VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Operators and precedence in programming language design

Najwani Razali

Supervisors: James Noble, Stuart Marshall

Submitted in partial fulfilment of the requirements for
Proposal Defence.

Abstract

This study will address the issues of operators and precedence in programming languages. It is very important to resolve this issue because incorrect understanding of operators and precedence concepts can create a security hole in programming. The purpose of this study is to explore users understanding, interpretation and perception of operators and precedence issues. Data for this study will be collected from students, programmers and the Qualitas Corpus. Data collection will involve questionnaires, think-aloud methods and semi-structured interviews. The findings may lead to new knowledge about the rules for operators and precedence in programming languages. The findings will also provide both theoretical analysis and guidelines for developers and language designers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Programming language design has evolved over time and many designers aim to create simple, well-defined and consistent languages [9]. It is not easy however, to design a good programming language which is understandable to others. Some new programming languages are introduced for specific domains such as the Grace language, which focuses on novice programmers [11].

Programming language can be divided into two main parts: syntax (or structure) and semantics (or meaning) [38, 59, 32, 13]. An important syntactic consideration in many programming languages is operator precedence. Operator precedence defines order of evaluation in any expression including arithmetic, bitwise, Boolean and other operators. The precedence rule for arithmetic (+,-,*,/), logical (&&) or other operators varies with programming languages [28]. In most programming languages, a table of rules applies to all operators. For example, the expression *13 + 5 * 2* has the value 23 if the expression is evaluated using the conventional order of evaluation in mathematics however, it has a value of 36 if the expression is evaluated from left to right.

As reported in several studies [50, 51, 19, 31, 52, 55] errors in operator precedence among students happened while learning a programming language, for example they are likely to interpret the *NOT* operator with lower precedence than the other boolean operators. This interpretation is the opposite of precedence rules in most programming languages. An incorrect understanding of operator precedence can also create security holes in programs. For example, in 2012, the Knight Capital Group (KCG) lost 440 million in the first 30 minutes of their business day. They suspect this loss might have been caused by a mistake in operator precedence in their latency cost equation [2].

1.1 Motivation

Many researchers have studied programming language design in various contexts. Several studies have been published on static and dynamic typing, developing tools for debugging errors, mistakes among students in programming and improvements in programming languages [47, 41, 8, 20, 80]. Unfortunately, the discussion related to features in programming

languages rarely consider operator precedence issues. Some researchers have conducted their research study on common mistakes among students [50, 51, 19, 31, 52, 55] and one of the mistakes was operator precedence, yet, not many people seem to considers operator precedence as major problem.

Pane and his colleagues [51] suggested that further research is needed to determine how languages should deal with operators and precedence issues. In addition, Derek [30] also studied whether precedence define by the language has any effect on developers' performance or not. Therefore, although many studies have been published related to programming language design, there is a lack of research, particularly empirical studies, on operators and precedence.

1.2 Research objectives

The aim is to study operators and precedence in programming language design, which is guided by four research questions (RQ):

- RQ1: Do students have problems with operators and precedence in Java?
- RQ2: How do students understand precedence and parentheses in Java?
- RQ3: How do programmers use precedence and parentheses in Java?
- RQ4: What operator precedence and parentheses rules should be used in Java-like languages?

1.3 Research contributions

At the end of this research, three main contributions are expected:

- Understanding students' knowledge of operator precedence in programming language syntax
- Analysis of the Qualitas Corpus of Java programs to determine how operators and precedence are used across multiple Java programs.
- A set of guidelines for designing or improving operator precedence in programming languages.

1.4 Organisation of proposal

This research proposal is organized as follows. Chapter 2 presents some background and a literature review related to programming language design. Chapter 3 describes the methodology of this research study. Chapter 4 provides a detailed description of the pilot study and some other preliminary results. Finally, Chapter 5 provides a detailed research plan for completing the thesis.

Chapter 2

Literature review

This chapter reviews the literature on programming language design, especially syntax and semantics. This chapter discusses different types of programming language and explains the differences in operator precedence rules.

2.1 Overview of language design, syntax and semantics

Syntax and semantics are important components of programming languages [77, 65, 79]. For example, in C syntax, the not equal operator is represented as `!=` while in Pascal syntax it is represented as <> but in both it has the same semantics disequality. Syntax is the form of expressions and statements while semantics are the meaning of the expressions and statements [38]. An expression contains operands and operators which produce a value of evaluation of the expressions. A complete expression will be executed following specific rules specified by the language design. Therefore, it is necessary to know the order of evaluation to understand the meaning of the expression.

The order of evaluation depends upon precedence rules. In programming, expressions can be more complex; therefore the use of precedence rules together with parentheses are vital. In any programming languages syntax acts as the user interface for programmers. For example, access of the ith element of an array which start from address x, whose elements are y words long, requires the computation of $x + (i - 1)y$ but is typically written *x[i]*, *x(i)* or *x!i* .

Operators in expressions can be written in three different notations which are: infix, prefix and postfix notation.

- Infix notation: In this notation, operators are placed between operands. For example, when writing the expression such as `x + y * z`.
- Prefix notation: This notation is preceded by operator and followed by operands. For example, to write the sum of `x` and `y` , the expression should be `(+ x, y)`.
- Postfix notation: The operator is placed after the operands in the expression. It is the reverse of prefix notation; `(x, y +)`.

Neither prefix nor postfix notation requires parentheses in the expressions. The order of evaluation is clear and it is not necessary to put parentheses to declare the order of evaluation. In contrast, infix notation requires parentheses to be used in the expression depending on certain conditions. Parentheses can be used to prioritize operators in an expression when needed. Therefore, decisions to use parentheses in expressions need to be clarified in order to remove ambiguity and confusion among users.

2.1.1 Criteria in designing programming language features

Sebesta [67] offers three main criteria in designing a programming languages: readability, writability and reliability.

Readability is the ease with which reader can read and understand the code/program

Writability is the measure of how easy the language can be used to create a program

Reliability is the program perform to its specification under all conditions

These three criteria can be affected by several characteristics which is shown in table 2.1.

	CRITERIA		
Charactersitcs	Readability	Writability	Reliability
Simplicity	•	•	•
Orthogonality	•	•	•
Control structures	•	•	•
Data types and structures	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Table 2.1: Language evaluation criteria and characteristic that affect them [67]

These criteria have been used and implemented for evaluating an appropriate language for certain applications [64]. Readability and writability are important in designing programming language syntax. For instance, the use of parentheses or spaces can be a problem for some syntactic features (i.e. LISP). It is not easy for language designers to make the right decisions when selecting the features based on the criteria however, it is vital for them to follow the criteria [57]. For example, forcing the use of parentheses can create complexity for some programmers and users. Thus, decisions can be made after testing the problem according to the criteria [70, 5].

2.2 Different types of programming language and their precedence

This section presents a selection of programming languages and their precedence rules alphabetically. Each programming language has its own precedence table. Each operator has its own level of precedence. The operators that have the same level of precedence will be evaluated according to associativity rules (left-to-right or right-to-left). The evaluation of prefix and postfix are done by stack rules.

In postfix, the evaluation example is discussed using the `21*6+2/`. The postfix notation does not depend on the precedence rules. The table below shows how the expression is evaluated based on a stack.

Input	Stack	Computation performed
2	2	-
1	2,1	-
*	2	2*1=2
6	2,6	-
+	8	2+6=8
2	8,2	-
/	4	8/2=4

Table 2.2: Postfix expression evaluated based on stack

Prefix notation also does not typically depend on precedence rules. The example below shows how to evaluate a prefix notation expression. The expression is `*2+55`.

Input	Stack	Computation performed
*	*	-
2	*,2	-
+	*,2,+	-
5	*,2,+,5	-
5	*,2,+,5,5	5+5=10
-	*,2,10	2*10=20
-	20	-

Table 2.3: Prefix expression evaluated based on stack

Infix notation needs the operators to be grouped or apply precedence rules in order to remove ambiguity. Infix expressions have ambiguity issues, for example between multiplication and addition operators, in the order of preference. Therefore, precedence rules need to be applied to infix expressions while performing the calculation.

The next section presents several different programming languages and their precedence rules.

2.2.1 C

The C language is widely used in real world. Table 2.4 summarizes the precedence rules for operators in C. All the operators are grouped together according to their precedence.

Operators	Precedence
(), [], ->	Highest
!, ~, ++, --, \&, *, unary +, unary -, (type), sizeof	-
*, /, \%	-
+, -	-
<<, >>	-
<, <=, >, >=	-
==, !=	-
\&	-
^	-
|	-
\&\&	-
||	-
?:	-
=, +=, -=, *=, /=, \%=, \&=, ^=, |=, <<=, >>=	-
,	Lowest

Table 2.4: Precedence rules of C

2.2.2 FORTRAN

FORTRAN was one of the earliest programming languages. Like other languages, FORTRAN also has precedence rules. The operators and rules are simpler compared to other languages [78].

Operators	Precedence
** (exponential)	Highest
*, / (multiplication, division)	-
+, – (addition, subtraction)	Lowest

Table 2.5: Precedence rules of FORTRAN

2.2.3 FORTRESS

Unlike other programming languages, FORTRESS precedence rules do not rely to operators. The rule relies on the defining groups of operators based on their meaning and shape and also based on specific relationships between the groups. Parentheses are required when there is no specific relationship between operators. The precedence rules in FORTRESS are not as straightforward as those in other programming languages. The general rules are described below according to the FORTRESS Language Specification:

Basic principles	Precedence
Member selection (.) and method invocation (`.name(...)`) are not operators, yet they have the highest precedence	Highest
Subscripting `[ ]` and any kind of subscripting operators, which can be any kind of enclosing operators), superscripting (^) and postfix operators	-
Tight juxtaposition, without intervening whitespace	-
Tight fractions that is the use of `/` with no space on either side	-
Loose juxtaposition with intervening whitespace	-
Prefix operators	-
Infix operators	-
The equal symbol "=", the assignment operator "`:=`" and compound assignments operators (`+=, -=` etc)	Lowest

Table 2.6: Basic precedence rules of FORTRESS [6]

2.2.4 Grace

Grace is a new object-oriented languages for novices [11]. Most Grace operators have the same precedence, but an expression with two different operators needs to be parenthesized to indicate the order of evaluation.

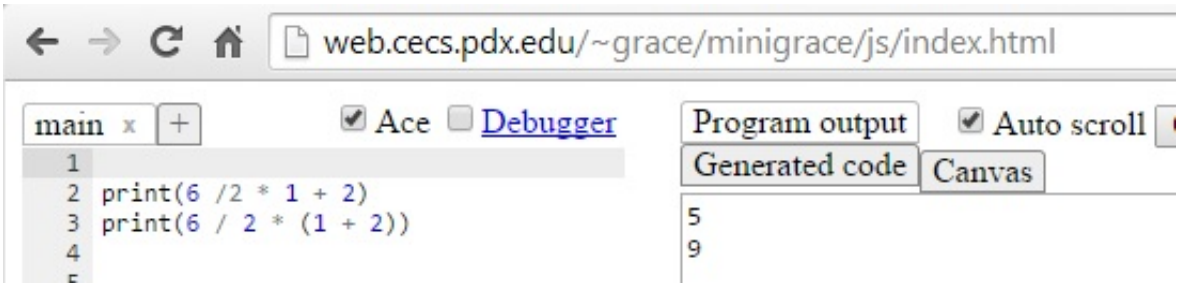

Figure 2.1: An example of arithmetic expression in Grace language

2.2.5 Java

The table below represents Java's operator precedence. The operators in the table below are listed according to the precedence order. The same operators at one line share similar level of precedence, yet the evaluation will be based on associativity rules [24].

Operators	Precedence
Postfix increments & decrements (e.g. x++)	1
Prefix increments & decrements (e.g. ++x)	2
Unary positive (+x), unary negative (-x) & logical negation (!x)	3
Binary arithmetic operators (e.g. multiply, divide, modulus)	4
Binary comparison operators (e.g. <, >, <=, ==, !=)	5
Binary logical operators (e.g. AND & OR)	6
Assignment operators	7

Table 2.7: Operator precedence table; from top to bottom is highest to lowest precedence order

2.2.6 LISP

LISP does not have operator precedence like other languages, rather the order of evaluation in any expression will be based on parentheses.The language requires parentheses in any expression which has more than one operator and it is written as a prefix expression (figure 2.2). Therefore, the order of evaluation will be read from left to right.

