected and exists behind a University firewall. The computer is located in a secure building.

Please don't hesitate to contact me if you now wish to withdraw or if you have any questions about the project. To do this, please contact Mrs. Sarah Alhammad, email: sarah.alhammad@plymouth.ac.uk

Thank you for taking time to read the information sheet.

Date: / /

If you are dissatisfied with the way the research is conducted, please contact the principal investigator in the first instance: telephone number 01752 586209. If you feel the problem has not been resolved please contact the secretary to the Faculty of Science and Technology Human Ethics Committee: Mrs Paula Simson 01752 584503

Research Information Sheet- student's experiment (Control group)

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND TECHNOLOGY

Tracing Learning Environment in JAVA programming Language
Research Information Sheet- student's experiment
Control group

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language: Evaluation of visualisation method in programming learning

You have been invited to take part in this tool evaluation study which is a voluntary study. So, if you are interested in participating in this study, please take some time to read the following information carefully and make sure the study and its procedure are clear before signing the consent form.

The study involves two groups: experiment group and control group

To participate in the study, you have been selected to be a member of the control group. This sheet is to describe the overall procedure that members of the control group will experience.

Aim of research

A plenty of visualization software has emerged recently with the aim of support the learning how to program for novices programmer. Each tool has its features that may or may not be useful for better understanding. The software that considered in this research is the one that based on using memory-referencing and visualizing what happens in the single execution of program statement. The effectiveness of the existing educational software is an essential part of gathering experts' feedback about the method for further improvements.

The purpose of this research is to evaluate the use of visualization in programming learning by comparing student's comprehension who is a member of the experiment group before and after using the visualization method. After that, The survey of experiment group will be compared with the survey of the control group who rely on lectures notes and lab practice in their comprehension.

Description of the overall procedure

The study will incorporate two surveys (pre-test and post-test). The two surveys will be conducted in two separate days Test1 is the pre-test which should be conducted before computer-lab practice that is part of the course, while Test 2 is the post-test which should be conducted after the students have the computer-lab practice.

All sessions will last approximately 30 minutes for each survey.

An overview of the timetable is given below for each survey:

6. Welcome (5 minutes). This includes an overview of sessions and its duration.
7. An introduction (5 minutes). This includes an overview of the subject and the study objectives
8. Task 1, OR 2 depending whether it is pre-test or post-test (15 minutes)
9. Debrief (5 minutes)

Description of risks

There are no procedures or content in this study that could be expected to cause physical or psychological harm.

Benefits of proposed research

This research will provide several significant benefits. Firstly, it is an opportunity to know the student's experience and background obtained during their programming learning courses. Secondly, to measure students' performance before and after having lab practice. Finally, it is an opportunity to gather suggestions and opinions about the subject so that framework can be updated to increase the usefulness of visualization method in the educational field.

Right to withdraw

If you note that to take part in this research, please keep this information sheet. Also, you have the freedom to withdraw your participation/data at any time **up to 6 weeks** after the date of the evaluation session. To withdraw, your participation in this study merely contact Mrs. Sarah Alhammad, stating your unique "participators code" that is this code given on the day.

Confidentially

None of the results reported from the study will include information that allows identification of named individuals. No reference will be made to individual persons or participants, either by name or type of data supplied. However, a coding scheme will be used. Each participator will be issued with a "code" which will be used to store their data. This code will be required should the user wish to withdraw their consent/data from the study.

All data collected from the questionnaire papers will be stored on the researcher computer which is a computer belongs to Princess Norah Bint Abdulrahman University. This computer is password protected and exists behind a University firewall. The computer is located in a secure building.

Please don't hesitate to contact me if you now wish to withdraw or if you have any questions about the project. To do this, please contact Mrs. Sarah Alhammad, email: sarah.alhammad@plymouth.ac.uk

Thank you for taking time to read the information sheet.

Date: / /

If you are dissatisfied with the way the research is conducted, please contact the principal investigator in the first instance: telephone number 01752 586209. If you feel the problem has not been resolved please contact the secretary to the Faculty of Science and Technology Human Ethics Committee: Mrs Paula Simson 01752 584503

Research Information Sheet- expert's interview

PLYMOUTH UNIVERSITY

FACULTY OF SCIENCE AND TECHNOLOGY

Tracing Learning Environment in JAVA programming Language
Research Information Sheet- expert's interview

Name of Principal Investigator

Sarah Alhammad

Title of Research

Tracing Learning Environment in JAVA programming Language: Evaluation of visualisation method in programming learning

You have been invited to take part in this tool evaluation study which is a voluntary study. So, if you are interested in participating in this study, please take some time to read the following information carefully and make sure the study and its procedure are clear before signing the consent form.

Aim of research

A plenty of visualization software has emerged recently with the aim of support the learning how to program for novices programmer. Each tool has its features that may or may not be useful for better understanding. The software that considered in this research is the one that based on using memory-referencing and visualizing what happens in the single execution of program statement. The effectiveness of the existing educational software is an essential part of gathering experts' feedback about the method for further improvements.

The purpose of this research is to evaluate the use of visualization in programming learning from the expert's perspective.

Description of the overall procedure

The study will incorporate interviews with experts in the field.

Interviews will be audio recording. Ethical approval for recording of these sessions is sought explicitly, so that note-taking does not interfere with the process. In particular, the discursive process when users are describing the problems they see/have experienced when using the tools.

All interviews' sessions will last approximately 1 hour,15 minutes.

1. Welcome (5 minutes). This includes an overview of sessions and its duration.
2. An introduction (5 minutes).This includes an overview of the subject and the study objectives
3. Sessions 1,2, and 3 (60 minutes divided over three interlocutors, 10 min, 20 min, 30 min)
4. Final feedback and thanks (5 minutes)

Description of risks

There are no procedures or content in this study that could be expected to cause physical or psychological harm.

Benefits of proposed research

This research will provide an opportunity to gather suggestions and opinions from experts on the subject so that the Ph.D. framework can be updated to increase the usefulness of visualization method in the educational field.

Right to withdraw

If you note that to take part in this research, please keep this information sheet. Also, you have the freedom to withdraw your participation/data at any time **up to 6 weeks** after the date of the evaluation session. To withdraw, your participation in this study merely contact Mrs. Sarah Alhammad, stating your unique “participants code” that is this code given on the day.

Confidentially

None of the results reported from the study will include information that allows identification of named individuals. No reference will be made to individual persons or participants, either by name or type of data supplied. However, a coding scheme will be used. Each participant will be issued with a “code” which will be used to store their data. This code will be required should the user wish to withdraw their consent/data from the study.

The researcher will use an audio recorder to record the interviews and store it on the researcher’s computer, and it will be deleted from the recorder directly. All data collected from the interviews will be stored on the researcher computer which is a computer belongs to Princess Norah Bint Abdulrahman University. This computer is password protected and exists behind a University firewall. The computer is located in a secure building.

Please don’t hesitate to contact me if you now wish to withdraw or if you have any questions about the project. To do this, please contact Mrs. Sarah Alhammad, email: sarah.alhammad@plymouth.ac.uk

Thank you for taking time to read the information sheet.

