UNIVERSITY OF LONDON | INTERNATIONAL PROGRAMMES

Introduction to Java and object-oriented programming Volume 1

S. Danicic

CO1109

2007

Undergraduate study in
Computing and related programmes

This is an extract from a subject guide for an undergraduate course offered as part of the University of London International Programmes in Computing. It contains sample material from Volume 1 of the subject guide plus the table of contents for Volume 2. Materials for these programmes are developed by academics at Goldsmiths.

For more information, see: www.londoninternational.ac.uk

Goldsmiths
UNIVERSITY OF LONDON

This guide was prepared for the University of London International Programmes by:

S. Danicic

This is one of a series of subject guides published by the University. We regret that due to pressure of work the author is unable to enter into any correspondence relating to, or arising from, the guide. If you have any comments on this subject guide, favourable or unfavourable, please use the form at the back of this guide.

University of London International Programmes
Publications Office
32 Russell Square
London WC1B 5DN
United Kingdom

www.londoninternational.ac.uk

Published by: University of London

Contents

1 Introduction **1**

1.1 How to Study this Course . . . 1
1.1.1 Reading List . . . 1
1.1.2 Suggested Schedule for Volume 1 . . . 1
1.1.3 Practice, Practice, Practice! . . . 2
1.1.4 The Challenging Problems . . . 2
1.1.5 The Examination . . . 2
1.1.6 Multiple Choice Questions . . . 3
1.2 The Course CD . . . 3
1.2.1 Course Material . . . 3
1.2.2 Books and Documentation . . . 3
1.2.3 Essential Software . . . 3
1.2.4 Extra Software . . . 4
1.3 Topics . . . 4
1.4 Books . . . 5
1.5 Installing Java . . . 5
1.6 Need Help Installing Java? . . . 5
1.7 Preliminaries . . . 5
1.8 Learning Outcomes . . . 6

2 Your First Java Program **7**

2.1 Learning Objectives . . . 7
2.2 Reading . . . 7
2.2.1 Main Reading . . . 7
2.2.2 Other Reading . . . 7
2.3 Directory Structure for the Course . . . 7
2.4 Task . . . 8
2.5 Your First Program . . . 8
2.5.1 `CLASSPATH` . . . 9
2.5.2 Setting the `CLASSPATH` on Windows XP . . . 9
2.5.3 Setting the `CLASSPATH` on Unix or Mac . . . 9
2.6 Editing, Compiling and Running your First Program . . . 9
2.6.1 Summary . . . 10
2.7 Analysis of the `HelloWorld` Program . . . 10
2.7.1 Comments . . . 10
2.7.2 The Other Way of Doing Comments . . . 11
2.7.3 The Program Heading . . . 11
2.7.4 Java is Case-Sensitive . . . 12
2.7.5 The Program Body . . . 12
2.7.6 `Strings` . . . 12
2.8 Some Compiler Error Messages . . . 13
2.8.1 Correcting Compilation Errors . . . 13
2.9 `print` vs. `println` . . . 14
2.10 Exercises on Chapter 2 . . . 15
2.10.1 Printing your Name . . . 15
2.10.2 Print your Name Three Times . . . 15

2.10.3 Print your Name Ten Times . . . 15
2.10.4 Print your Name a Hundred Times . . . 15
2.10.5 Print your Name a Thousand Times . . . 15
2.11 Summary . . . 16