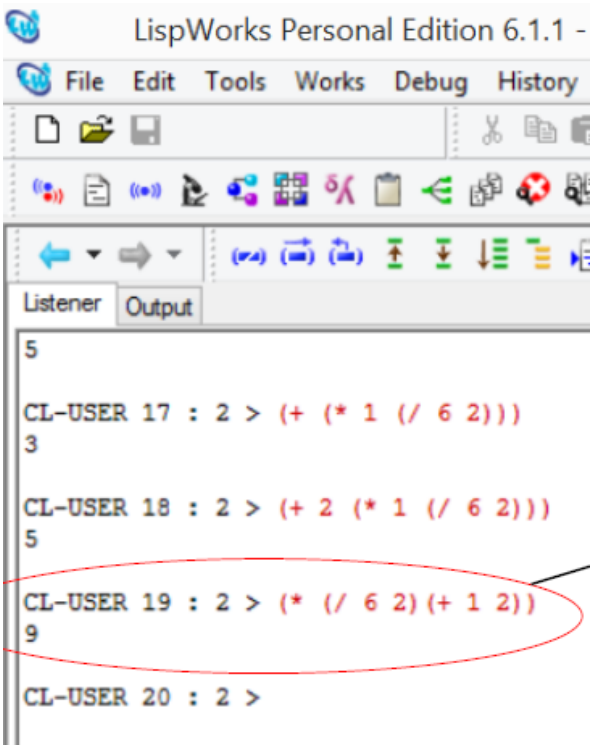

Figure 2.2: An example of arithmetic expression in LISP language

In figure 2.2, the arguments of the function are evaluated before the function is executed. In the example, argument (+1 2) is evaluated to give 3. Then the second argument is evaluated to give 3. And finally the last function is applied to 3 to give 9 as the result. All expression evaluation in LISP is evaluated from right to left and also requires parentheses

for each argument.

2.2.7 Pascal

Operator precedence in Pascal is much simpler than Java or C. Table 2.8 and the Figure 2.3 show examples of Pascal expression ċiteborland.

Operators	Precedence
`not`	highest
`* / div mod and`	-
`+ - or`	-
`= <> < <= > >= in`	lowest

Table 2.8: Operator precedence table for Pascal language

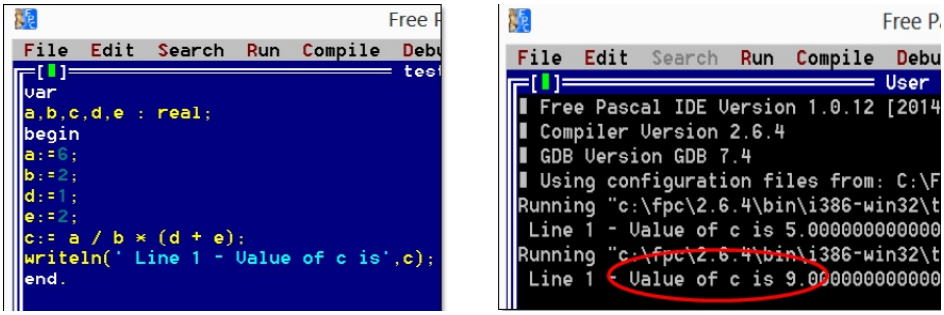

Figure 2.3: An example of arithmetic expression in Pascal language

In Pascal, users need to be more careful with the precedence issue, especially with the logical conjunction (`and`) operator. When writing a conditional expression such as if `a < b and c < d` without parentheses, the compiler will produce an error. This error caused by the `and` operator which has higher precedence and should also be a boolean values. Therefore, the expression should use a parentheses such as `(a < b) and (c < d)`.

2.2.8 Pyret

Pyret is also another example which requires parentheses in any expression which has more than one operator but unlike LISP, PYRET is infix notation. In order to indicate the order of evaluation, an error will be prompted when two different operators are in an expression without parentheses [34].

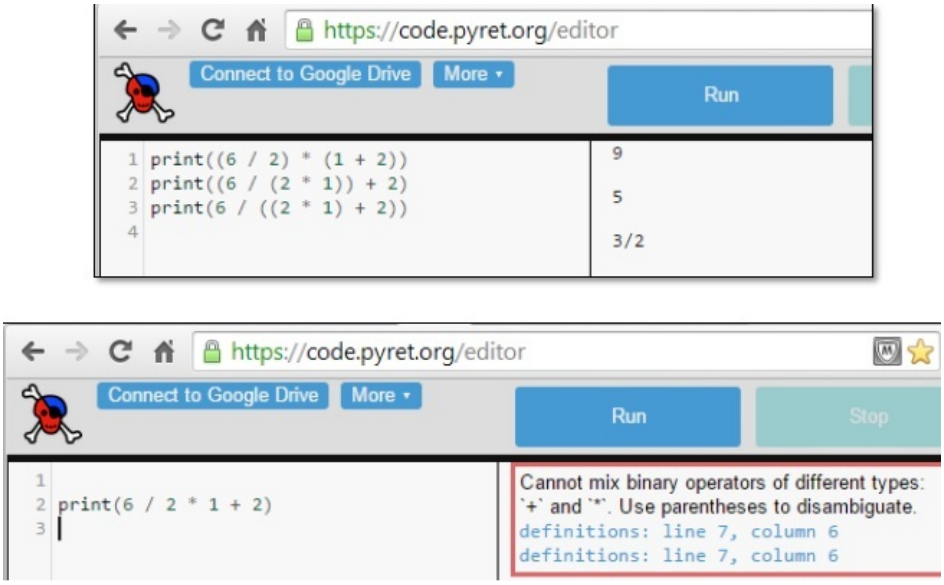

Figure 2.4: An example of arithmetic expression in Pyret language

2.2.9 Smalltalk

In Smalltalk, operator precedence is based on order of evaluation, organized as below:

- 1. Parentheses
- 2. Unary message (left to right)
- 3. Binary message (left to right)
- 4. Keyword message sends
- 5. Assignments
- 6. Returns

The figure below shows the example of arithmetic expressions in Smalltalk.

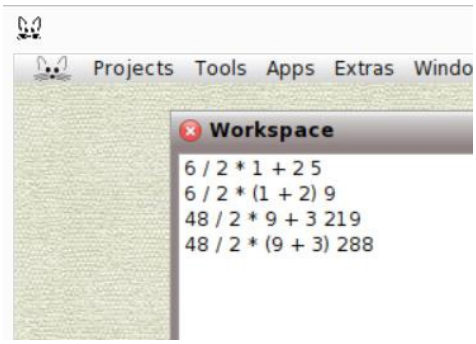

Figure 2.5: An example of arithmetic expression in Smalltalk language

In Smalltalk, operators have no specific precedence unlike other programming languages. For example, multiplication has no priority over addition. The use of parentheses can override the precedence of other expressions types. For example, `3 + 4 * 5` , is evaluated as two binary messages which produces `35` as the result, not `23`. That is because there is no priority for multiplication and addition and the expression is evalauted from left-to-right [10].

2.2.10 Summary

Different programming languages apply different precedence rules and notations, yet sometimes they share similar rules. Several programming languages such as Java, C, Pascal etc use infix notation while some languages use prefix notation, eg. LISP. The use of different notations can affect the order of evaluation. In general, the expressions in prefix or postfix notations have no ambiguity issues regarding the order of evaluation. The table below summarizes an example showing the precedence for the evaluation of an expression when written in different languages.

Programming language	Expressions	Output	Rules

Java, Grace, Pascal	`6 / 2 * (1 + 2)`
	`6 / 2 * 1 + 2`
	`2 + 6 / 2 + 1`
		9
		5
		5
			Precedence

Smalltalk	`6 / 2 * (1 + 2)`
	`6 / 2 * 1 + 2`
	`2 + 6 / 2 * 1`
		9
		5
		4
			Left-to-right

LISP	`(* (/ 6 2)(+ 1 2))` `(/ 6 2(+ 1 2))`	9 1	Parentheses & prefix
Pyret	`((6 / 2) * (1 + 2))`	9	Parentheses

Table 2.9: Summary of different programming language related to operator precedence issue

2.3 Real world examples of precedence problems

Bitwise operator precedence bugs

Operator precedence is important in complex expressions or calculations. The linux-kernel mailing list [12] discusses real bugs in the linux kernel. These problems have been reported in archives and discussed by other researchers and programmers. One example shown below:

```
*delay = (u160) (val & 140E_PRTDCB_GENC_PFCLDA_MASK >>
140E_PRTDCB_GENC_PFCLDA_SHIFT);
```

In C, the `&` operator has lower precedence than the >> operator, so the expression was not executed as to programmer intended. The programmer intend to shift both arguments (`val & 140E_PRTDCB_GENC_PFCLDA_MASK`) to the `140E_PRTDCB_GENC_PFCLDA_SHIFT`.

However, the code is wrong because the shift >> operator has higher precedence than & operator, so the code only shift the mask. Therefore, it is a must for programmer to put a parentheses around the & subexpression.
The correct expression is:

```
*delay = (val & 140E_PRTDCB_GENC_PFCLDA_MASK) >>
140E_PRTDCB_GENC_PFCLDA_SHIFT;
```

Parentheses around both arguments will let the program know the order of evaluation for the code.

Arithmetic operators

The Mitre organisation [1] also collects and publishes errors of operator precedence along with other programmers errors. An example shows a simple mistake calculating return on investment. In Java, the / operator has higher precedence than the – operator, therefore, without parentheses the code below is wrong because the divide operator is executed before the minus operator

```
public double calculateReturnOnInvestment(double currentValue,
double initialInvestment)
{
   double returnROI = 0.0;
   // calculate return on investment
   returnROI = currentValue – initialInvestment / initialInvestment;
   return returnROI;
}
```

The basic calculation of returnROI is correct however, it requires parentheses in order to prioritize the execution of operators. It is correctly calculated below:

```
returnROI = (currentValue – initialInvestment) / initialInvestment;
```

Parentheses are important to produce correct calculations. In real life business situations, this small logic error can create a big loss to the company profit.

Bitwise operator precedence bugs

```
# include ``stdio.h''
# include ``stdlib.h''
int main(){
int a = atoi(``10'');
int b = atoi(``20'');
int c = atoi(``30'');
printf(``\%''u\backslash n'', a | b <= c);
```

```
return 0;
}
```

In D programming language forum [3], an issue of bugs caused by bitwise operator has been reported. Bitwise operators have low precedence so, they are error-prone in C/C++/D languages. It is error because the relational operator <= is executing with | operator after checking the condition of `b | c`. The expression should perform bitwise operator | before executes relational operator <=.
The solution is suggested to provide extra parentheses around the comparison operator "|" [3].

```
# include ``stdio.h''
# include ``stdlib.h''
int main(){
int a = atoi(``10'');
int b = atoi(``20'');
int c = atoi(``30'');
printf(``\%''u\backslash n'', (a | b) <= c);
return 0;
}
```

Chapter 3

Research methodology

This chapter describes the mixed methods research methodology which will be used in this study. The first section provides an overview of mixed methods research design and how it can be applied to this study. The later sections deal with the details of data collection and data analysis.

3.1 Philosophical worldviews

When planning a research study, researchers need to think in detail about three main components, viz., philosophical *worldview*, research *designs* and also research *methods* (Figure 3.1) [16]. We use Creswell's [16] term *worldview* to describe the philosophy of research while others use a different term: *paradigm* [37, 42].