Date: / /

If you are dissatisfied with the way the research is conducted, please contact the principal investigator in the first instance: telephone number 01752 586209. If you feel the problem has not been resolved please contact the secretary to the Faculty of Science and Technology Human Ethics Committee: Mrs Paula Simson 01752 584503

Appendix E- Evaluation survey- Focus Group questions- Expert interviews questions

E-1 Pre-survey for control and visualisation group

Student Survey Questionnaire of evaluating the use of visualisation in programming (Visual code flow)

Call procedure lesson

Section1:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of confidence in programming					
I can program competently in at least one text-based programming language.					
I can use variables.					
I can use the relational operators.					
I can use the arithmetic operators.					
I can use the procedures.					
I know how to call procedures					
I know how to pass a parameter(s).					
I know how procedures return value(s) to the program.					
I can debug syntax errors					
I can debug logical errors					

Section 2:

Trace the following code carefully and complete the questionnaire regarding the code

1	class parameterExample
2	{
3	public static void main (String[] args)
4	{
5	double width = 10.0 ;
6	double length = 20.0 ;
7	calculateArea (width , length);
8	}
9	static void calculateArea(double theWidth, double theLength)
10	{
11	double area;
12	area = theWidth * theLength ;
13	System.out.println("The area is : ");
14	System.out.println (area);
15	}
16	}

Please answer the questions in the first column and then choose the level of your confidence in solving the question

Question	Answer	Not confident	Confident	Very confident
		Participant's level of confidence in solving the questions		
What is the purpose of the program?				
What is the output of the program?				
What does the statement in line 7 do?				
What is the value of the variable width after executing line 7				
What is the value of the variable length after executing line 7				
What is the value of the variable theWidth after executing line 9				
What is the value of the variable theLength after executing line 9				
What is the value of the variable area after executing line 12				
What is displayed after executing line 13				
What is displayed after executing line 14				
Which line(s) of the code (if any) did you struggle with understanding	<input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7 <input type="radio"/> 8 <input type="radio"/> 9 <input type="radio"/> 10 <input type="radio"/> 11 <input type="radio"/> 12 <input type="radio"/> 13 <input type="radio"/> 14 <input type="radio"/> 15 <input type="radio"/> 16 <input type="radio"/> Non			

Student Survey Questionnaire of evaluating the use of visualization in programming (Visual Java code)

Classes and objects lesson

Section1:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of confidence in programming					
I can program competently in at least one text-based programming language.					
I can use variables.					
I can use the relational operators.					
I can use the arithmetic operators.					
I can use classes and objects.					
I know how to use constructors					
I can debug syntax errors					
I can debug logical errors					

Section 2:

Trace the following code carefully and complete the questionnaire regarding the code

```
1 public class Accountexample
2 {
3     public static void main(String[] args)
4     {
5         Account account1 = new Account ( 50 );
6         Account account2 = new Account ( -7.53 );
7         System.out.print("First account: ");
8         System.out.println(account1.getBalance ( ));
9         System.out.print("Second account: ");
10        System.out.println(account2.getBalance ( ));
11        double bonus = 100;
12        account1.addMoney ( bonus );
13        account2.addMoney( bonus );
14        System.out.print("First account after adding bonus: ");
15        System.out.println(account1.getBalance ( ));
16        System.out.print("Second account after adding bonus: ");
17        System.out.println(account2.getBalance ( ));
18    }
19 }
20 class Account
21 {
22     double balance;
23     public Account ( double initialBalance)
24     {
25         balance = initialBalance;
26     }
27     public void addMoney( double amount )
28     {
29         balance = balance + amount ;
30     }
31     public double getBalance()
32     {
33         return balance;
34     }
35 }
```

Please answer the questions in the first column and then choose the level of your confidence in solving the question

Question	Answer	Not confident	Confident	Very confident
		Participant's level of confidence in solving the questions		
What is the purpose of the program?				
What is the output of the program?				
What is the value of variable balance in account1 after executing line 5				
What is the value of variable balance in account2 after executing line 6				
What is displayed after line 8 is executed				
What is displayed after line 10 is executed				
What is the value of variable balance in account1 after executing line 12				
What is the value of variable balance in account2 after executing line 13				
What is displayed after line 15 is executed				
What is displayed after line 17 is executed				
Which line(s) of the code (if any) did you struggle with understanding	<div style="display: flex; flex-wrap: wrap; gap: 5px;"> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 </div> <div style="display: flex; flex-wrap: wrap; gap: 5px;"> 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 Non </div>			

Student Survey Questionnaire of evaluating the use of visualization in programming (Visual Java code)

Class inheritance lesson

Section1:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of confidence in programming					
I can program competently in at least one text-based programming language.					
I can use variables.					
I can use the relational operators.					
I can use the arithmetic operators.					
I can use classes and objects.					
I can use the class inheritance.					
I can debug syntax errors					
I can debug logical errors					

Section 2:

Trace the following code carefully and complete the questionnaire regarding the code

```
1 public class InheritanceExample
2 {
3     public static void main(String[] args)
4     {
5         int input1 = 20;
6         int input2 = 10;
7         MathsLevel1 test = new MathsLevel1 ();
8         test.addition(input1,input2);
9         MathsLevel2 example = new MathsLevel2 ();
10        example.subtraction(input1,input2);
11        example.multiplication(input1,input2);
12    }
13 }
14 public class MathsLevel1
15 {
16     int z;
17     public void addition(int x, int y)
18     {
19         z = x + y;
20         System.out.println("The sum of the given numbers:"+z);
21     }
22     public void subtraction(int x, int y)
23     {
24         z = x - y;
25         System.out.println("The difference between the given numbers:"+z);
26     }
27 }
28 public class MathsLevel2 extends MathsLevel1
29 {
30     public void multiplication(int x, int y)
31     {
32         z = x * y;
33         System.out.println("The product of the given numbers:"+z);
34     }
35 }
```

Please answer the questions in the first column and then choose the level of your confidence in solving the question

Question	Answer	Not confident	Confident	Very confident
		Participant's level of confidence in solving the questions		
What is the purpose of the program?				
What is the output of the program?				
What does the statement in line 28 do?				
What are the attributes of object test after executing line 7				
What is the value of variable z after executing line 8				
What is displayed after line 20 is executed				
What are the attributes of object example after executing line 9				
What is the value of variable z after executing line 10				
What is displayed after line 25 is executed				
What is the value of variable z after executing line 11				
What is displayed after line 33 is executed				
Which line(s) of the code (if any) did you struggle with understanding	<div style="display: flex; flex-wrap: wrap; gap: 5px;"> 123456789101112131415161718 1920212223242526272829303132333435Non </div>			

E-2 Post-survey for control and visualisation group

Note: Section 1 and 2 have been presented to both groups, while section 3 has been presented for visualization group only

Student Survey Questionnaire of evaluating the use of visualisation in programming (Visual Java code)

Call procedure lesson

Section1:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of confidence in programming					
I can program competently in at least one text-based programming language.					
I can use variables.					
I can use the relational operators.					
I can use the arithmetic operators.					
I can use the procedures.					
I know how to call procedures					
I know how to pass a parameter(s).					
I know how procedures return value(s) to the program.					
I can debug syntax errors					
I can debug logical errors					

Section 2:

Trace the following code carefully and complete the questionnaire regarding the code

```
1 class parameterExample
2 {
3     public static void main ( String[] args )
4     {
5         double width = 15.0 ;
6         double length = 12.0 ;
7         calculatePerimeter ( width , length );
8     }
9     static void calculatePerimeter ( double theWidth, double theLength )
10    {
11        double perimeter;
12        perimeter = 2 * (theWidth + theLength);
13        System.out.println("The perimeter is : ");
14        System.out.println (perimeter);
15    }
16 }
```

Please answer the questions in the first column and then choose the level of your confidence in solving the question

Question	Answer	Not confident	Confident	Very confident
		Participant's level of confidence in solving the questions		
What is the purpose of the program?				
What is the output of the program?				
What does the statement in line 7 do?				
What is the value of the variable width after executing line 7				
What is the value of the variable length after executing line 7				
What is the value of the variable theWidth after executing line 9				
What is the value of the variable theLength after executing line 9				
What is the value of the variable perimeter after executing line 12				
What is displayed after executing line 13				
What is displayed after executing line 14				
Which line(s) of the code (if any) did you struggle with understanding	<input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7 <input type="radio"/> 8 <input type="radio"/> 9 <input type="radio"/> 10 <input type="radio"/> 11 <input type="radio"/> 12 <input type="radio"/> 13 <input type="radio"/> 14 <input type="radio"/> 15 <input type="radio"/> 16 <input type="radio"/> Non			

Section 3:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of satisfaction					
Using the tool helped me understand the procedure call					
I would prefer to use the tool to trace the code rather than manual approach in section 2					
I would prefer to use the manual approach in section 2 to trace the code, rather than the tool					
The tool was useful for learning Java					
Using the tool helped me understand the changes in the variable values					
The tool is easy to use					
I would use a tool like this one					

Please feel free to add any comments or suggestions to improve the tool in the following box

Student Survey Questionnaire of evaluating the use of visualization in programming (Visual Java code)

Classes and objects lesson

Section1:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of confidence in programming					
I can program competently in at least one text-based programming language.					
I can use variables.					
I can use the relational operators.					
I can use the arithmetic operators.					
I can use classes and objects.					
I know how to use constructors					
I can debug syntax errors					
I can debug logical errors					

Section 2:

Trace the following code carefully and complete the questionnaire regarding the code

```
1 public class ExamGrade
2 {
3     public static void main(String[] args)
4     {
5         Exam exam1 = new Exam ( 90 );
6         Exam exam2 = new Exam( 70 );
7         System.out.print("First exam grade : ");
8         System.out.println(exam1.getGrade ( ) );
9         System.out.print("Second exam grade: ");
10        System.out.println(exam2.getGrade ( ) );
11        double bonus = 5.0;
12        exam1.addBonus ( bonus ) ;
13        exam2.addBonus( bonus ) ;
14        System.out.print("First exam grade after adding bonus: ");
15        System.out.println(exam1.getGrade ( ) );
16        System.out.print("Second exam grade after adding bonus: ");
17        System.out.println(exam2.getGrade ( ) );
18    }
19 }
20 class Exam
21 {
22     double grade;
23     public Exam ( double initialGrade)
24     {
25         grade = initialGrade;
26     }
27     public void addBonus( double plus )
28     {
29         grade = grade + plus;
30     }
31     public double getGrade()
32     {
33         return grade;
34     }
35 }
```

Please answer the questions in the first column and then choose the level of your confidence in solving the question

Question	Answer	Not confident	Confident	Very confident
		Participant's level of confidence in solving the questions		
What is the purpose of the program?				
What is the output of the program?				
What is the value of variable grade in exam1 after executing line 5				
What is the value of variable grade in exam2 after executing line 6				
What is displayed after line 8 is executed				
What is displayed after line 10 is executed				
What is the value of variable grade in exam1 after executing line 12				
What is the value of variable grade in exam2 after executing line 13				
What is displayed after line 15 is executed				
What is displayed after line 17 is executed				
Which line(s) of the code (if any) did you struggle with understanding	<div style="display: flex; flex-wrap: wrap; gap: 5px;"> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 </div> <div style="display: flex; flex-wrap: wrap; gap: 5px;"> 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 Non </div>			

Section 3:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of satisfaction					
Using the tool helped me understand classes					
Using the tool helped me understand objects					
Using the tool helped me understand the difference between classes and objects					
Using the tool helped me understand the class constructor					
Using the tool helped me understand how to set the variable value defined in the class					
Using the tool helped me understand how to get the variable value defined in the class					
I would prefer to use the tool to trace the code rather than manual approach in section 2					
I would prefer to use the manual approach in section 2 to trace the code, rather than the tool					
The tool was useful for learning Java					
Using the tool helped me understand the changes in the variable values					
The tool is easy to use					
I would use a tool like this one					

Please feel free to add any comments or suggestions to improve the tool in the following box

Student Survey Questionnaire of evaluating the use of visualization in programming (Visual Java code)

Class inheritance lesson

Section1:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of confidence in programming					
I can program competently in at least one text-based programming language.					
I can use variables.					
I can use the relational operators.					
I can use the arithmetic operators.					
I can use classes and objects.					
I can use the class inheritance.					
I can debug syntax errors					
I can debug logical errors					

Section 2:

Trace the following code carefully and complete the questionnaire regarding the code

```
1 public class InheritanceExample
2 {
3     public static void main(String[] args)
4     {
5         int input1 = 50;
6         int input2 = 2;
7         MathsLevel1 test = new MathsLevel1 ();
8         test.multiplication (input1,input2);
9         MathsLevel2 example = new MathsLevel2 ();
10        example.division(input1,input2);
11        example.remainder(input1,input2);
12    }
13 }
14 public class MathsLevel1
15 {
16     double z;
17     public void multiplication(int x, int y)
18     {
19         z = x * y;
20         System.out.println("The product of the given numbers:"+z);
21     }
22     public void division(int x, int y)
23     {
24         z = x / y;
25         System.out.println("The quotient of the given numbers:"+z);
26     }
27 }
28 public class MathsLevel2 extends MathsLevel1
29 {
30     public void remainder(int x, int y)
31     {
32         z = x % y;
33         System.out.println("The remainder after dividing the given numbers:"+z);
34     }
35 }
```

Please answer the questions in the first column and then choose the level of your confidence in solving the question

Question	Answer	Not confident	Confident	Very confident
		Participant's level of confidence in solving the questions		
What is the purpose of the program?				
What is the output of the program?				
What does the statement in line 28 do?				
What are the attributes of object test after executing line 7				
What is the value of variable z after executing line 8				
What is displayed after line 20 is executed				
What are the attributes of object example after executing line 9				
What is the value of variable z after executing line 10				
What is displayed after line 25 is executed				
What is the value of variable z after executing line 11				
What is displayed after line 33 is executed				
Which line(s) of the code (if any) did you struggle with understanding	<div style="display: flex; flex-wrap: wrap; gap: 5px;"> 123456789101112131415161718 1920212223242526272829303132333435Non </div>			

Section 3:

Please complete the following questionnaire, by placing a CROSS in the appropriate space

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Participant's level of satisfaction					
Using the tool helped me understand the class inheritance					
Using the tool helped me understand how to set the value of the variable that is inherited from the super-class					
Using the tool helped me understand how to get the value of the variable that is inherited from the super-class					
I would prefer to use the tool to trace the code rather than the manual approach in section 2					
I would prefer to use the manual approach in section 2 to trace the code, rather than the tool					
The tool was useful for learning Java					
The tool is easy to use					
I would use a tool like this one					

Please feel free to add any comments or suggestions to improve the tool in the following box

E-3 Questions for Focus Groups

List of open-ended questions in the focus group discussion to evaluate the use of visualisation in programming learning (Visual code flow)

Group reference :		Date of discussion:	
Interview Start and End Times:		Moderator :	

Dear participant

Thank you for taking the time to have this discussion. The discussion is an evaluation of using visualization method that assists the leaning of programming and gathering the students's opinion and suggestions.