3 Arithmetic Expressions **17**
3.1 Learning Objectives . . . 17
3.2 Reading . . . 17
3.3 Introduction . . . 17
3.4 Quotes Make All the Difference . . . 18
3.5 Multiplication is Written with an Asterisk * . . . 18
3.6 Division is Written with a Forward Slash / . . . 18
3.7 Converting Centigrade to Fahrenheit . . . 19
3.8 More About Division . . . 20
3.8.1 Integer Division . . . 20
3.8.2 Non-Integer Division . . . 20
3.8.3 Concatenating `Strings` . . . 21
3.9 Operator Precedence . . . 21
3.9.1 Brackets . . . 22
3.10 Exercises on Chapter 3 . . . 23
3.10.1 Pence to Dollars . . . 23
3.10.2 Ten Times Table . . . 23
3.10.3 One Hundred and Thirty Seven Times Table . . . 23
3.10.4 Operator Precedence . . . 23
3.10.5 Seconds in a Year . . . 23
3.10.6 Months in a Millennium . . . 23
3.10.7 Bits in a Megabyte . . . 23
3.10.8 Bits in a Gigabyte . . . 24
3.10.9 My Snail . . . 24
3.10.10 Feeding my Snail . . . 24
3.11 Summary . . . 25

4 Variables **27**
4.1 Learning Objectives . . . 27
4.2 Reading . . . 27
4.3 Introduction . . . 27
4.4 Declaring Variables . . . 28
4.4.1 Other Types . . . 28
4.5 Variable Names . . . 28
4.5.1 Important Fact about Replacing Variable Names . . . 29
4.6 Exercise: Boris Yeltsin's Pet Rabbit . . . 29
4.6.1 Exercise . . . 29
4.7 Wrong Assignments . . . 30
4.7.1 Executing Assignment Statements . . . 31
4.7.2 A Common Mistake . . . 31
4.7.3 Another Common Mistake . . . 32
4.8 Assigning to the Same Variable More Than Once . . . 32
4.9 A Common Mistake - Forgetting to Declare Variables . . . 33
4.10 Shorthand . . . 34
4.11 Exercises on Chapter 4 . . . 34
4.11.1 Add One . . . 34
4.11.2 Double . . . 34
4.11.3 Arithmetic . . . 35
4.11.4 `String` Concatenation . . . 35

4.11.5 `String` and `int` Concatenation . . . 35
4.11.6 Division by `int` . . . 36
4.11.7 Division by Real . . . 36
4.11.8 Division by Zero . . . 36
4.11.9 Further Exercises (no solutions) . . . 37
4.12 Summary . . . 38

5 Calling Methods **39**
5.1 Learning Objectives . . . 39
5.2 Reading . . . 39
5.3 Introduction . . . 39
5.4 What is a Method? . . . 39
5.5 How to Call a Method . . . 40
5.6 Some Simple Methods for Random Numbers . . . 40
5.7 Some Simple Methods for Graphics . . . 40
5.7.1 Instances of Objects . . . 42
5.8 Method Signatures . . . 42
5.8.1 The Class `java.lang.Math` . . . 42
5.8.2 Max . . . 44
5.9 Exercises on Chapter 5 . . . 45
5.9.1 Square . . . 45
5.9.2 Drawing a Cube . . . 45
5.9.3 Drawing a Childish Picture . . . 45
5.9.4 Signatures . . . 45
5.9.5 Exercise . . . 45
5.10 Summary . . . 45

6 Keyboard Input **47**
6.1 Learning Objectives . . . 47
6.2 Reading . . . 47
6.3 Introduction . . . 47
6.4 Prompting the User For Input . . . 49
6.5 Inputting `ints` . . . 50
6.5.1 nextInt() . . . 51
6.6 Exercises on Chapter 6 . . . 52
6.6.1 Double . . . 52
6.6.2 Add Two Numbers . . . 52
6.6.3 Average . . . 52
6.6.4 Question . . . 52
6.6.5 Task . . . 52
6.6.6 Task . . . 52
6.7 Summary . . . 53

7 Boolean Expressions and Conditional Statements **55**
7.1 Learning Objectives . . . 55
7.2 Reading . . . 55
7.3 Introduction . . . 55
7.3.1 Example of the `if - else` Statement . . . 56
7.3.2 Example of the `if` Statement . . . 56
7.4 Syntax and Semantics . . . 57
7.4.1 Syntax . . . 57
7.4.2 Semantics . . . 57
7.4.3 Biggest of Three . . . 57
7.4.4 Question . . . 58