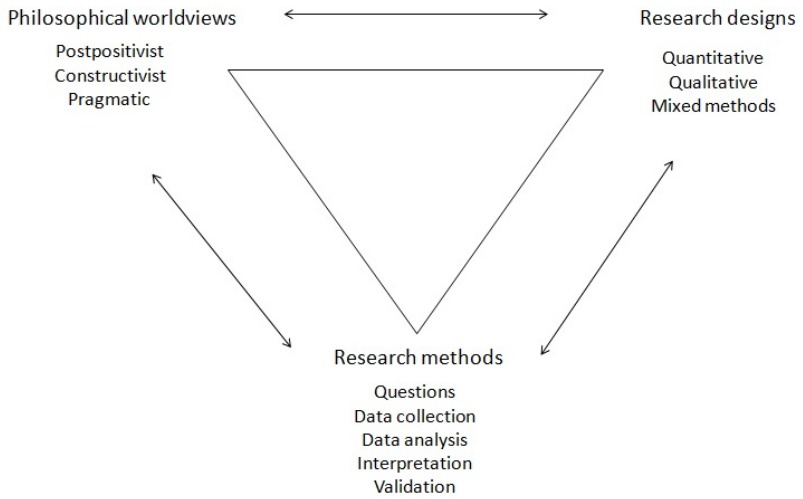

Figure 3.1: A framework of research approach [16]

There are three common worldviews which have been adopted in software engineering research [16, 21, 46, 26]:

- Postpositivism: Postpositivists study the behaviour of individuals and develop numerical measurement of observations [16]. Knowledge is discovered and verified through direct observation and measurements of the phenomena [33].This worldview begins with a theory, collects data to either support or reject the theory and then makes decisions about the theory.
- Constructivism: This paradigm describes knowledge as a reality that humans understand through their life experience and actions. Constructivism attempts to understand human perception, emotion and opinion within their social environment. Constructivism focusses less on theory verification and more on understanding how humans react to certain phenomena. Crotty [17] assumed that human beings construct ideas or meanings by interpreting their actions while engaged to the real situation. Constructivists also create more interest in qualitative research which gathers data about human activities in relations to real situation.
- Pragmatism: This paradigm comes out of actions, situations, and consequences. Pragmatism does not focus on methods, but emphasizes the research problem. Pragmatism uses all approaches or attempts to use the available knowledge for a certain practical problem as long as it is useful in order to understand the problem [61]. This worldview does not set out to criticize or verify existing knowledge. Researchers are free to use any available methods or knowledge to shed light on their research problem [16]. This paradigm underpins much mixed methods research where it is vital to focus on the research problem for deriving knowledge about the problem [72, 45, 54].

This study is not based on any existing theory. Therefore, a postpositivist worldviews is not suitable for this study. Due to the lack of the theory, this study will be guided by the four research questions present include:

- RQ1: Do students have problems with operators and precedence in Java?
- RQ2: How do students understand precedence and parentheses in Java?
- RQ3: How do programmers use precedence and parentheses in Java?
- RQ4: What operator precedence and parentheses rules should be used in Java-like languages?

The research questions focus on the problem in the human context by asking 'what' and 'how'. In order to understanding the research problem, this study requires the researchers to learn from users' experience and responses to certain phenomena. All data will be gathered and interpreted in relation to the research problem. For example, we aim to understand how a student uses operator precedence in programming language and also, how an expert uses precedence in his/her coding. This research will investigate and interpret users' experience and precedence data in a corpus in order to determine the findings on operator precedence in programming languages. Thus, this study will adopt pragmatism and constructivist perspectives.

3.2 Research design

There are three types of research designs viz., quantitative, qualitative and mixed methods research designs (Figure 3.2).

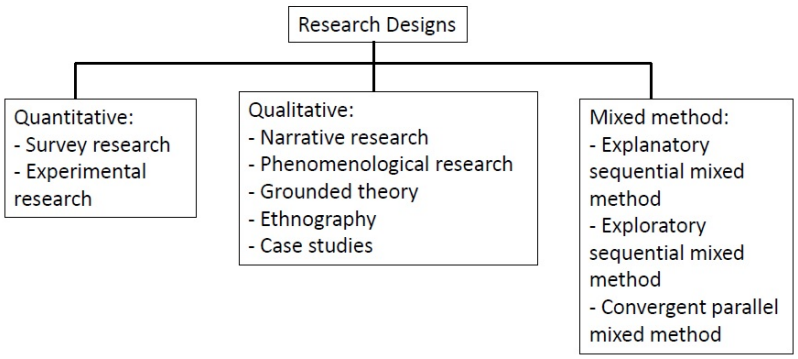

Figure 3.2: Categorization of Research Designs [16]

3.2.1 Quantitative research

Quantitative research is an exploration that explains phenomena by interpreting numerical data [16, 18]. For example, analyzing students answers to questions, catagorizing the answers and analyzing the numerical data. This method is focusing on students, developers andalso corpus thus, this method will be applied to this research study. However, quantitative research has some weaknesses/limitations. Information can be difficult to gather through structured data instruments such as questionnaire, and self-reported information from questionnaires may be incomplete or inaccurate.

3.2.2 Qualitative research

Qualitative methods gather data in the field of study where participants experience the problem. Researchers try to understand the phenomena/problem in the real world [23]. Qualitative research methods have been accepted in software engineering, especially when dealing with human factors [53, 66]. Qualitative methods are relevant in this study because they help to explore phenomena from participants' perspectives e.g. using semi-structured interviews. There are two major weaknesses/limitations from qualitative research. Firstly, knowledge which produced from this research method might not generalize to all people or other situations. Secondly, the results may be influenced by researcher's personal perspectives.

3.2.3 Mixed methods research

The weaknesses/limitations of qualitative and quantitative research requires this study to implement the mixed methods research. Mixed methods research can provide strong evi-

dence for the end results through convergence or divergence of the findings. This method can also produce more complete knowledge which can add insights and understanding which might be missed when using a single method.

Mixed methods research is a combination of quantitative and qualitative methods. According to Onwuegbuzie et.al and Greene et.al [49, 25], there are five purposes of conducting mixed methods research: *Triangulation* (compare quantitative findings with qualitative findings); *Complementarity* (results from one analysis are interpreted to improve the other analysis); *Development* (the data are collected sequentially and the findings are used to inform data collected and analyzed using other analysis types); *Initiation* (identified the contradiction to reframe the research questions); and *Expansion* (expand study's scope and focus using quantitative and qualitative analysis). There are three types of mixed methods study viz., sequential methods (following one method after another) and convergent methods (using the methods in parallel) [16]:

- Explanatory sequential mixed method: Researchers conduct the quantitative research first followed by qualitative research.
- Exploratory sequential mixed method: The reverse of the explanatory sequential method, i.e., begin with a qualitative method followed by quantitative research.
- Convergent parallel mixed method: This method allows the researchers to perform the methods in parallel. Researchers merge the data in order to provide a comprehensive analysis.

This study requires an in-depth understanding and exploration of operators and precedence. Among the five purposes of selecting mixed methods research, two of the purposes are applicable to this study: *triangulation* and *expansion*. *Triangulation* is comparing think-aloud data, semi-structured interviews data and also corpus data. *Expansion* is expanding the findings from quantitative and qualitative studies, which will help in exploring the focus of the problem.

We selected the convergent parallel mixed methods for this study, because the combination of quantitative and qualitative data will cover detailed information about the problem. Different types of information such as detailed views of participants and their pattern of answers will be gathered. The different perspectives of participants' answers and program corpora will be analyzed in this study. Findings from different data can either converge or diverge in the end results, therefore, convergent parallel mixed methods are suitable for this study.

3.2.4 Convergent parallel mixed methods

We plan various methods of data collection. In the beginning of this research, a small pilot study was done among students in the first year Java programming class in University Utara Malaysia's (UUM) Computer Science School. The purpose of the pilot study was to

confirm that there is a problem with operators and precedence. The findings showed several problems occured among students and are discussed in depth in chapter 4. Convergent parallel mixed methods will be applied in the body of the study in order to gather different types of data and merge or diverge the data for the conclusion. Figure 3.3 shows how these research methods will be applied overall.

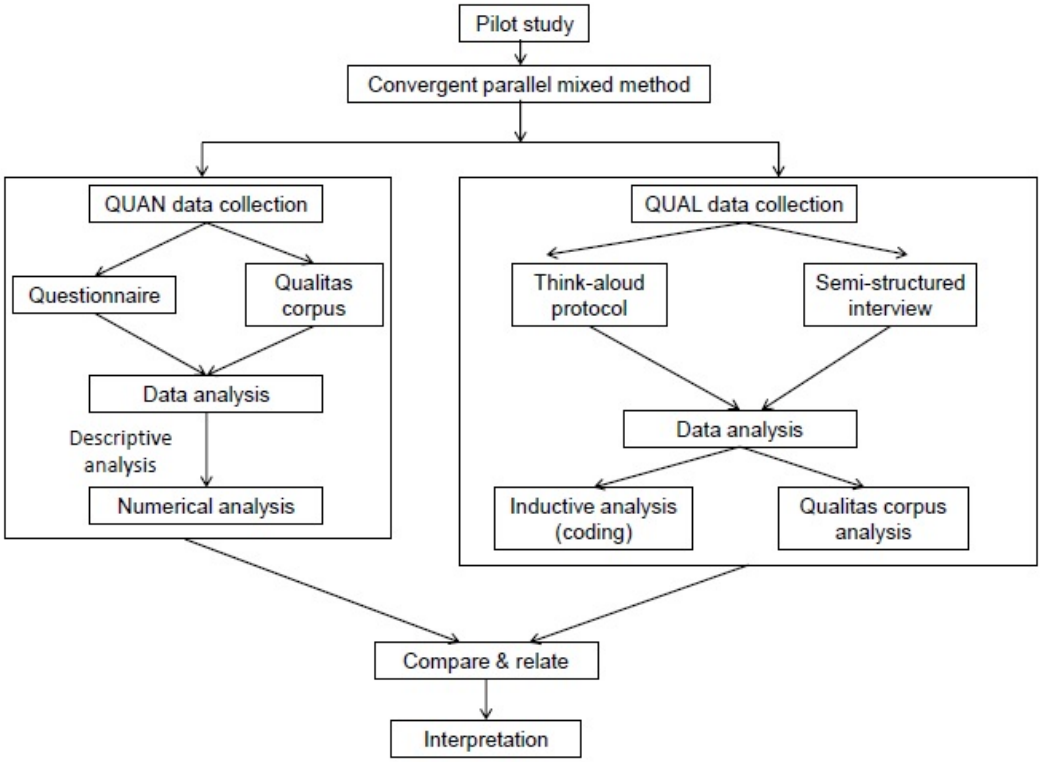

Figure 3.3: A framework of convergent parallel mixed methods; QUAN is quantitative; QUAL is qualitative

3.3 Research methods

3.3.1 Pilot study

A pilot study has been conducted to confirm that there is a problem with operator precedence. This study also gave a general idea about what kind of problems occured among students.

In this pilot study, first year students were asked to answer a set of questions related to operator precedence in Java. This set of questions (Appendix B) was designed based on a previous study [19] and also information and suggestions from an expert. The selection of the questions is covering all topics which is based on the syllabus in their course outline.

There were a total of 16 questions in this test and the participants were required to answer all the questions. Each question was purposely investigating some information regarding the participants' knowledge of operator precedence in Java. The questions included

boolean operators, arithmetic operators etc. The survey test questions were divided according to several criteria of operator precedence in Java, viz., participants' understanding on precedence rules (basic arithmetic and complex expression), interpretation of expressions with or without parentheses, and the side effects of operators. The pilot study is explained in depth in chapter 4.

3.3.2 Data collections

The data collection methods which will be used in this study are questionnaires, think-aloud methods [60], semi-structured interviews and corpus analysis [74]. These methods will gather students' understanding and problem that occured among students in Java. The aim of this data collection is to understand the problem from participants' perspectives.

Participants

The participants in this research are a) students who learn Java and b) programmers or practitioners who had experiences in programming. The selection of participants is not random and it is based on their background. The selection of participants is through programming class lecturers, email lists and developers. The locations of participants are not restricted: it will include participants from New Zealand and Malaysia.

All participants will be required to briefly explain their background knowledge including their level and list of previous subjects taken (for students).

The same participants will be involved in questionnaires, think-aloud protocol and interviews. All three methods will require the same participants in order to triangulate to their answers.