Regards

Sarah Alhammad

Section one: Short Questions

Participants reference # :

Which level or course of programming do you study now?

Which programming language do you learn ?

What materials or web-sites do you use beside the lecture notes in studying programming ?

Have you ever use visualization tools to improve your understanding for programming? (interviewer should explain the nature of visualization tool)

- If so, What is the tool?

- How often you use it?

- What the aspects you like/dislike on the tool?

Section two: Open-ended questions

Note that the discussion evolve depending on the topic and on the students answers.

The following is suggested questions to guide the discussion

What are the strengthen(s) and weakness(es) of the tool in general?

How did you find the method representation ? things you like and dislike ?

How did you find the variable representation ? things you like and dislike ?

How did you find the expression of evaluation ? things you like and dislike ?

How did you find the passing parameters and calling method representation ? things you like and dislike ?

How did you find the class representation ? things you like and dislike ?

How did you find the object representation ? things you like and dislike ?

How did you find the class inheritance representation ? things you like and dislike ?

How did you find the output representation ? things you like and dislike ?

How did you find the animation? things you like and dislike ?

How did you find the control of execution ? things you like and dislike ?

would you use the tool if it's possible? Explain why or why not

Do you think it is beneficial and may improve programming learning?

What should other types of improvements be implemented to build on this tool?

E-4 Questions for semi-structured interviews- expert interview

List of open-ended questions to evaluate the use of visualisation in programming learning (Visual code flow)- expert evaluation

Participant code:		Date of Interview:	
Interview Start and End Times:		Notes were taken by:	

Dear participant

Thank you for taking the time to have this discussion. The discussion is an evaluation of using visualization method that assists the leaning of programming and gathering the expert's opinion and suggestions.

Regards

Sarah Alhammad

Personal information	
Name (optional)	
Age	
Gender	
Degree	
Occupation	

section1: Background information (10 minutes)

Did you teach Programming language(s), if yes proceed to the following questions

How long have you been teaching programming ?

Which level or course of programming do you teach?

Which programming language do you teach?

What is your experience (if any) on using visualization tools to improve your teaching for programming courses?(interviewer should explain the nature of visualization tool)

If yes, the following questions will be asked:

- o What is the tool? (describe how it works and its method)
- o How often you use it?
- o What are the tool features?
- o What the aspects you like/dislike on the tool?
- o To what extent it was helpful?

Section 2: Evaluated questions for Visual Java Code tool(20 minutes for each activity) same questions

What are the strengthen(s) and weakness(es) of the tool?

Will you use it if possible? Explain why or why not

Do you think it is beneficial and may improve programming learning?

What should other types of improvements be implemented to build on this tool?

How did you find the method representation ? things you like and dislike

How did you find the variable representation ? things you like and dislike

How did you find the expression of evaluation ? things you like and dislike

How did you find the passing parameters and calling method representation ? things you like and dislike

How did you find the class representation ? things you like and dislike

How did you find the object representation ? things you like and dislike

How did you find the class inheritance representation ? things you like and dislike

How did you find the output representation ? things you like and dislike

How did you find the animation? things you like and dislike

How did you find the control of execution ? things you like and dislike

Appendix F- Interviews and focus groups transcript

F-1 Interviews with programming students in the data collection Phase

Coding Framework Table

Initial coding framework

Note: letter P refer to the participant

Interview transcript	Initial coding framework
<p>P1: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P1: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P1: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P1: e-learning systems.</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P1: lack of e-learning system, more training.</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:’ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ? P1: <u>Jeliot:</u> ‘Like: the appearance of execution which is next to the code directly Dislike:1- the tool cannot go back to previous steps or forward to next steps (no control on the execution steps) 2- in case of representing the classes and inheritance , it was not clear because the classes cascaded and not represented as hierarchy ‘</p>	<p>Windows preference Execution preference Object oriented representation Tools availability Expression evaluation preference Tracing the program code</p>

<p>Describe why? P1: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P1: Online python tutor because the hierarchy of inheritance was sequential</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P1: Both , I think it is clear in both tools and any representation I can understand it</p> <p>P2: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P2: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P2:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P2: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P2: web site that solved programming exercises</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P2: lack of practice.</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:' What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p>	<p>Animation interface preference Debugging and error explanation Tools availability Expression evaluation</p> <p>Execution preference Animation Use of control buttons</p> <p>Statement execution indicator Animation Use of the control button</p>
--	---

<p>P2</p> <p><u>Jeliot:</u> 'Dislike: 1- no control on the steps of execution or saving the action history 2- the representation of classes was not clear , the arrows showed to represent the relation between classes was not obvious Like: the control in the execution speed ' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the loop tracing Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing</p> <p><u>Online python tutor:</u> 'Like: 1- the use of green and red pointers (arrows) to indicate the statement execution whether it is under execution or will be the next statement to be executed 2- she likes the control of execution (the existence of next and back buttons)' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the tracing, and how it show the classes and object Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing.</p> <p><u>Visual logic:</u> 'It is an advanced tool that could be used to learn the flow of programming but not how to write program since there is no code to be blogged and evaluated It is suitable for program designers not for code programming' Interviewer:Will you use it if possible ? explain why or why not</p> <p>No not for learning programming but may be to understand the programming steps through flowchart. Interviewer:Do you think it is beneficial and may improve programming learning? No</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool? Support the coding trace in addition to the flowchart</p>	<p>Error explanation Execution speed Trace the program code Tools availability</p> <p>Interface preference Tools availability Use of control buttons</p>
---	---

<p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P2: Jeliot because I like the loop tracing.</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P2: Online python tutor because the hierarchy of inheritance was clear</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P2: jeliot, because the highlighted lines were clear and shows the parameters</p> <p>P3: Name (optional) Age : 22-24 Gender: Female Occupation : graduated without job</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P3: C,C++,PHP,HTML</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P3: few months in the graduation project</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P3: yes, the built in tracing tool in C programming language.</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P3: e-learning systems.</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P3: nothing</p> <p>TASK2: Evaluated questions for each tool(25 minutes) Interviewer:' What are the strengthen(s) and weakness (es) of the</p>	<p>Interface preference Code font Error explanation Expression evaluation Tools availability Tracing the program code Execution speed</p>
--	---

<p>Support program code and trace the code line by line. Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P3: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P3: Online python tutor because the class and object with their attribute is good presented .</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P3: jeliot. I think it is clear when trace the line of calling statement and then jump to the procedure header .</p> <p>P4: Name (optional) Age : 19-21 Gender: male Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P4: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P4:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P4: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P4: learning software that has a bank questions.</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P4: extra practice using learning software and learning websites.</p>	<p>Animation Error explanation Execution preference Interface preference Program language Tools availability</p> <p>Animation Error explanation Execution preference Interface preference Program language Tools availability</p> <p>Animation Error explanation Execution preference Interface preference Ease of use Tools availability</p> <p>Error explanation Execution preference Interface preference Tools availability</p> <p>Error explanation</p> <p>Animation Interface preference Program language Tools availability</p>
---	---