7.5 The Syntax of the `if - else` Statement . . . 58
7.6 The Semantics of the `if - else` Statement . . . 58
7.6.1 The Empty Statement . . . 60
7.7 The Sequential Statement . . . 61
7.7.1 A Common Mistake is to Leave out Curly Brackets . . . 61
7.7.2 Exercise . . . 62
7.7.3 Reminder . . . 62
7.8 Program Layout . . . 62
7.9 `if` Statements . . . 63
7.9.1 The Syntax of the `if` Statement . . . 63
7.10 The Semantics of the `if` Statement . . . 63
7.10.1 Leaving out the Curlies . . . 63
7.11 Boolean Expressions . . . 64
7.11.1 The type `boolean` . . . 64
7.11.2 The Simplest Boolean Expressions . . . 64
7.11.3 Combining Boolean Expressions using Logical Operators . . . 65
7.12 Exercises on Chapter 7 . . . 66
7.12.1 Not Not . . . 66
7.12.2 Truth Table for AND . . . 66
7.12.3 Truth Table for OR . . . 66
7.12.4 Truth Table for Implication . . . 67
7.12.5 Sorting Two Numbers . . . 67
7.12.6 Sorting Three Numbers . . . 67
7.12.7 Notes on these Exercises . . . 67
7.12.8 Validating One Input . . . 67
7.12.9 Validating Two Inputs . . . 67
7.12.10 Sorting Four Numbers . . . 67
7.13 Summary . . . 68

8 Simple Loops **69**
8.1 Learning Objectives . . . 69
8.2 Reading . . . 69
8.3 Introduction . . . 69
8.3.1 Exercise . . . 70
8.4 Syntax of `for` Loops . . . 70
8.4.1 Exercise . . . 70
8.5 The Semantics of the `for` Loop . . . 70
8.5.1 Example . . . 71
8.5.2 Exercises . . . 72
8.6 Number of Iterations Depending on User Input . . . 72
8.6.1 Incrementing and Decrementing Shorthand . . . 73
8.7 `while` Loops . . . 73
8.7.1 The Syntax of `while` Loops . . . 74
8.7.2 The Semantics of a `while` Loop . . . 74
8.7.3 A Program that Goes On for Ever . . . 74
8.7.4 Exercise . . . 75
8.8 Crude Animations . . . 75
8.8.1 Random Animations . . . 76
8.9 Exercises on Chapter 8 . . . 77
8.9.1 One to Ten . . . 77
8.9.2 While . . . 77
8.9.3 Non-terminating . . . 77
8.9.4 Descending Sequence from Ten to One . . . 77
8.9.5 Even Numbers . . . 77

8.9.6 Odd Numbers . . . 78
8.9.7 Ten Times Table . . . 78
8.9.8 Multiples of Three . . . 78
8.9.9 Multiples . . . 78
8.9.10 Simple Times Table . . . 78
8.9.11 Largest of Ten . . . 79
8.9.12 Largest (User First Says How Many) . . . 79
8.9.13 Largest of As Many Numbers as Until Zero is Input . . . 79
8.9.14 A Guessing Game . . . 79
8.9.15 Factorial . . . 80
8.9.16 Exercise (No Solution) . . . 80
8.9.17 Moving Balls . . . 80
8.9.18 Random Animation . . . 80
8.10 Summary . . . 81

9 More on Calling Methods **83**
9.1 Learning Objectives . . . 83
9.2 Reading . . . 83
9.3 Different Uses of Method Calls . . . 83
9.3.1 Method Calls as Statements . . . 83
9.3.2 Method Calls as Expressions . . . 83
9.3.3 Void and Non-void Methods . . . 84
9.3.4 More Useful Methods in the Class `java.lang.Math` . . . 84
9.4 Static vs. Instance Methods . . . 85
9.4.1 The Class `java.lang.String` . . . 86
9.4.2 Instances of Objects . . . 86
9.4.3 `length()` . . . 86
9.4.4 `charAt()` . . . 87
9.4.5 `compareTo()` . . . 87
9.5 Type-Checking . . . 88
9.6 Parsing `Strings` that Represent Integers . . . 88
9.6.1 `Integer.parseInt()` . . . 88
9.7 Method Overloading . . . 89
9.8 Exercises on Chapter 9 . . . 90
9.8.1 Exercise . . . 90
9.8.2 Exercises – Type Checking . . . 90
9.8.3 Trying Methods . . . 90
9.8.4 Integer Methods . . . 91
9.8.5 Dictionary Order . . . 91
9.9 Summary . . . 92