Human ethics approval (HEC)

Dealing with human participants requires the researchers to get approval from the Human Ethics Committee (HEC) of Victoria University (Appendix A). The purpose of this approval is to protect participants' and researchers' rights which include ethical issues, particularly participant confidentiality such as no identification of participants (details of background or the results of tests). We received approval number 0000021215 on 31st October 2014.

3.3.3 Questionnaire

A questionnaire will be given to the participants and the questionnaires are related to operator precedence. The aim of the questionnaire is to gather participants' knowledge and discover how they perceive and interpret operator precedence in programming languages.

Questionnaire data

The questions that will be asked need to be chosen wisely in order to achieve the purpose of the study. The questions are related to participants' background, basic knowledge of operators and precedence, how they interpret the operators and the precedence in Java programs and their solution to problems. The questions may be changed or modified based on the requirements of the problem for further investigation [44]. The questions will take approximately 20 to 40 minutes to answer. The questionnaire is divided into several categories. The selected issues for the questionnaire are described below:

- Checking the use of parentheses in expressions.
- Gathering information on the use of redundant parentheses in the expressions.
- Testing participants' knowledge about complex expressions without any parentheses.

Some questions will include a fragment code of program or a sentence that requires the participants to generate a fragment code.

Participants' answer can be different because, peoples' thoughts can be different from one another. Therefore, a think-aloud protocol will be used for collecting further data.

3.3.4 Think-aloud protocol

Think-aloud protocol is a research method that allows the participants to speak aloud any thoughts in their mind as they worth an actual programming tasks. The researcher ask questions related to the issue such as "what do you think?" or "what is your strategy" in order to remind the participants to express what they think [62, 35].

Think-aloud protocol

The think-aloud protocol will be conducted for approximately an hour depending on the information needed during the discussion. This method is suitable because the researcher can obtain detailed information based on the participants' concerns after they answer the questions [44, 36]. During this session, participants will be asked to explain their thinking as they work. Researchers are allowed to ask any question related to the their answer, ideally without guiding participants.

3.3.5 Semi-structured interviews

Semi-structured interviews also gather qualitative data. This interview is done by setting up the situation which allows participants the time and scope to explain their opinion on particular topics. The focus of these interviews will be decided by the researcher related the participants answers to questionnaire and think aloud protocol. This interview a conversation where the researcher will ask "tell me more about..." or "Earlier you did mentioned this, can you please explain further?". The questions during the interview are not necessarily be

similar for all participants [81, 27, 39].

In this study, the semi-structured interview will be done after the questionnaire and think-aloud protocol. The purpose of this interview is to explore further perceptions and opinions from the participants.

3.3.6 Corpus study

Qualitas corpus

In order to understand how programmers use operators and precedence, the Qualitas corpus will be used in the data collection stage. All code in the corpus will be gathered and categorized according to different metrics such as redundancy, frequency, complexity and consistency of operators and precedence.

Qualitas Corpus is a collection of Java programs which can be used for empirical studies [73]. The Qualitas Corpus provides a compilation of 111 systems of Java projects which contain more than 18 million Lines of Code (LOC), 200,000 compiled classes and 1.5 million compiled methods. Ricardo et.al. [74] have published a compiled version of the Qualitas Corpus known as Qualitas.class Corpus [76]. They contributed to the Qualitas Corpus by compiling and organizing the Java programs including gathering a large amount of metrics data, updating the version, and also fixing a few errors in some projects.

3.4 Data analysis

There are several data analysis methods for mixed methods research such as side-by-side comparison, data transformation, and joint display of data [16]. Data analysis in mixed methods can either be quantitizing (conversion of qualitative data into a numerical code that will be analyzed quantitatively) or qualitizing (conversion of quantitative data into narrative data that will be analyzed qualitatively) [49, 71]. A side-by-side comparison approach quantitizing and qualitizing will be applied including both later.

3.4.1 Inductive analysis

Research studies can be divided into two type: a) deductive (begin with theory or hypothesis) and b) inductive (use research questions to explore problems and narrow the scope of the study) [14]. This study is not based on any theory or hypotheses, therefore an inductive approach will be applied here.

In inductive analysis, researchers begin the study by trying to understand the problem, and allow theory to emerge from the data [75]. An inductive approach is recommended if a study is not based on theory testing [16, 58]. This study is categorized as an inductive approach because the exploration of data is not based on any theme or theory. The main purpose is to explore and discover the themes or theories that emerge from the data. In inductive content analysis, several processes can be applied such as open coding, creation of categories and abstraction [22, 63, 40].

Coding is derived from analytical thinking and the generation of categories and themes. Coding is not only labelling, but linking and leading from data to idea and vice versa [63]. Coding breaks the collected data into pieces and each piece is assigned a label (code) [16, 40]. The coding style can be in several forms such as wording, abbreviations of keywords, numbers etc. The coding data will be extracted from different sources including interviews and documents. The process of categorizing the codes can be done using an approach called concept mapping [29, 40]. Kristin et.al. [29] mentioned that word-based analysis methods and code-based analysis methods for coding process have weaknesses such as bias in coding schemes. The authors propose the implementation of concept mapping as an alternative approach for open-ended survey questions. According to Miles et.al, Pope et.al and Punch [56, 43, 58], there are three procedures use to analyze the qualitative data using any coding approach; *data reduction* (reduce the overlap and redundant data), *data display* and *conclusion drawing or verification*. The procedure of inductive analysis is summarized in Figure 3.4 below.

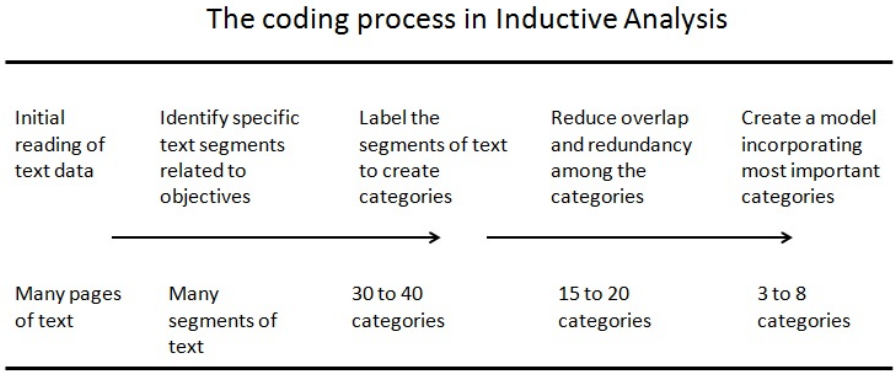

Figure 3.4: The overview of process and procedure of inductive analysis [15, 75].

The process of inductive coding

Researchers begin by reading all the collected text data and transcribe interesting information during the analysis stage. While reading the data, researchers will try to identify any meaningful patterns and create a label for each pattern. At the same time, those labels will be categorized. During the reading and labelling stage, researchers also develop an initial meaning for each category and write it as a memo. The process of inductive coding can be summarized as [75]:

- Gather all the raw data files.
- Do a close reading of the data files for full understanding.
- Create a categorization of the data files. In the inductive process, categorization is done from the actual phrases from the actual and specific data.

- Reduce or remove unrelated, overlapping and uncoded text from the data files.
- Continue revising and refining the categories.

3.4.2 Qualitas Corpus analysis

In order to analyse the Qualitas Corpus, a simple tool will be developed to analyze the large amount of Java source code in Qualitas Corpus. This tool will parse each program in order to detect operators, precedence and parentheses. The analysis will look into how programmers use operators, precedence and parentheses in the program.
This analysis will be based on categorization of metrics for Qualitas Corpus as described below:

- Measuring the redundancy of parentheses.
- Collecting the frequency of usage of operators, precedence and parentheses.
- Categorizing the number of complex expressions which use parentheses
- Assessing the consistency of operators, precedence, and parentheses.

3.5 Threats to validity

> "Any research can be affected by different kinds of factors which, while extraneous to the concerns of the research, can invalidate the findings" [68]

Validating the whole process is important to achieve the goals of this study. Threats to validity may arise from different sources; corpus data, questions, participants, methods, etc.

Firstly, a corpus study is one of the threat to validity in this study. The data sample from the corpus only comes from open source projects. Thus, the validity of the findings may have a problem to be generalized to other software sample.

Secondly, questions may be biased, incorrect wording may be used in questions, or the questions may not guide the researchers to achieve their goals. This threat can happen when researchers wrongly create the questions, which are not understood by the participants.

Thirdly, participants recruitment is another threats to validity. Participants' background knowledge may affect their results or performance, including their level of programming knowledge, subjects learnt in previous study, nationality, and years of doing programming. Therefore, we will make sure to gather and collect comprehensive prior background information and demographics.

Lastly, threats of validity may also occur during the research design stage. Insufficient knowledge, contradictory logic or confusion while selecting methods may become a threat to validity. Threats due to technique can also occur during the design stage. For example, methods or techniques selection can be chosen based on researchers' interest [48]. Threats affect data collection such as bias from researchers' thoughts while doing the interviews can happen. There are several ways to minimise these threats for validation which will employ three approaches: *data triangulation* (collecting data from multiple sources) [40], *expert assistance* (getting advice and critique from experts) [7, 69] and *member checking* (sharing the findings with the participants).

Data triangulation

Use different data sources for validating the study. Successful convergence of several sources of data proves the validity of the study [40]. In this study, there are two different sources of data used for validating the study: students' responses and corpus data. There are also two different data collection techniques: semi-structured interviews and think-aloud methods.

Expert assistance

Get advice and critique from experts. This study design involves discussion of the methods and findings with supervisors who are experts in this area. All comments will be documented and revised again for validation purposes.

Member checking

Researchers take the findings back to the participants. The findings from data analysis will be shown to the participants in order to validate data interpretations.

Other threats to validity include:

- Invalid interpretation of the data. This threat can be mitigated by gathering all evidence related to the findings from the beginning of the study during analysis stage.
- Generalization of the findings is another threat which is categorized as an external validity threat. In this study, the concern is whether the end results (framework model) can be used or generalized to other groups or people outside this study. Alternative explanations from different analyses (mixed methods) will mitigate this threat later on.

Chapter 4

A pilot study: Initial findings

This chapter presents findings of our pilot study. This chapter begins with the explanation of selected participants and development of the questionnaire. The overall findings will show that problems occured among students with Java and thus demonstrate interest to explore in detail (RQ1) ”Do students have problems with operators and precedence in Java?”.

4.1 The pilot study

A pilot study was conducted as an online survey using open ended questions. The study aim to identify if programmers (especially students) had a problem with operator precedence.

We recruited a convenience sample of 29 participants from among first year students from Computer Science at University Utara Malaysia (UUM) who were enrolled in the introduction to programming course. They read the details of this study through the online survey before agreeing to continue to answer the questions. The majority of the participants were beginners in programming. Subjects were requested to answer a questionnaire related to operators and precedence in Java. During the session, students were not allowed to discuss the task and the overall time spent for each subject was recorded. Each participant was asked to complete the questionnaire with the level of complexity for each question is increasing from question 1 to 17 as shown in Appendix B.

4.2 Results

Students were requested to answer question 1 declaring themselves novice, beginner, intermediate or experienced programmer. Questions 2 to 5 asked the students to add parentheses to the expresssions given. The complexity of each question gradually increases from question 1 to question 4. In question 6 to 9 students were asked to provide output for Java code fragments. Appendix A shows the questions distributed to the participants.

4.2.1 Question 1

Question 1 requested students to indentify themselves as novice, beginner, intermediate or experienced programmers. The majority of the participants were at the novice and beginners levels. Table 4.1 below summarizes participants backgrounds:

Level	Total
Novice	17
Beginner	11
Intermediate	1
Expert	0
	29

Table 4.1: A summary of participants level in programming

4.2.2 Question 2

Question 2 is a basic Java arithmetic expression comprising plus, minus and multiplication operators. Although it is an arithmetic expression, Table 4.2 present the different patterns of participants' responses.