<p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:' What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p>P4:</p> <p><u>Jeliot:</u> 'Strengthen: like the animation, the interface, and error explanation There is no weakness in general but I prefer the online tools '</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the navigation buttons, the expression explanations, I like the tracing and how it shows the changing in the variables.</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Support moreprogramming languages.</p> <p><u>Online python tutor:</u> 'Strengthen: online tool, clear tool Weakness: the design need to be developed, similar to the manual tracing, and the absence of statement expression'</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the tracing, and the class representation. I like that it is easy to access online</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool? In each statement in the code , it doesn't show in details the change of the variables' values. I think it need to be added.</p> <p><u>Visual logic:</u> 'It helps on learning the concepts of programming but not how to program'</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>No, because I already know how to work with flowcharts.</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning? Only for people who wants to understand the flowcharts.</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing</p>	<p>Correction suggestion Tools availability Interface preferences</p> <p>Animation Error explanation Execution speed and control Interface preference Program language Tools availability</p> <p>Animation Error explanation Execution preference Interface preference Program language Tools availability</p> <p>Error explanation Trace the code Tools availability</p>
--	---

<p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P4: Jeliot because expression and variables changing is much clear.</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P4: Online python tutor because the hierarchy of inheritance was clear.</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P4: Jeliot and online tutor.</p> <p>P5: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P5: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P5:Few days for Programming homework and during lab activity.</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P5: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P5: learning websites.</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P5: extra practice time and questions.</p> <p>TASK2: Evaluated questions for each tool(25 minutes) Interviewer:' What are the strengthen(s) and weakness (es) of the</p>	
--	--

<p>Jeliot\online python tutor\visual logic tools ?</p> <p>P5:</p> <p><u>Jeliot:</u> 'I like all the features in the tool Weakness: the execution control in not enough , I prefer if I can control the execution more ,for example, going back and forth for each statement ' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like all of the features especially the navigation buttons and the windows division. Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool? The execution control in not enough , I prefer if I can control the execution more ,for example, going back and forth for each statement '</p> <p><u>Online python tutor:</u> 'The control button is suitable for beginners but not for the experts Weakness: The absence of animation' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the tracing and navigations. Also I like the classes representation. Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool? I would like to add buttons to play the execution and pause it .</p> <p><u>Visual logic:</u> 'It does not help on how to program, it is beneficial to learn how to understand the programming concepts' Interviewer:Will you use it if possible ? explain why or why not</p> <p>No ,It does not help on how to program, it is beneficial to learn how to understand the programming concepts.</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool? Code tracing like Jeliot and online python tutor. Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p>	
---	--

<p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P5: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P5: Online python tutor because the hierarchy of inheritance was sequential</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P5: Both , I think it is clear in both tools and any representation I can understand it</p> <p>P6: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P6: Java</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days in the lab activity.</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P6: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P6: e-learning systems.</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P6: lack of e-learning system.</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:' What are the strengthen(s) and weakness (es) of the</p>	
---	--

<p>Jeliot\online python tutor\visual logic tools ?</p> <p>P6:</p> <p><u>Jeliot:</u> 'I like the tool and I like how it represents the results for each statement in the code Weakness: nothing ' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the tracing Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing</p> <p><u>Online python tutor:</u> 'Strengthen : the use of red and green arrows as indication of statement execution Weakness: The absence of animation , I don't like the control button and I prefer the animation' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the tracing, and how it show the classes and object Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing</p> <p><u>Visual logic:</u> 'It does not help on how to program, it is beneficial to learn how to understand the programming concepts' Interviewer:Will you use it if possible ? explain why or why not</p> <p>No Interviewer:Do you think it is beneficial and may improve programming learning? No</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p>	
---	--

<p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P6: Jeliot because the loop counter was clear</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P6: Online python tutor because , objects attribute was clearly presented also the inheritance representation easy to understand</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P6: jeliot , I need to lear more in Java programming.</p> <p>P7: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P7: JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P7:Few days when I solve a programming homework.</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P7: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P7: more websites like Khan-academy.</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P7: we don't know much tracing tools that may help us to understand programming.</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:' What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p>	
--	--

<p>P7:</p> <p><u>Jeliot:</u> 'Strengthen : error explanation, animation, showing what happen to the variables in memory during execution Weakness: I want to slow down the execution speed ' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like error explanation, animation, showing what happen to the variables in memory during execution Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>I want to slow down the execution speed</p> <p><u>Online python tutor:</u> 'Strengthen : the tracing is good Weakness: I don't like the online tools to avoid the connection problems' Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the tracing, and how it show the classes and object Interviewer:Do you think it is beneficial and may improve programming learning? yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing except I don't like the online tools to avoid the connection problems</p> <p><u>Visual logic:</u> 'It does not help on how to program, it is beneficial to learn how to understand the programming concepts' Interviewer:Will you use it if possible ? explain why or why not It help flowchart learners only as beginning on how to program</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning? No</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>No</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p>	
--	--

<p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P7: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P7: Online python tutor because the hierarchy of inheritance was sequential</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P7: Both , I think it is clear in both tools and any representation I can understand it</p> <p>P8: Name (optional) Age : 19-21 Gender: Male Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P8: Java</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P8:Few days practice for exam.</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P8: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P8: I don't know</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P8: more training through bank questions in learning website.</p> <p>TASK2: Evaluated questions for each tool(25 minutes) Interviewer:' What are the strengthen(s) and weakness (es) of the</p>	
--	--

<p>Jeliot\online python tutor\visual logic tools ?</p> <p>P8:</p> <p><u>Jeliot:</u> 'Strengthen : the interface, and the tool is showing what happen to the variables in memory during execution '</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the interface, and the tool is showing what happen to the variables in memory during execution</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning?</p> <p>yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>Nothing</p> <p><u>Online python tutor:</u> 'Strengthen : I like the online tool, Weakness: the execution control buttons because every time I need to click on the forward and back buttons'</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>Yes, I like the interface, is the tool online and support many programming langauges.</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning?</p> <p>yes</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>the execution control buttons because every time I need to click on the forward and back buttons</p> <p><u>Visual logic:</u> 'It does not help on how to program, it is beneficial to learn programming concepts'</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>No, I am interested.</p> <p>Interviewer:Do you think it is beneficial and may improve programming learning?</p> <p>No</p> <p>Interviewer:What other types of improvements should be implemented to build on this tool?</p> <p>I don't know may be support code tracing.</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ?</p>	
--	--

<p>Describe why? P8: Jeliot , I like the loop tracing more in jeliot than online tutor.</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P8: Online python tutor</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P8: jeliot for loop and variable tracing but online tutor for class representation and object oriented programming.</p> <p>P9: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P9: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P9: No</p> <p><u>Jeliot:</u> 'Strengthen : the interface, the speed execution Weakness: the code font is small '</p> <p><u>Online python tutor:</u> 'Strengthen : it is online Weakness: the error explanation was not good , and the absence of expression evaluation'</p> <p><u>Visual logic:</u> 'It is nothing to do on how to write a program , I do not think it will help me to write or trace a program '</p> <p>P10 Name (optional) Age : 19-21 Gender: male Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p>	
--	--