10 One-Dimensional Arrays **93**
10.1 Learning Objectives . . . 93
10.2 Reading . . . 93
10.3 Introduction . . . 93
10.4 Array Index Out of Bounds . . . 96
10.4.1 Array Limitations . . . 97
10.5 Exercises on Chapter 10 . . . 99
10.5.1 Reverse . . . 99
10.5.2 Largest . . . 99
10.5.3 Crash . . . 99
10.5.4 Bad Input . . . 100
10.5.5 Array Largest, Smallest, Sum and Average . . . 100
10.5.6 Backwards . . . 101

10.5.7 Occurrences . . . 101
10.5.8 Longest `String` . . . 101
10.5.9 Exercise (No Solution) . . . 101
10.6 Summary . . . 102

11 Nested Loops **103**
11.1 Learning Objectives . . . 103
11.2 Reading . . . 103
11.3 Squares and Rectangles . . . 103
11.3.1 Exercise . . . 104
11.3.2 Exercise . . . 105
11.4 Non-rectangular Shapes . . . 105
11.4.1 Exercise: Left Top Triangles . . . 106
11.4.2 Exercise: Right Top Triangles . . . 106
11.4.3 Exercise: Hollow Squares . . . 107
11.5 Exercises on Chapter 11 . . . 108
11.5.1 RightBottomTriangleOfStars . . . 108
11.5.2 HollowRectangleOfStars . . . 108
11.5.3 HollowLeftBottomTriangleOfStars . . . 108
11.5.4 HollowLeftTopTriangleOfStars . . . 108
11.5.5 HollowRightTopTriangleOfStars . . . 108
11.5.6 HollowBottomRightTriangleOfStars . . . 108
11.5.7 Producing Multiplication Tables . . . 108
11.5.8 Multiplication and Exponentiation in Terms of Addition . . . 109
11.5.9 Multiplication in Terms of Addition . . . 109
11.5.10 Exponentiation in Terms of Addition . . . 109
11.5.11 A Clock Animation . . . 109
11.6 Summary . . . 111

12 Defining Your Own Methods **113**
12.1 Learning Objectives . . . 113
12.2 Reading . . . 113
12.3 Introduction . . . 113
12.3.1 The Purpose of Parameters . . . 114
12.4 The Structure of a Method Definition . . . 115
12.4.1 The Method Heading . . . 115
12.4.2 The Body of the Method . . . 115
12.5 Calling Methods . . . 116
12.6 Hollow Lines of Stars . . . 117
12.7 Some Nice Things about Methods . . . 117
12.7.1 Readability . . . 117
12.7.2 Reusability . . . 118
12.7.3 Breaking Down Problems into Smaller Ones . . . 118
12.8 Right Triangles Using Methods . . . 118
12.9 Methods Calling Other Methods . . . 118
12.9.1 Easy Exercise . . . 118
12.10 The Names of Formal Parameters . . . 120
12.11 Hard Exercise: Drawing a Hollow Diamond . . . 121
12.12 Non-void Static Methods . . . 122
12.13 Differences between Void and Non-void Methods . . . 123
12.14 The Clock Using Methods . . . 124
12.15 Exercises on Chapter 12 . . . 126
12.15.1 Left Bottom Triangle of Stars . . . 126
12.15.2 Left Top Triangle of Stars . . . 126