Answer	Coding ref.	Pattern of answers	Answers	#. of patterns
Correct answer	*a*	`1 + (2 * 3) - 4`	3	12
Correct answer	*b*	`1 + (2 * 3) - 4`, `1 + 6 - 4`, `7 - 4 = 3`	3	13
Correct answer	*c*	`(1 + (2 * 3)) - 4`	3	3

Table 4.2: Pattern of responses for question 1

All of the students were able to give the correct answer (3) and also managed to insert parentheses into the expression: although all participants gave correct answer however, the expression was parenthesized differently. The expected output was:

```
(1 + (2 * 3)) - 4 = 3
```

Some of the participants produced a result referring to coding reference '*a*' which also gives the output as 3. In addition, some students answered the question by showing the order of evaluation for the expression (coding reference '*b*') and this answer is similar to the expected output mentioned earlier.

4.2.3 Question 3

In question 3, students were asked to add parentheses to the expression:

```
a + b++ * c / a * b
```

Each variable was assigned to its own value; a=1, b=2, c=3. There were only 4 students who managed to answer correctly with coding reference *f*, g and i. Table 4.2 showed a variety of answers from students.

Answers	Coding ref.	Pattern of answers	#. patterns
Wrong answer	*a*	`a + ((b++ * c) / (a * b))`	6
Wrong answer	*b*	`a + (b++ * c) / (a * b)`	4
Wrong answer	*c*	`a + ((b++) * c) / (a * b)`	3
Wrong answer	*d*	`(a + ((b++ * c) / (a * b))`	1
Wrong answer	*e*	`a + (((b++) * c) / (a * b))`	2
Correct answer	*f*	`a + (((((b++) * c) / a) * b)`	2
Correct answer	*g*	`a + (((b++ * c) / a) * b)`	1
Wrong answer	*h*	`((a + b++) * c) / a * b)`	1
Correct answer	*i*	`a + (b++) * c / a * b`	1
Wrong answer	*j*	Error	8
		Total number of students	29

Table 4.3: Pattern of responses for question 3

The aim of question 3 is to understand students' understanding of precedence. The question includes increment and arithmetic operators. In general, an increment operator has higher precedence compared to an arithmetic operator. Thus, increment operators should be evaluated before other operators. Table 4.3, only 4 students answered correctly with the coding reference *f, g, i*. Coding reference *f* is the most correct answer which shows the correct order of evaluation by looking at the parenthesis. On the other hand, many students put parentheses around the `a * b` subexpression and also parenthesized operands of the multiplication operator.

4.2.4 Question 4 & Question 5

Variables:
`x = 3, y = 2, z = 1`

- Question 4: `z - 3 * x != y + 5`

Answer	Coding ref.	Pattern of answers	Values	#. patterns
Correct answer	*a*	`z - (3 * x) != y + 5`	True	2
Correct answer	*b*	`(z - (3 * x)) != (y + 5)`	True	7
Correct answer	*c*	`z - (3 * x) != (y + 5)`	True	4
Wrong answer	*d*	-	Error	16
			Total number of students	29

Table 4.4: Pattern of responses for question 4

- Question 5: `x + y < 10 && x / y == 3 || z != 10`

Answers	Coding ref.	Pattern of answers	Values	#. patterns
Correct answer	*a*	`((x + y) < 10) && (((x / y) == 3) || (z != 10))`	True	1
Correct answer	*b*	`((x + y) < 10) && ((x / y) == 3) || (z != 10)`	True	4
Correct answer	*c*	`(((x + y) < 10) && ((x / y) == 3) || (z != 10 )`	True	5
Wrong answer	*d*	-	Error	19
			Total students	29

Table 4.5: Pattern of responses for question 5

The expressions in questions 4 and 5 involved arithmetic, conditional, and equality operators. In Java arithmetic operators have higher precedence than the other two operators. A majority of the students produced a wrong answer (coding ref. *d*). In question 4, only 13 out of 29 participants managed to give the correct answer however, only 7 students showed their interpretation of the order of evaluation and precedence by parenthesizing the expression correctly (coding ref. '*b*'). For question 5, the total number of participants that managed to give the correct answer was reduced to 10 participants. Fortunately, students with correct answers faced no confusion with conditional and equality operators.

4.2.5 Questions 6-9

The aim of questions 6-9 is to gather students' interpretation of an expression with or without parenthesis. The questions are listed below:
What is the output of the following Java fragment program?

```
a = 1;
b = 3;
c = 2;
```

- Question 6:
  `System.out.println(a + b < a - c)`

Answers		Total number of answers
Correct answer	*'False'*	17
Wrong answer	Unique answers	12
	Total number of students	29

Table 4.6: Pattern of responses for question 6

- Question 7:

```
System.out.println((a + b) < a - c)
```

Answers		Total number of answers
Correct answer	*'False'*	18
Wrong answer	Unique answers	11
	Total number of students	29

Table 4.7: Pattern of responses for question 7

- Question 8:

```
System.out.println((a + b) < (a - c))
```

Answers		Total number of answers
Correct answer	*'False'*	18
Wrong answer	Unique answers	11
	Total number of students	29

Table 4.8: Pattern of responses for question 8

- Question 9:

```
System.out.println(a + (b < a) - c)
```

Answers		Total number of answers
Correct answer	*'Error'*	17
Wrong answer	Unique answers	12
	Total number of students	29

Table 4.9: Pattern of responses for question 9

Although the expression in questions 6-9 were required students to evaluate either '*True*' or '*False*', only 17 or 18 out of 29 students managed to answer correctly. Based on these questions, it is seen that although parentheses are added to the expression, there are still students who were not able to give the correct answer.

Question 9 is particularly interesting; according to the basic rule of operator precedence, the expression produced an error. That is because the expression `(b < a)` produces

a boolean result and it cannot be added with the addition operator (figure 4.1). In this question, there were 12 students who were not able to give the correct answer.

```
Main.java:16: error: bad operand types for binary operator '+'
        System.out.println(a + (b < a) - c);
                           ^
  first type:  int
  second type: boolean
1 error
```

Figure 4.1: Error produce when compiling question 9

4.2.6 Questions 10-15

In this section all questions aimed to check student understanding on side effects [19] in Java programs and the questions include increment operators.
Given the Java fragment program below:

- Question 10:
  What is the value of x if `i=-1` ?

```
if(i != -1) {
        System.out.println(++i);
        System.out.println(x=i++); }
else {
        System.out.println(x=i);
}
```

  According to table 4.10, there were 20 students who managed to provide correct answers. A total of 9 students gave a variety of wrong answers. Some of them gave the correct answer value however, the output style was wrong for example `x=-1`.

Answers	Students' answer	Number of students
Correct answers	`-1`	20
Wrong answers	`x=i`	1
	`x=-1`	4
	`0`	1
	`1`	1
	`-1 !=-1 false`	1
	`3`	1
	Total number of students	29

Table 4.10: Pattern of responses for question 10

- Question 11:
  What is the value of x if `i=0` ?

```
if(i != -1) {
        System.out.println(++i);
        System.out.println(x=i++); }
else {
        System.out.println(x=i);
}
```

In this question, only 3 students were able to provide correct answers, while the other students gave variety of wrong answers. As shown in Table 4.11, the number of students who gave the correct answer was smaller than the number for question 10. Confusion in dealing with increment operators led the students to provide the wrong answers.

Answers	Students' answer	Number of students
Correct answers	1 1	3
Wrong answers	1234	1
	1	8
	0	3
	2	1
	2output:11	2
	1x=1	2
	1x=0	1
	-1	1
	1 2	2
	0 1	1
	2 2	1
	false	1
	0x=1	2
	Total number of students	29

Table 4.11: Pattern of responses for question 11

- Question 12:
  What is the value of x if `i=1` ?

```
if(i != -1) {
        System.out.println(++i);
        System.out.println(x=i++); }
else {
        System.out.println(x=i);
}
```

The answers to this question were similar to question 11. There were only 4 students who managed to give correct answers while others gave the wrong answers. 10 of the students failed to get the correct answer because the answers were not following the coding style; they answered `2` instead of `2 2`.

Answers	Students' answer	Number of students
Correct answers	`2 2`	4
Wrong answers	`2`	10
	`1`	4
	`x!=-1`	1
	`1x=2`	1
	`3output:22`	2
	`2x=2`	2
	`1x=12x=2`	1
	`3`	1
	`-1 1`	1
	`2x=1`	1
	`1!=-1false`	1
	Total number of students	29

Table 4.12: Pattern of responses for question 12

- Question 13:
  What is the value of x if `i=-1` ?

```
if(i > 0) {
        System.out.println(x=i++); }
else {
        System.out.println(x=i);
        System.out.println(i=x++);
}
```

Table 4.13 shows another decrease in the number of correct answers. Only 2 out of 29 managed to give correct answers. Then, 10 of the students considered as wrong answer because they were not following the style of answering according the coding provided, while other students stated their answers differently.

Answers	Students' answer	Number of students
Correct answers	-1 -1	2
Wrong answers	-1	10
	1	3
	x=-1 i=-1 x=-1 i=0	2
	0output:-1-1	2
	x=-1 i=0	2
	error	2
	-1 0	2
	0 -1	1
	-1 i=-1	1
	()	1
	0	1
	Total number of students	29

Table 4.13: Pattern of responses for question 13

- Question 14:
  What is the value of x if `i=0` ?

```
if(i > 0) {
        System.out.println(x=i++); }
else {
        System.out.println(x=i);
        System.out.println(i=x++);
}
```

Question 14 also showed no improvement among students when dealing with increment operators. Only 4 students gave correct answers while the other 25 students failed to provide correct answers.

Answers	Students' answer	Number of students
Correct answers	0 0	4
Wrong answers	0	11
	x=0 i=0 x=0 i=1	1
	1output:0 0	2
	x=0 i=1	3
	x=0 i=1 x=1 i=2	1
	-1	1
	1	1
	2	1
	0 1	3
	1>0 true	1
	Total number of students	29

Table 4.14: Pattern of responses for question 14

- Question 15:
  What is the value of x if `i=1` ?

```
if(i > 0) {
        System.out.println(x=i++); }
else {
        System.out.println(x=i);
        System.out.println(i=x++);
}
```

In this question, the total number of students who managed to provide correct answer increased to 16 out of 29.

Answers	Students' answers	Number of students
Correct answers	`1`	16
Wrong answers	`234`	1
	`2output:1`	2
	`x=1 i=1 x=1 i=2`	1
	`x=2`	2
	`x=1 x=2`	1
	`1 2`	3
	`x=1`	1
	`()`	1
	`3`	1
	Total number of students	29

Table 4.15: Pattern of responses for question 15

As a summary of questions 10-15, students seemed to have difficulties dealing with pre and post increment operations. Besides, the total number of student who provided correct answers showed that students do have problems with side effect issues in programming.

4.2.7 Question 16

Question:
"Write an expression for the problem below and add parentheses if it requires:
People can enter the spin and draw competition unless they are above 25 years old, with salary below 1000 dolars and has siblings less than 3."
This question examined students' understanding of long conditional expressions without parentheses. The majority of the students were able to produce the correct answer (Table 4.16).

Answers	Students' answers	Total number of answers
Correct answer	`False`	21
Wrong answers	`true`	4
	`a>b&&45<=sum<a+d&&d>90`	1
	`-100`	1
	`false therefore the output:true`	2
	Total number of students	29

Table 4.16: Pattern of responses for question 16

4.3 Discussion

There were a total of 16 questions in the survey. The majority of the questions presented difficulties for the students. In question 2, only 13 students showed they understood the order of evaluation by parenthesizing correctly. In questions 6 to 9, some students thought that complex expressions require parentheses to make the order of evaluation clear, yet, not all expressions need to be parenthesized. We can see that not all students managed to answer the questions, whether the arguments are parenthesized or not.

Questions 10-15 focused on operators and side effects. The overall results showed that most of the students were not able to provide correct answers especially for questions 11-14. Those questions required students to give an output after executing the fragment code given. The questions dealing with increment operators and with more than one output had fewer correct answers from the students. We assume this is because of their confusion while interpreting the code and dealing with side effects at the same time.