<p>P10: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P10: No</p> <p><u>Jeliot:</u> 'Strengthen : the whole tracing method Weakness: it supports only one language '</p> <p><u>Online python tutor:</u> 'Strengthen : it is online Weakness: the error explanation was not good , non-friendly interface, and it is not suitable for beginners'</p> <p><u>Visual logic:</u> 'It does not include the code so it does not help '</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P10: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P10: Online python tutor because the hierarchy of inheritance was sequential</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P10: Both , I think it is clear in both tools and any representation I can understand it</p> <p>P11: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P11: C,C++,JAVA</p>	
--	--

<p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P11: No</p> <p>Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P11: assignments</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P11: lack of training</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:’ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> ‘Strengthen : the animation and the visualization that presents the effect of execution on the memory – the error explanation was great Weakness: the execution control should be more flexible ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : the control buttons that make the control of execution more flexible Weakness: the error explanation was not good ,not suitable for beginners because the interface is not well arranged, it is better that the tool in offline mode and run as software on the local machine’</p> <p><u>Visual logic:</u> ‘It does not help on how to program, it is beneficial to learn how to understand the programming concepts ‘</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P11: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P11: Online python tutor because the hierarchy of inheritance was sequential</p>	
--	--

Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why?

P11: Both , I think it is clear in both tools and any representation I can understand it

P12:

Name (optional)

Age : 19-21

Gender: Female

Occupation : Undergraduate student

TASK1: Background information (10 minutes)

Interviewer:What is\are Programming language(s) you have learned

P12: C,C++,JAVA

Interviewer:When was the last time you wrote a program ? what was the purpose? Reason

P12:Few days for Programming homework

Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming?

(interviewer should explain the nature of visualisation tool)

P12: No

Jeliot:

'Strengthen : the interface, the animation , the tracing

Weakness: does not support other languages, the control execution '

Online python tutor:

'Strengthen :

Weakness: the error explanation was not good , I prefer the offline mode'

Visual logic:

'It does not help on how to program, it is beneficial to learn how to understand the programming concepts '

Interviewer:Which tool do you prefer for solving the OBJECT activity ?

Describe why?

P12: Online python tutor because the hierarchy of inheritance was sequential

Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why?

P12: Both , I think it is clear in both tools and any representation I can understand it

Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes)

The discussion will be held for each activity on the effectiveness of the tool

<p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P12: Jeliot , I like the loop demonstration</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P12: Online python tutor because the hierarchy of inheritance was clear and easy to understand</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P12: Both</p> <p>P13: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P13: JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P13:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P13: No Interviewer : What kind of technical assistance would you like to improve your programming learning? Suggestion: (e-learning system, software(s), example databases)</p> <p>P13: learning web sties</p> <p>Interviewer : What technical hurdles remain to you being able to learn programming? P13: tutorial online</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:' What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> 'Strengthen : the error explanation</p>	
--	--

<p>Weakness: the animation needs more control in was fast ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : support more than one language Weakness: how to control the execution was not good, online, the interface’</p> <p><u>Visual logic:</u> ‘It does not include the code so it does not help ‘</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P13: Online python tutor because the hierarchy of inheritance was sequential</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P13: Both , I think it is clear in both tools and any representation I can understand it</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P13: Jeliot because the loop counter is easy to understand</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P13: Online python tutor because the hierarchy of inheritance is easy to understand</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P13: Both , I think it is clear in both tools.</p> <p>P14: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P14: C,C+</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p>	
--	--

P14: No

TASK2: Evaluated questions for each tool(25 minutes)

Interviewer:' What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?

Jeliot:

'Strengthen : control execution, error explanation, the interface

Weakness: the code font is very small '

Online python tutor:

'Strengthen : the control execution is good I like the (forward and backward buttons), online

Weakness: the interface not friendly'

Visual logic:

'Strengthen : trace the flowchart

Weakness: the control execution '

Interviewer:Will you use it if possible ? explain why or why not

No, because it is based on flowchart tracing not the code.

Interviewer:Do you think it is beneficial and may improve programming learning?

No

Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes)

The discussion will be held for each activity on the effectiveness of the tool

Interviewer:Which tool do you prefer for solving the LOOP activity ?

Describe why?

P14: Jeliot because the loop counter was clear and the change of the body loop also

Interviewer:Which tool do you prefer for solving the OBJECT activity ?

Describe why?

P14: Online python tutor because the hierarchy of inheritance was sequential

Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why?

P14: Both , I think it is clear in both tools and any representation I can understand it

P15:

Name (optional)

Age : 19-21

Gender: Female

Occupation : Undergraduate student

TASK1: Background information (10 minutes)

Interviewer:What is\are Programming language(s) you have learned

<p>P15: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P15:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P15: No</p> <p><u>Jeliot:</u> ‘Strengthen : the error explanation, the tracing is clear for novices Weakness: no weakness ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : support more than one language online tool Weakness: the interface is not friendly , I prefer the animation to control the execution ‘</p> <p><u>Visual logic:</u> ‘Weakness: not helpful for tracing especially for long flowchart ‘</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>No, because it is based on flowchart tracing not the code. Interviewer:Do you think it is beneficial and may improve programming learning? No</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P15: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P15: Online python tutor because the hierarchy of inheritance was sequentially represented</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P15: Both</p> <p>P16: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p>	
--	--

<p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P16: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P16:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P16: No</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:’ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> ‘Strengthen : I like all its features Weakness: the correction suggestion for the error is not helpful enough ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : it is an online tool Weakness: interface need to be more colourful ‘</p> <p><u>Visual logic:</u> ‘Can be used for implementing projects but not for writing code ‘</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P16: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P16: Online python tutor because the hierarchy of inheritance was sequential</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P16: Both , I think it is clear in both tools and any representation I can understand it</p> <p>P17: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p>	
--	--

<p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P17: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P17:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P17: No</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:’ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> ‘Strengthen : the interface, the windows, expression evaluation , error explanation Weakness: I did not like the speed of animation ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : it is an online tool, support more than one programming language, the execution control Weakness: interface need to be improved to be more attractive ‘</p> <p><u>Visual logic:</u> ‘Suitable to understand how the flowchart works and to understand the programming semantics and keywords ‘</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P17: Jeliot because the loop counter was clear and easy to know the values of variables involved in the loop</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P17: both are the same , for many classes we can use jeliot while if we have few classes I may choose online python tutor.</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P17: Both of them were good.</p>	
--	--

<p>P18: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P18: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P1:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P18: No</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:‘ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> ‘Strengthen : the interface, the animation in the execution control, error explanation Weakness: nothing ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : it is an online tool, support more than one programming language Weakness: the error explanation was not clear ‘</p> <p><u>Visual logic:</u> ‘Could be used to teach flowchart and programming concepts ‘</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P18: Jeliot because the loop counter was easy to understand.</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P18: Online python tutor because the hierarchy of inheritance was easy to trace</p> <p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why? P18: Both</p>	
--	--

<p>P19: Name (optional) Age : 19-21 Gender: Female Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P19: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P19:Few days for Programming homework</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P19: No</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:‘ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> ‘Strengthen : error explanation , the tracing visualization Weakness: nothing ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : the tracing and the control in the execution Weakness: I do not like the online tool ‘</p> <p><u>Visual logic:</u> ‘It does not show any code so I do not think it will be helpful for learning how to write program concepts ‘</p> <p>Interviewer:Will you use it if possible ? explain why or why not</p> <p>No, because it is based on flowchart tracing not the code. Interviewer:Do you think it is beneficial and may improve programming learning? No</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P19: Jeliot because the loop counter was clear and the change of the body loop also</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P19: Online python tutor because the hierarchy of inheritance was sequential</p>	
---	--

<p>Interviewer:Which tool do you prefer for solving the PASSING_PARAMETER activity ? Describe why?</p> <p>P19: Both , I think it is clear in both tools and any representation I can understand it</p> <p>P20: Name (optional) Age : 19-21 Gender: male Occupation : Undergraduate student</p> <p>TASK1: Background information (10 minutes) Interviewer:What is\are Programming language(s) you have learned</p> <p>P20: C,C++,JAVA</p> <p>Interviewer:When was the last time you wrote a program ? what was the purpose? Reason P20:Few days for project</p> <p>Interviewer : What are your experiences (if any) on using visualisation tools to improve your understanding to programming? (interviewer should explain the nature of visualisation tool)</p> <p>P20: No</p> <p>TASK2: Evaluated questions for each tool(25 minutes)</p> <p>Interviewer:’ What are the strengthen(s) and weakness (es) of the Jeliot\online python tutor\visual logic tools ?</p> <p><u>Jeliot:</u> ‘Strengthen : the error explanation, the tracing very helpful Weakness: the animation needs more control ‘</p> <p><u>Online python tutor:</u> ‘Strengthen : it is easy to use, the interface not complicated Weakness: how to control the execution was not good, the error explanation , online mode’</p> <p><u>Visual logic:</u> ‘It good to understand the logic of programming but not for writing code ‘</p> <p>Task3:Comparison between the tools on solving three kind of activities (LOOPS- Objects- Passing parameters) (10 minutes) The discussion will be held for each activity on the effectiveness of the tool</p> <p>Interviewer:Which tool do you prefer for solving the LOOP activity ? Describe why? P20: Jeliot because the loop counter was clear and the change of the body loop also easy to see the values of variables.</p> <p>Interviewer:Which tool do you prefer for solving the OBJECT activity ? Describe why? P20: Online python tutor because the hierarchy of inheritance was</p>	
---	--

<p>sequential</p> <p>Interviewer: Which tool do you prefer for solving the PASSING_PARAMETER activity? Describe why?</p> <p>P20: Both, I think it is clear in both tools and any representation I can understand it</p>	
---	--

Table: final coding framework

final coding framework	initial coding framework
<p>Online or offline</p> <p>Error explanation</p> <p>Expression evaluation</p> <p>Programming languages</p> <p>Control execution</p> <p>Hierarchy of classes</p> <p>Saving history</p> <p>Interface</p>	<p>Tools availability</p> <p>Debugging and error explanation Correction suggestion</p> <p>Expression evaluation preference Trace the program code Animation</p> <p>Programming languages preferences</p> <p>Execution preferences Animation Use of control buttons Execution speed</p> <p>Object oriented representation</p> <p>Save action</p> <p>Windows preferences Statement execution indicator Interface preferences Code font Ease of use</p>

F-2 Interviews with Experts in the Evaluation Phase

Interview- transcript

Section1:

Q1: Did you teach Programming language(s), if yes proceed to the following questions
How long have you been teaching programming ?

Q2: Which level or course of programming do you teach?

Q:3 Which programming language do you teach?

Q4: What is your experience (if any) on using visualization tools to improve your teaching for programming courses?(interviewer should explain the nature of visualization tool)

If yes, the following questions will be asked:

Q4-1 What is the tool? (describe how it works and its method)

Q4-2 How often you use it?

Q4-3 What are the tool features?

Q4-4 What the aspects you like/dislike on the tool?

Q4-5 To what extent it was helpful?

Section2:

Q5: What are the strength(s) and weakness(es) of the tool?

Q6 :Will you use it if possible? Explain why or why not

Q7: Do you think it is beneficial and may improve programming learning?

Q8: What should other types of improvements be implemented to build on this tool?

Q9: How did you find the method representation ? things you like and dislike

Q10:How did you find the variable representation ? things you like and dislike

Q11:How did you find the expression of evaluation ? things you like and dislike

Q12:How did you find the passing parameters and calling method representation ? things you like and dislike

Q13: How did you find the class representation ? things you like and dislike

Q14:How did you find the object representation ? things you like and dislike

Q15:How did you find the class inheritance representation ? things you like and dislike

Q16:How did you find the output representation ? things you like and dislike

Q17:How did you find the animation? things you like and dislike

Q18:How did you find the control of execution ? things you like and dislike

Table: initial coding framework

Interview transcript	Initial coding framework
<p>Expert1: Interviewer: Q1: yes, for 7 years Q2: programming I and II Q3: Java Q4: NONE Q4-1: -- Q4-2: Q4-3: Q4-4: Q4-5: Q5: The tool is good method for visualisation and tracing the code , it is very clear and easy to use, I think it will be very helpful Using different shapes and styles to represent variables, objects, and classes will help the students to distinguish the difference between them. The tool not flexible , the students cannot plugged in their own code It has limited examples Q6: Yes, It will save time and effort Q7: yes, I am expecting that it will increase the students' understanding Q8: more examples and support error debugging Q9: Helps to understand that sending and receiving parameters have different memory location. Q10: very good Q11: clear, I like the transition of values of variables and how it represents the results Q12: clear, I like the transition of the parameters in and to the methods Q13: Using different shapes and styles to represent objects, and classes will help the students to distinguish the difference between them. Q14: same answer for Q13 Q15: I like using such a diagram similar to UML to show what the objects belongs to superclass Q16: similar to reality , as it happen in programming applications Q17: good but I suggested to control the speed of animation or skip some of them Q18: good and suitable</p> <p>Expert2: Interviewer: Q1: yes, for 10 years Q2: programming I and II Q3: Java, C, C++ Q4: yes Q4-1: Jeliot Q4-2: I Used it once Q4-3: visualisation</p>	<p>Animation Ease of use The three problems representation Object oriented representation</p> <p>Animation visualisation</p>