4.4 Limitations

This pilot study has a number of limitations:

- The participants in the study were from the same university and had the same background in programming
- The total number of participants was relatively small (29)
- All of the survey questions were untested. The survey was based on previous publications and suggestion from programming websites.
- The survey is lacking of justification from the findings because some participants might have misinterpreted the questions. The misinterpretation might be caused by the unverified questions.

Chapter 5

Project plan

This chapter briefly explains the overall research plan of this thesis. The research plan is listed below:

- Phase 1: Review selected literature related to this study (in progress: see chapter 2).
- Phase 2: An early stage, pilot study among students who have computer science background. This has been completed and the findings of this study have been recorded as preliminary findings of this thesis (completed: see chapter 4).
- Phase 3: Design a comprehensive experimental study for the data collection. This includes use of questionnaires, interviews and also observations (in progress: see chapter 3).
- Phase 4: Data collection - carry out the study designed in phase 3
- Phase 5: Data analysis - analyze the data collected in phase 4
- Phase 6: Theory development - develop a theory
- Phase 7: Thesis completion

5.0.1 Research timeline

The table below shows the appropriate time frame for this research study.

Research Timeline	
Action plan	Dates
Initial literature review	July - Oct 2014
Human Ethic Approval (HEC)	31 Oct 2014
Preliminary study	Nov 2014
Preliminary analysis	Dec 2014
Attending workshop at Brisbane	23 - 25 Nov 2014
Presenting research proposal	Mid 2015
Further data collection and analysis	Jun 2015 - March 2016
Major Literature review	Dec 2014 - Jun 2016
Attending conferences	2015 - 2016
Writing thesis	2015 - 2016

5.1 Thesis outline

A thesis outline for this study is as follows:

- Chapter 1 : Introduction - This chapter will introduce the research problem, objectives, hypotheses, research questions and thesis outline.
- Chapter 2 : Literature review - This chapter will provide background about the selection of syntax in programming language design, and result in the area.
- Chapter 3 : Research Methodology - This chapter will explain the details of the research methodology which is appropriate for this study.
- Chapter 4 : Data Collection and Analysis - This chapter will present the data collection and the analysis.
- Chapter 5 : Research Findings - This chapter will present the findings from this research.
- Chapter 6 : Discussion and Conclusion - This chapter will interpret the findings and provide a detailed description of how the findings contribute to new theories or improve current theories. This chapter will conclude by summarizing the contributions, limitations and future work.

Appendix A

Human Ethics Application and Approval

Human Ethics Application

Application ID :	0000021215
Application Title :	operator precedence in programming language design
Date of Submission :	16/10/2014
Primary Investigator :	Najwani Razali
Other Investigators :	Prof Dale Carnegie
	Prof James Noble
	Dr Stuart Marshall

Research Form

Information

Welcome to the Human Ethics Application Form

The following advice will assist you in completing this process:

Help contact

For information about Human Ethics, go to the Human Ethics web page.

For help, please email the Ethics Administrator.

Policy

You must read the Human Ethics Policy before beginning your application. The Policy includes a link to a sample consent form, information sheet, and transcribing confidentiality form which may be useful (see last page).

Health research may require HDEC approval. To find out if your research requires this, read the HDEC Guidelines or contact the chairperson of your committee for clarification.

Student research

If you are a student, check with your supervisor before filling in this form. You may need to complete School requirements before applying for ethical approval.

Student applications will be automatically forwarded to supervisor(s) and then Head of School/Delegate for approval when the form is submitted. Once the Head of School has approved it, the form will be automatically forwarded for committee review.

Technical

This online system works best on Internet Explorer and Safari. It may not work on your iPad or tablet.

A guide to using this online form, which includes a workflow showing how the approval process works, can be downloaded here.

If your application involves other researchers, you can use the Comments function of this form to communicate about the application with each other. Click on the Application Comments or Page Comments icon on the top right of the screen to view and add comments. Comments left on the form once it is submitted will be visible to your Head of School and committee reviewers, so **remember to delete any private comments before submitting the form**.

Process

All applications will be automatically forwarded to the Head of School for review when the application is submitted. Once the Head of School has approved it, the form will be automatically forwarded for committee review.

You will normally receive an outcome of the review of your application within three weeks, unless you apply during an advertised close-down period (for instance, applications may not be reviewed in December and January). NO part of the research requiring ethical approval may commence prior to approval being given.

To apply for an amendment or extension to an approved application, open the approved form and click on Apply for amendment/extension. You will then be able to complete the Amendment/Extension page and resubmit the form.

Application Details

1. Ethics category code*

   Human

   Clearance Purpose code

   Research Only

2. Application ID

   0000021215

3. Please select the appropriate committee below. Please note that:

   - The School of Information Management committee is not yet processing forms online.
   - Education applications are now handled by the Human Ethics Committee.

   *Human Ethics Committee

4. Title of project*

   operator precedence in programming language design

5. School or research centre*

Engineering and Computer Science

6. Please list all personnel involved in this project. Ensure that all are listed with the correct role. **If you are a student, do not add your supervisor here: you will be asked to add this information on the next page.**

   Please ensure that only one person is listed as Principal Investigator.

   To add a person, search for their Victoria ID if known, otherwise *either* their first *or* last name (whichever is the most unusual). Click on the magnifying glass to search for results.

   Press the **green tick** at the bottom right corner to save the person record.

   Add anybody who is involved in this project as:

   - Associate Investigator
   - Other Researcher
   - PhD Student
   - Masters Student
   - Research Assistant

   Click on the help button if you are having difficulty adding people to the list.*

1	Given Name	Najwani
	Surname	Razali
	Full Name	Najwani Razali
	AOU	SECS
	Position	Principal Investigator
	Primary?	Yes

7. Are any of the researchers from outside Victoria?*
   - ○ Yes
   - ◉ No

8. Is the principal investigator a student?*
   - ◉ Yes
   - ○ No

Next time you save this form or move to a new page, a Student Research page will appear after this one. Please complete the two questions on the Student Research page.

Student Research

7a. What is your course code (e.g. ANTH 690)?*

COMP690

7b. Please add your primary supervisor (the supervisor who should review this application).

If your supervisor is also the Head of School or the school ethics officer, you will need to discuss with your School who should approve this application as Head of School or delegate. The supervisor and Head of School or delegate **must not be the same person.**

To add your supervisor, search for their Victoria ID if known, otherwise *either* their first *or* last name (whichever is the most unusual).

Press the **green tick** at the bottom right corner to save the person record. *

1	Given Name	Robert
	Surname	Noble
	Full Name	Prof James Noble
	AOU	SECS
	Position	Supervisor
2	Given Name	Stuart
	Surname	Marshall
	Full Name	Dr Stuart Marshall
	AOU	SECS
	Position	Supervisor

If your supervisor is also the Head of School, you will need to assign a different person to the Head of School or Delegate role on the Signoff page.

7c. What is your email address? (this is needed in case the committee needs to contact you about this application)*

najwani84@gmail.com

Note that system-generated emails (eg approval notifications) will not necessarily come to this address. System-generated emails will come to the email address stored for you in Student Records. To change the record in Student Records, log into My Victoria, and click on Student Records. You will be able to update your email address from there.

Project Details

9. Describe the objectives of the project*

The objective of the project is to gather conceptual understanding among student about programming language. The collected information will be used to design a framework model for understanding their knowledge on programming languages. Based on the findings, we will propose better solutions for designing or improving representations for programming features.

10. Describe the benefits and scholarly value of the project*

Benefits from this study are:
- Information can be used guide approaches to teaching
- The proposed experimental framework can be used for comprehensive testing on other related issue in relation to programming languages

11. Describe the method of data collection. Note that later in this form, in the Documents section, you will need to upload any relevant documentation such as interview schedule, survey, questionnaires, focus group rules, observation protocols etc. Delays are likely if the interview questions are missing from the Documents section. *

The data collection will take place in programming language course in Victoria University. A selection of participants will be based on the academic record. This record will be discuss with the lecturer's that in charge for the course. Firstly, they were asked to answer a questionnaire within specific time. While participants answering the questionnaire, they will be observe in term of time taken to complete the task given. Then, an interview session will be done in order to gather a details information about their answers.
Methodology:
- Questionnaires
- Interviews
- Observations

12. Does your research have more than one phase?*

- ○ Yes
- ◉ No

Key Dates

If approved, this application will cover this research project from the date of approval

13. Proposed end date for data collection*

27/11/2015

14. Proposed end date for research project as a whole*

31/10/2016

Proposed source of funding and other ethical considerations

15. Indicate any sources of funding, including self-funding (self-funding means that you are paying for research costs such as travel, postage etc. from your own funds) (tick all that apply)

- ☐ Internally funded
- ☑ Externally funded
- ☐ Self-funded

15a. Describe the source of funding*

Malaysian Government

15b. Indicate any ethical issues or conflicts of interest that may arise because of sources of funding e.g. restrictions on publication of results*

none

16. Is any professional code of ethics to be followed?*

- ○ Yes
- ◉ No

17. Is ethical approval required from any other body?*

- ○ Yes
- ◉ No

18 Depending on the characteristics of your participants or location of the research, you may need to arrange permission from another body or group before proceeding. If this is the case, explain and describe how you are addressing this*

The study will be held at Victoria University and this study will focus on programming subject class. Therefore, I will ask permission from the current lecturer that responsible for the class before continue the experiment.

Treaty of Waitangi

19. How does your research conform to the University's Treaty of Waitangi Statute? (you can access the statute from Victoria's Treaty of Waitangi page)*

N/A

Information about participants

20. How many participants will be involved in your research? If you are using records (e.g. historical), please estimate the number of records*

Roughly 30 participants

21. What are the characteristics of the people you will be recruiting?*

They are experts people who have computer science background and students from computer science school

22. Are you specifically recruiting any of the following groups?

- Māori
- Pasifika
- Children/youth
- Students
- People who are offenders and/or victims of crime
- People with disabilities
- People in residential care
- People who are refugees

Please indicate below.*

- ○ Yes
- ◉ No

23. Have you undertaken any consultation with the groups from which you will be recruiting?*

Not yet

24. Provide details of consultation you have undertaken or are planning*

At the first meeting, they will be briefed about the experiment. They will be asked to answer a questionnaire. During the experiment, participants will be observed. Once they are finished with the tests, they will be asked to explain their interpretation of the problem given. All of their answers and justifications will be recorded.

25. Outline the method(s) of recruitment you will use for participants in your study*

It will be completely voluntary and it will be advertised in the class of forums. Information sheets and consent form will be distributed and explained to the research participants before obtaining their consent.

26. Will your participants receive any gifts/koha in return for participating?*

- ◉ Yes
- ○ No

26a. Describe the gifts/koha and the rationale*

Token of appreciation for the participants is a Thumb-drive with a value of nzd5 each.
This experiment will be done during the semester session and it will take some of their times for participating in this study. Their answer and action will all be observe. Therefore, I think the participants deserve a higher value for their contribution and participation.

27. Will your participants receive any other assistance (for instance, meals, transport, release time or reimbursements)?*

- ○ Yes
- ◉ No

28. Will your participants experience any special hazard/risk including deception and/or inconvenience as a result of the research?*

- ○ Yes
- ◉ No

29. Is any other party likely to experience any special hazard/risk including breach of privacy or release of commercially sensitive information?*

- ○ Yes
- ◉ No

30. Do you have any professional, personal, or financial relationship with prospective research participants?*

- ○ Yes
- ◉ No

31. What opportunity will participants have to review the information they provide? (tick all that apply)*

- [x] They will be given a transcript of their interview
- [ ] They will be given a summary of their interview
- [x] Other
- [ ] They will not have an opportunity to review the information they provide

31a. Explain how participants will be able to review the information they provide*

They are allowed to ask for a meeting or an email will be sent to them for the explanation about the significant findings from the study.

Informed consent

32. Will participation be anonymous? **'Anonymous' means that the identity of the research participant is not known to anyone involved in the research, including researchers themselves.** It is not possible for the researchers to identify whether the person took part in the research, or to subsequently identify people who took part (e.g., by recognising them in different settings by their appearance, or being able to identify them retrospectively by their appearance, or because of the distinctiveness of the information they were asked to provide).*

- ○ Yes
- ◉ No

33. Will contributions of participants be confidential? Confidential means that those involved in the research are able to identify the participants but will not reveal their identity to anyone outside the research team. Researchers will also take reasonable precautions to ensure that participants¿ identities cannot be linked to their responses in the future.*

- ◉ Yes
- ○ No

33a. How will confidentiality be maintained in terms of access to the research data? (tick all that apply)*

- [ ] Access to the research will be restricted to the investigator
- [x] Access to the research will be restricted to the investigator and their supervisor (student research
- [ ] Focus groups will have confidentiality ground rules
- [ ] Transcribers will sign confidentiality forms
- [ ] Other

33b. How will confidentiality be maintained in terms of reporting of the data? (tick all that apply)*

- [x] Pseudonyms will be used
- [ ] Participants will be named only in a list of interviewees
- [ ] Data will be aggregated and so not reported at an individual level
- [x] Participants will be referred to by role or association with an organisation rather than by name
- [ ] Names will be confidential, but other identifying characteristics may be published with consent
- [ ] Other

34. How will informed consent be obtained? (tick all that apply to all phases of the research you are describing in this application)*

- [ ] Informed consent will be implied through voluntary participation (anonymous research only)
- [x] Informed consent will be obtained through a signed consent form
- [ ] Informed consent will be obtained by some other method

Access, storage, use, and disposal of data

35. What procedures will be in place for the storage of, access to and disposal of data, both during and at the conclusion of the research? (tick all that apply)*

- [x] All written material will be kept in a locked file; access restricted to investigator(s)
- [x] All electronic information will be password-protected; access restricted to the investigator(s)
- [x] All questionnaires, interview notes and similar materials will be destroyed
- [x] Any audio or video recording will be returned to participants and/or electronically wiped
- [ ] Other procedures

35b. Will the data be destroyed immediately after the conclusion of the research?*

- ○ Yes
- ◉ No

35c. How many years after the conclusion of the research will the materials be destroyed?

3.00

36. If data and material are not to be destroyed, indicate why and the procedures envisaged for ongoing storage and security

The data obtained from this research will be valuable and provide insights into other aspects in future works. Hence, all the research data will be kept securely in storage as well as online, with access restricted to me.

Dissemination

37 How will you provide feedback to participants?*

Verbal feedback will be given to the participants as required. Also, when the research is completed and written, an email will be sent to participants to ask them whether they would like a copy of the thesis.

38. How will results be reported and published? Indicate which of the following are appropriate. The proposed form of publications should be indicated on the information sheet and/or consent form*

- [x] Publication in academic or professional journals
- [x] Dissemination at academic or professional conferences
- [x] Availability of the research paper or thesis in the University Library and Institutional Repository
- [ ] Other

39. Is it likely that this research will generate commercialisable intellectual property? (check the help text for more information about IP)*

- ( ) Yes
- (•) No

Documents

40. Please upload any documents relating to this application. Ensure that your files are small enough to upload easily, and in formats which reviewers can easily download and review*

Description	Reference	Soft copy	Hard copy
Participant information sheet(s)	informationSheet-me.pdf	✓	
Participant consent form(s)	consentFormme.pdf	✓	
Questionnaire or survey	test operators questionVersiontwo.pdf	✓	

Getting feedback on your application

You can seek feedback on your draft application, for instance from a mentor or a school Ethics representative **before** submitting it for review.

There are two ways of doing this:

1. Emailing your application to someone

You can email your application and any associated documents to another person at Victoria. To do this:

1. Click on the Action tab (on the left of the screen)
2. Click on Email application
3. Search for the person using **either** their first name **or** their last name (whichever is the most unusual)
4. Select the documents to include from the Document list (eg the Application PDF)
5. Click on Send or Zip and send

If you wish to send your application to someone outside Victoria, one option is emailing the application to yourself and then forwarding it.

2. Assigning a peer reviewer

You can add someone to the form as 'peer reviewer'. This means that they will be able to access your form by logging onto ResearchMaster. They will also be able to comment on your form online. **If you are a student, don't add your supervisor to the form as a peer reviewer - to get supervisor feedback, submit the form. Your supervisor may then make comments on it and ask you to review it further before it goes to the committee for review.** To do this:

1. Click on the Review tab on the left of the screen
2. Click on 'Peer reviewers'
3. Search for the person using their person code if known, or **either** their first name **or** their last name (whichever is the most unusual)
4. Click on the person's name
5. You may then also want to send the peer reviewer a notification, by clicking on Notify Peer Reviewer on the Actions tab

Checklist

Please check the information below and tick the box at the bottom of the page. Then follow the instructions to submit.

- Have you read the Human Ethics Policy?

- Have you included an information sheet for participants which explains:
  - the nature and purpose of your research;
  - the proposed use of the material collected
  - who will have access to the material collected
  - whether the data will be kept confidential to you
  - how anonymity or confidentiality is to be guaranteed?

- Does your information sheet also include:
  - a statement about participants' right to withdraw and the final date for doing so (and is this also referred to in the consent form)?
  - a statement confirming that the research has been approved by Victoria University of Wellington Human Ethics Committee?
  - a statement about the destruction of the data at the end of the project?
  - (for students) your supervisor's name and email address?

- Have you used your VUW email address?

- Have you included a written consent form?

  - If not, have you explained on the application form why you do not need to get written consent?
- Are your information sheets and consent forms on VUW letterhead?

- Have you included a copy of any questionnaire or interview schedule you propose using?

I have gone through the checklist and completed all the relevant tasks. *

☑ Yes

Signoff

41. This section records sign-off by all other researchers involved in the project **(principal investigators do not need to sign off).**

    If co-researchers are external to Victoria University they may be unable to access this site. In this instance, the Principal Investigator may sign off on their behalf. Please upload evidence of the co-researchers' signoff (e.g., a scanned email) to the Documents page.

    To sign off:

    1. Click on the pencil icon on the far right of the line with your name on it
    2. Click on I Accept
    3. Add the date
    4. Click on the green tick icon on the bottom of the signoff window
    5. Go to the Actions tab and click on 'Notify lead researcher that signoff is complete'

    This question is not answered.

42. Please add the Head of School (or delegate. e.g. school ethics officer) who should approve this application. This will be your own Head of School, or the person in your School responsible for approving Ethics applications. The form will be forwarded to this person automatically once it is submitted. **Please check with your School administration team if you are unclear who should be assigned this role. Adding the wrong person could lead to delays in processing your application.**

    If you are a student, the Head of School or delegate must not be the same person as your supervisor.

    Once you've searched for your Head of School/delegate, click on the green tick to add them, and then also save the application before submitting.

    Heads of School or delegates should process this form by clicking on the Actions tab and either approve it, or return it to the researcher for further changes. *

1	Given Name	Dale
	Surname	Carnegie
	Full Name	Prof Dale Carnegie
	AOU	SECS
	Position	Head of School (or delegate)

Please ensure that you **save your application before submitting it**. Once you have saved your application, to submit it, click on 'Actions' on the left hand side of the screen and then 'Submit for review'.

If you are a student, your application will go to your supervisor and then Head of School for approval once you submit it. If you are a staff member, your application will go straight to the Head of School for approval once you submit it.

If you have any feedback about this online form, please email it to ethicsadmin@vuw.ac.nz

Appendix B

Pilot Study Questionnaire

Test on operators and operator precedences in JAVA programming.
Please answer all the questions and thank you for your participation.

Write the correct answer based on description below:

1. Do you consider yourself: Please choose only one of the following:

   - A novice
   - A beginner programmer
   - An intermediate programmer
   - An experienced programmer

2. Add parentheses to the following expression based on their precedence in order to remove ambiguity and give the output:

   ```
   1 + 2 * 3 - 4
   ```

3. Add parentheses to the following expression based on their precedence in order to remove ambiguity and give the output:

   ```
   a + b++ * c / a * b
   ```

4. Add parentheses to the following expression based on their precedence in order to remove ambiguity and give the output:

   ```
   z - 3 * x != y + 5
   ```

5. Add parentheses to the following expression based on their precedence in order to remove ambiguity and give the output:

   ```
   x + y < 10 && x/y == 3 || z != 10
   ```

   Given the Java fragment program below, what is the output of the following code?

   ```
   int a = 1;
   int b = 3;
   int c = 2;
   ```

6. ```
   System.out.println(a+b<a-c);
   ```

   Answer:

7. ```
   System.out.println((a+b)<a-c);
   ```

   Answer:

8. ```
   System.out.println((a+b)<(a-c));
   ```
   ```

Answer:

9. `System.out.println(a+(b<a)-c);`

   Answer:

10. Given the Java fragment program below:

```
if (i != -1){
System.out.println(++i);
System.out.println(x = i++)
}
else {
System.out.println(x = i);
}
```

    What is the final value of x if i = -1?

11. Given the Java fragment program below:

```
if (i != -1){
System.out.println(++i);
System.out.println(x = i++)
}
else {
System.out.println(x = i);
}
```

    What is the final value of x if i = 0?

12. Given the Java fragment program below:

```
if (i != -1){
System.out.println(++i);
System.out.println(x = i++)
}
else {
System.out.println(x = i);
}
```

    What is the final value of x if i = 1?

13. Given the Java fragment program below:

```
if (i != -1){
System.out.println(x = i++)
}
else {
System.out.println(x = i);
 System.out.println(x++);
}
```

    What is the final value of x if i = -1?

14. Given the Java fragment program below:

```
if (i != -1){
System.out.println(x = i++)
}
else {
System.out.println(x = i);
 System.out.println(x++);
}
```

What is the final value of x if i = 0?

15. Given the Java fragment program below:

```
if (i != -1){
System.out.println(x = i++)
}
else {
System.out.println(x = i);
 System.out.println(x++);
}
```

What is the final value of x if i = 1?

16. Given the Java fragment program below, what is the output of the following code?

```
int a = 10;
int b = 5;
int d = 100;
int sum = a + b;

\lstinline $if( ! ( a < b && 45 <= sum || sum < a + d && d > 90)) {
System.out.println(``True'');
}
else {
System.out.println(``False'');
}$
```

17. Write an expression for the problem below and add parentheses if it requires:
"People can enter the spin and draw competition unless they are above 25 years old, with salary below 1000 dolars and has siblings less than 3"

Bibliography

[1] Operator precedence logic error. http://cwe.mitre.org/data/definitions/783.html. Accessed April 10, 2015.

[2] Operator precedence lab. See Url http://web.stanford.edu/~coopers/securecoding/OperatorPrecedenceLab.pdf, (2009). Accessed January 11, 2015.

[3] New: Bugs caused by bitwise operator precedence. See Url http://forum.dlang.org/thread/bug-4077-3@http.d.puremagic.com%2Fissues%2F, (2010). Accessed Jun 5, 2015.

[4] Unfortunate error message for incorrect operator precedence between 'new' and '-¿'. See Url https://llvm.org/bugs/show_bug.cgi?id=10326, (2011). Accessed 5 April, 2015.

[5] AL-QAHTANI, S. S., PIETRZYNSKI, P., GUZMAN, L. F., ARIF, R., AND TEVOEDJRE, A. Comparing selected criteria of programming languages java, php, c++, perl, haskell, aspectj, ruby, cobol, bash scripts and scheme revision 1.0-a team cplgroup comp6411-s10 term report. *arXiv preprint arXiv:1008.3434* (2010).

[6] ALLEN, E., CHASE, D., HALLETT, J., LUCHANGCO, V., MAESSEN, J.-W., RYU, S., STEELE JR, G. L., TOBIN-HOCHSTADT, S., DIAS, J., EASTLUND, C., ET AL. The fortress language specification. *Sun Microsystems 139* (2005), 140.

[7] ANDA, B., BENESTAD, H. C., AND HOVE, S. E. A multiple-case study of software effort estimation based on use case points. In *Empirical Software Engineering, 2005. 2005 International Symposium on* (2005), ieee, pp. 10–pp.

[8] BEZANSON, J., KARPINSKI, S., SHAH, V. B., AND EDELMAN, A. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145* (2012).