<p>Q4-4: the control of execution- visualisation- array examples Q4-5: good but need some practicing to fully understand it Q5: The tool is good method for visualisation and tracing the code . The structure and branching that have been used in the memory frame will help in understand which item is belong to what method or class , for example the student can recognise that variable (width) and variable (length) belongs to the method (main) which belongs to class (parameterExample) also the objects (test) is an object from class (MathsLevel1) where object (example) is an object from class (MathsLevel2). The tool need more examples such as the use of array, pointers, and recursion Q6: I would like to use the visualisation as a supplement in lectures or even before starting the lab activity to improve the student’s comprehension Q7: yes, I am expecting that it will increase the students’ understanding Q8: more examples Q9: Helpful Q10: accurate and easy to understand Q11: helpful Q12: I like how you spate the variables in the method header and method calling for two distinct memory places Q13: I like the different styles to distinguish between classes and objects Q14: same answer for Q13 Q15: very good, it shows what variables and methods that should be inherited and not inherited Q16: very clear because it shows in separate window Q17: good and easy to use Q18: I liked the two different mode (options) for animation</p> <p>Expert3: Interviewer: Q1: yes, for 15 years Q2: programming I and II Q3: Java, .net Q4: yes Q4-1: Jeliot Q4-2: I read a paper about jeliot and I decided to try it Q4-3: visualisation Q4-4: I like the whole idea and the concepts of the tool Q4-5: good as self- training tool Q5: The tool is good method for visualisation and tracing the code . - I like the highlighting of every single statement before, and during the execution, that will show the flow of the code and give enough time for the student to understand what happens during execution. - I like the windows, the colours, the boxes, arrows, and the animation buttons The tool need more examples such as the use of array. Q6: it could be used as extra supplement and tutorial or self-learning method Q7: yes Q8: the user should try his own code Q9: I like using the lines to represent what method belongs to what class</p>	<p>Ease of use The three problems representation Object oriented representation</p> <p>Animation Ease of use The three problems representation Object oriented representation</p>
--	--

<p>Q4-3: Q4-4: Q4-5: Q5: The tool is good method for tracing the code similar to what we do in class on board or on paper - I like the colour of the windows but font need to be larger The tool need more programming codes and examples Q6: yes or suggestion to be use at home as self-learning Q7: yes Q8: large font Q9: it is clear what methods belongs to what class Q10: good Q11: easy to understand Q12: Showing how the parameters travel from where it is calling to memory and finally to their place in the header Q13: Students now become able to know the difference between object and class. Q14: great Q15: I like the representation and it could be represented like UML diagrams Q16: similar to real executions and final output window Q17: good Q18: good Expert 6: Interviewer: Q1: yes, for 17 years Q2: programming I and II Q3: Java,Pascal ,C,C++, Python Q4: yes Q4-1: online python tutor Q4-2: I used it during my learning python language Q4-3: tracing, representing linked list and arrays, classes and methods calling, error findings and explanation Q4-4: same as the features mentioned above Q4-5: very helpful Q5: good tool that can be used during the lecture to explain the lesson , and It is easy to use and doesn't need any training sessions so the students can use it as self-learning The tool need more programming codes and examples , I prefer representing recursion, arrays, pointers and control statements Q6: yes or suggestion to be use at home as self-learning Q7: yes Q8: The tool need more programming codes and examples as mentioned in Q5 Q9: clear Q10: good and easy to understand Q11: easy to understand Q12: help to understand the calling by value Q13: good Q14: great Q15: Showing how the attributes inherited from the superclass by using drawing and animation will add support for me when teaching the class inheritance lesson since it always confuses the students</p>	<p>Execution preference Interface preference</p> <p>Animation Interface Ease of use Object oriented programming (inheritance)</p>
---	---

Q16: suitable and clear	
-------------------------	--

Q17: I like it	
----------------	--

Q18: I like it	
----------------	--

F-3 Focus Group Discussion with The Students In The Evaluation Phase

Focus Group Transcript

What are the strengthen(s) and weakness (es) of the tool in general?

Discussion include other questions such as

How did you find the output representation ? things you like and dislike ?

How did you find the animation? things you like and dislike ?

How did you find the control of execution ? things you like and dislike ?

Interface – usefulness- control of the execution -animation
--

A.2.1 “The portions of the window was clear.”

C.3.4 “The position of each window has to make sense.”

C.1.7 “Colors were not distracting me.”

B.1.1 “Better than using the manual trace “

C.1.6 “Prefer the tracing that shows the steps of code flow.”

C.1.2 “Great for tracing and having an application for the tool would be easier for students to use rather than a website.”

B.5.6 “I would like to use this method after finishing each chapter.”

A.3.2 “Very useful.”

A.5.1 “I like the idea of the tool, and I think it will help me to learn Java.”

B.1.5 “I may use the manual trace and then refer to the tool to make sure that my answer is correct.”

A.5.2 “Make easier to understand and suggest to add visual examples of how the method can be used to program certain things like video games.”

B.3.3 “Control buttons were clear and easy to understand.”

C.1.4 “The motion of going step by step was suitable.”

B.2.6 “Prefer to control my tracing.”

How did you find the method representation ? things you like and dislike ?

Animation

A.1.2 “Before, I was thinking that changing the values of method’s parameters inside the method will take effect on the values of variables in the main method.”

A.4.5 “I thought that the variables that have been passed to the method have the same memory location for the method’s parameters themselves”

A.2.3 “ now I understand why the values not changed in the main”

A.2.1 “ I like how the tool represent each method as part of class using lines and menu”

How did you find the variable representation ? things you like and dislike ?

Interface- Animation

B.5.6 “ I like boxes to represent the name of the variable and the value”

C.2.3 “ the variable representation similar to what we do during the class on board”

How did you find the expression of evaluation ? things you like and dislike ?

Interface - expression evaluation

A.3.4 “Helps me to identify the value of the area “

B.4.5 “Understand the calculated balance.”

How did you find the passing parameters and calling method representation ? things you like and dislike ?

Animation – interface

A.1.2 “Before, I was thinking that changing the values of method’s parameters inside the method will take effect on the values of variables in the main method.”

A.4.5 “I thought that the variables that have been passed to the method have the same memory location for the method’s parameters themselves”

How did you find the class/object representation ? things you like and dislike ?

Animation – interface – class and object representation – class constructor

B.4.1 “It makes me understand the concept of classes and objects because I misunderstood them.”

B.4.4 “I knew the concept of the class as abstract, but now I understand how it works. Moreover, I understand how we can define more than one object from the same class, but each object has different memory locations so the changing in one attribute for one object will not affect the same attribute in the other objects. “

B.2.3 “ I was thinking that class constructor is a method that should be invoked same as any other method, the step by step tracing for the code makes me understand that class constructor is automatically invoked when creating the object.”

B.3.1 “Similar to UML representation.”

B.4.6 “Helps me know the difference between class and object.”

B.3.2 “Like the representation of data inside object.”

B.2.1 “Like the referring exists that relate the class to its objects.”

How did you find the class inheritance representation ? things you like and dislike ?

Animation – interface – representation of inheritance

C.2.1 “Like the transfer of the inherited object to the memory location.”

C.4.3 “Using shapes to represent inheritance with (box) inside (box) makes me understand what attributes are should be in the subclass.”

C.1.6 “Before, I thought that any change happens inside any inherited variable or method should be done to every copy exist in any other object for the same variable or method.”

would you use the tool if it's possible? Explain why or why not

B.4.4 “ yes as self-learning”

C.3.1 “ yes, as practice after the lecture”

A.1.3 “ yes, during the lab activity”

Do you think it is beneficial and may improve programming learning?

A.1.4 “ yes as self-learning”

B.5.1 “ yes, I think my grades will improve”

C.2.3 “ yes, especially if the instructor use it during the lecture”

What should other types of improvements be implemented to build on this tool?

A.2.5 “ control the animation speed and have more examples”

B.4.4 “ I need to bulged my own code to test the code correctness and output”

C.1.2 “ more code and examples such as using iteration and pointers ”