[9] BJORK, R. Language evaluation criteria. http://www.mathcs.gordon.edu/courses/cs323/lectures-2009/LanguageEvaluationCriteria.pdf, 2009.

[10] BLACK, A., DUCASSE, S., NIERSTRASZ, O., POLLET, D., CASSOU, D., AND DENKER, M. *Squeak by example*. Square Bracket Associates, 2007.

[11] BLACK, A. P., BRUCE, K. B., AND NOBLE, J. The grace programming language draft specification version 0.3. 1303.

[12] CARPENTER, D. *Precedence bug in i40e get pfc delay*.

[13] CLARKE, S. Evaluating a new programming language. In *13th Workshop of the Psychology of Programming Interest Group* (2001), pp. 275–289.

[14] CLOUGH, P., AND NUTBROWN, C. *A Student's Guide to Methodology*. SAGE Publications, 2012.

[15] CRESWELL, J. W. *Educational research: Planning, conducting, and evaluating quantitative*. Prentice Hall, 2002.

[16] CRESWELL, J. W. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2013.

[17] CROTTY, M. *The Foundations of Social Research: Meaning and Perspective in the Research Process*. SAGE Publications, 1998.

[18] DENZIN, N., AND LINCOLN, Y. *The SAGE Handbook of Qualitative Research*. Sage Handbook Of. SAGE Publications, 2011.

[19] DOLADO, J. J., HARMAN, M., OTERO, M. C., AND HU, L. An empirical investigation of the influence of a type of side effects on program comprehension. *Software Engineering, IEEE Transactions on 29*, 7 (2003), 665–670.

[20] DYER, R., RAJAN, H., AND CAI, Y. An exploratory study of the design impact of language features for aspect-oriented interfaces. In *Proceedings of the 11th annual international conference on Aspect-oriented Software Development* (2012), ACM, pp. 143–154.

[21] EASTERBROOK, S., SINGER, J., STOREY, M.-A., AND DAMIAN, D. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[22] ELO, S., AND KYNGÄS, H. The qualitative content analysis process. *Journal of advanced nursing 62*, 1 (2008), 107–115.

[23] GOLAFSHANI, N. Understanding reliability and validity in qualitative research. *The qualitative report 8*, 4 (2003), 597–607.

[24] GOSLING, J., JOY, B., STEELE JR, G. L., BRACHA, G., AND BUCKLEY, A. *The Java Language Specification*. Pearson Education, 2014.

[25] GREENE, J. C., CARACELLI, V. J., AND GRAHAM, W. F. Toward a conceptual framework for mixed-method evaluation designs. *Educational evaluation and policy analysis 11*, 3 (1989), 255–274.

[26] GUBA, E. G., LINCOLN, Y. S., ET AL. Competing paradigms in qualitative research.

[27] HARRELL, M. C., AND BRADLEY, M. A. Data collection methods. semi-structured interviews and focus groups. Tech. rep., DTIC Document, 2009.

[28] HECHT, M., DECKER, D., GRAFF, S., GREEN, W., LIN, D., DINSMORE, G., AND KOCH, S. Review guidelines for software languages for use in nuclear power plant safety systems. Tech. rep., DTIC Document, 1997.

[29] JACKSON, K. M., AND TROCHIM, W. M. Concept mapping as an alternative approach for the analysis of open-ended survey responses. *Organizational Research Methods 5*, 4 (2002), 307–336.

[30] JONES, D. Developer beliefs about binary operator precedence. *C Vu 18*, 4 (2006), 14–21.

[31] KACZMARCZYK, L. C., PETRICK, E. R., EAST, J. P., AND HERMAN, G. L. Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical symposium on Computer science education* (2010), ACM, pp. 107–111.

[32] KARSAI, G., KRAHN, H., PINKERNELL, C., RUMPE, B., SCHINDLER, M., AND VÖLKEL, S. Design guidelines for domain specific languages. *arXiv preprint arXiv:1409.2378* (2014).

[33] KRAUSS, S. E. Research paradigms and meaning making: A primer. *The qualitative report 10*, 4 (2005), 758–770.

[34] KRISHNAMURTH, S. *Programming and programming languages*, 2014 (accessed April 19, 2015). Available at `http://papl.cs.brown.edu/2014/index.html`.

[35] LAND, S. M., AND HANNAFIN, M. J. Patterns of understanding with open-ended learning environments: A qualitative study. *Educational Technology Research and Development 45*, 2 (1997), 47–73.

[36] LEGARD, R., KEEGAN, J., AND WARD, K. In-depth interviews. *Qualitative research practice: A guide for social science students and researchers* (2003), 138–169.

[37] LINCOLN, Y. S., LYNHAM, S. A., AND GUBA, E. G. Paradigmatic controversies, contradictions, and emerging confluences, revisited. *The Sage handbook of qualitative research 4* (2011), 97–128.

[38] LOUDEN, K., ET AL. *Programming languages: principles and practices*. Cengage Learning, 2011.

[39] LOUISE BARRIBALL, K., AND WHILE, A. Collecting data using a semi-structured interview: a discussion paper. *Journal of advanced nursing 19*, 2 (1994), 328–335.

[40] MARSHALL, C., AND ROSSMAN, G. B. *Designing qualitative research*. Sage publications, 2010.

[41] MAYER, C., HANENBERG, S., ROBBES, R., TANTER, É., AND STEFIK, A. An empirical study of the influence of static type systems on the usability of undocumented software. In *ACM SIGPLAN Notices* (2012), vol. 47, ACM, pp. 683–702.

[42] MERTENS, D. M. *Research and evaluation in education and psychology: Integrating diversity with quantitative, qualitative, and mixed methods*. Sage Publications, 2014.

[43] MILES, M. B., AND HUBERMAN, A. M. *Qualitative data analysis*. Sage Newbury Park,, CA, 1985.

[44] MINICHIELLO, V., ARONI, R., AND HAYS, T. *In-depth interviewing: Principles, techniques, analysis*. Pearson Education Australia, 2008.

[45] MORGAN, D. L. Paradigms lost and pragmatism regained methodological implications of combining qualitative and quantitative methods. *Journal of mixed methods research 1*, 1 (2007), 48–76.

[46] MYERS, M. D. Qualitative research in information systems.

[47] NASEHI, S. M., SILLITO, J., MAURER, F., AND BURNS, C. What makes a good code example?: A study of programming q&a in stackoverflow. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on* (2012), IEEE, pp. 25–34.

[48] ONWUEGBUZIE, A. J. Expanding the framework of internal and external validity in quantitative research.

[49] ONWUEGBUZIE, A. J., AND COMBS, J. P. Data analysis in mixed research: A primer. *International Journal of Education 3*, 1 (2011), E13.

[50] PANE, J. F., AND MYERS, B. A. Improving user performance on boolean queries. In *CHI'00 extended abstracts on Human factors in computing systems* (2000), ACM, pp. 269–270.

[51] PANE, J. F., AND MYERS, B. A. Tabular and textual methods for selecting objects from a group. In *Visual Languages, 2000. Proceedings. 2000 IEEE International Symposium on* (2000), IEEE, pp. 157–164.

[52] PANE, J. F., MYERS, B. A., AND RATANAMAHATANA, C. A. Studying the language and structure in non-programmers' solutions to programming problems. *Int. J. Hum.-Comput. Stud. 54*, 2 (Feb. 2001), 237–264.

[53] PATTON, M. *Qualitative Research & Evaluation Methods*. SAGE Publications, 2002.

[54] PATTON, M. Q. *Qualitative evaluation and research methods* . SAGE Publications, inc, 1990.

[55] PILLAY, N., AND JUGOO, V. R. An analysis of the errors made by novice programmers in a first course in procedural programming in java. *Preface of the Editors* (2006), 84.

[56] POPE, C., ZIEBLAND, S., AND MAYS, N. Qualitative research in health care: analysing qualitative data. *BMJ: British Medical Journal 320*, 7227 (2000), 114.

[57] POPEK, G. J., HORNING, J. J., LAMPSON, B. W., MITCHELL, J. G., AND LONDON, R. L. Notes on the design of euclid. In *ACM SIGOPS Operating Systems Review* (1977), vol. 11, ACM, pp. 11–18.

[58] PUNCH, K. F. *Introduction to social research: Quantitative and qualitative approaches*. Sage, 2013.

[59] RAMSEY, N. On teaching* how to design programs*: observations from a newcomer. In *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming* (2014), ACM, pp. 153–166.

[60] ROBSON, C. *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*. Regional Surveys of the World Series. Wiley, 2002.

[61] ROSSMAN, G. B., AND WILSON, B. L. Numbers and words combining quantitative and qualitative methods in a single large-scale evaluation study. *Evaluation review 9*, 5 (1985), 627–643.

[62] RUNESON, P., AND HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering 14*, 2 (2009), 131–164.

[63] SALDAÑA, J. *The coding manual for qualitative researchers*. No. 14. Sage, 2012.

[64] SAMIMI-DEHKORDI, L., KHALILIAN, A., AND ZAMANI, B. Programming language criteria for model transformation evaluation. In *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on* (2014), IEEE, pp. 370–375.

[65] SCOTT, M. L. *Programming language pragmatics*. Morgan Kaufmann, 2000.

[66] SEAMAN, C. Qualitative methods in empirical studies of software engineering. *Software Engineering, IEEE Transactions on 25*, 4 (Jul 1999), 557–572.

[67] SEBESTA, R. W. *Concepts of Programming Languages*, 9th ed. Addison-Wesley Publishing Company, USA, 2009.

[68] SELIGER, H. W., SHOHAMY, E., AND SHOHAMY, E. *Second language research methods*, vol. 31. Oxford University Press Oxford, 1989.

[69] SJOBERG, D. I., DYBA, T., AND JORGENSEN, M. The future of empirical methods in software engineering research. In *2007 Future of Software Engineering* (2007), IEEE Computer Society, pp. 358–378.

[70] SLONNEGER, K., AND KURTZ, B. L. *Formal syntax and semantics of programming languages*, vol. 340. Addison-Wesley Reading, 1995.

[71] TASHAKKORI, A., AND TEDDLIE, C. *Mixed methodology: Combining qualitative and quantitative approaches*, vol. 46. Sage, 1998.

[72] TASHAKKORI, A., AND TEDDLIE, C. *Sage handbook of mixed methods in social & behavioral research*. Sage, 2010.

[73] TEMPERO, E., ANSLOW, C., DIETRICH, J., HAN, T., LI, J., LUMPE, M., MELTON, H., AND NOBLE, J. Qualitas corpus: A curated collection of java code for empirical studies. In *2010 Asia Pacific Software Engineering Conference (APSEC2010)* (Dec. 2010), pp. 336–345.

[74] TERRA, R., MIRANDA, L. F., VALENTE, M. T., AND BIGONHA, R. S. Qualitas. class corpus: A compiled version of the qualitas corpus. *ACM SIGSOFT Software Engineering Notes 38*, 5 (2013), 1–4.

[75] THOMAS, D. R. A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation 27*, 2 (2006), 237–246.

[76] TONELLI, R., CONCAS, G., MARCHESI, M., AND MURGIA, A. An analysis of sna metrics on the java qualitas corpus. In *Proceedings of the 4th India Software Engineering Conference* (2011), ACM, pp. 205–213.

[77] TUCKER, A. B. *Programming Languages: Principles and Paradigms*, 2 revised ed. McGraw-Hill Higher Education, 2007.

[78] VAN MOURIK, T. Fortran 90/95 programming manual.

[79] VICTOR, K. R., AND NELSON, K. N. Formal semantics, syntax, pragmatics: An essence of programming language design. *Academic Research International 4*, 2 (2013), 124.

[80] VITOUSEK, M. M., KENT, A. M., SIEK, J. G., AND BAKER, J. Design and evaluation of gradual typing for python. In *Proceedings of the 10th ACM Symposium on Dynamic languages* (2014), ACM, pp. 45–56.

[81] WHITING, L. S. Semi-structured interviews: guidance for novice researchers. *Nursing Standard 22*, 23 (2008), 35.