12.15.3 Hollow Left Bottom Triangle . . . 126
12.15.4 Hollow Left Top Triangle . . . 126
12.15.5 Right Bottom Triangle . . . 126
12.15.6 Right Top Triangle . . . 126
12.15.7 Hollow Right Bottom Triangle . . . 126
12.15.8 Hollow Right Top Triangle . . . 127
12.15.9 Complete `Shapes` . . . 127
12.15.10 Use Shapes . . . 127
12.15.11 Re-do Times Table Exercise . . . 127
12.15.12 Addall . . . 127
12.15.13 Array Methods . . . 128
12.15.14 Rewrite Array Sum Average etc. . . . 128
12.15.15 Rewrite Array Assignment . . . 128
12.15.16 Mult in Terms of Add . . . 128
12.15.17 Power in Terms of Add . . . 128
12.16 Summary . . . 129

13 Conclusion **131**
13.1 Topics . . . 131

II Appendices 133

A Challenging Problems **135**
A.1 Try out a Program [1,2] . . . 135
A.2 Rolling a Die [1,5] (dice.class) . . . 135
A.2.1 Hint . . . 136
A.3 Leap Years [1,7] . . . 136
A.3.1 Hint . . . 136
A.4 Drawing a Square [1,7] . . . 136
A.4.1 Hint . . . 136
A.5 How Old Are You? [1,7] (age.class) . . . 136
A.5.1 Hint . . . 137
A.6 Guessing Game [1,8] . . . 137
A.6.1 Hint . . . 137
A.7 Mouse Motion [1,8] (mouseInRect.class) . . . 138
A.7.1 Hint . . . 138
A.8 Maze [1,8] (maze.class) . . . 138
A.8.1 Hint . . . 138
A.9 Hangman [1,9] (hangman.class) . . . 139
A.9.1 Hint . . . 139
A.10 Roman Numerals [1,9] (Roman.class) . . . 140
A.10.1 Hint . . . 140
A.11 Shuffling a Pack of Cards (1) [1,10] (deal1.class) . . . 140
A.11.1 Hint . . . 140
A.12 Shuffling a Pack of Cards (2) [1,10] (deal2.class) . . . 141
A.12.1 Hint . . . 141
A.13 Noughts and Crosses (1) [1,11] (tictac.class) . . . 141
A.13.1 Hint . . . 141
A.14 Mastermind [1,11] (mastermind.class) . . . 142
A.15 Noughts and Crosses (2) [1,11] (tictac2.class) . . . 142
A.15.1 Hint . . . 142
A.16 Noughts and Crosses (3) [1,11] (tictac3.class) . . . 142
A.16.1 Hint . . . 143

A.17 Nim [1,11] (nim.class) . . . 143
A.17.1 Hint . . . 143
A.18 Clock [1,12] . . . 144
A.19 Spell-Checker [2,7] . . . 145
A.20 Diary Program [2,9] . . . 145
A.20.1 Hints . . . 148
A.20.2 Methods needed for Date Class . . . 148
A.20.3 Methods needed for Event Class . . . 149
A.20.4 Methods needed for Diary Class . . . 149

B Multiple Choice Questions **151**

C Answers to Exercises **165**

D Reading List **197**

Chapter 1

Introduction

1.1 How to Study this Course

This is an introductory programming course in Java. It is intended for students with no previous programming experience.

This is the first volume of two. You should work your way through each chapter in order. It is expected that you spend roughly one week studying each chapter. To study a chapter do the following:

1. Read through the chapter, trying out all examples on your computer as you go along.
2. Having read the chapter, attempt all the exercises at the end of chapter. It is important that you spend a considerable amount of time on each exercise before you look up the solution at the back of the guide.
3. If you cannot understand the solutions, try running them on your computer. If you are still having difficulty, then refer to the reading list at the beginning of each chapter.
4. Read the first item on the reading list for a different explanation of the topic covered in the chapter.

1.1.1 Reading List

The first section of each chapter has suggested reading. For example:

- [Dow03] Chapter 2
- [DD07] pages 53-57
- [Fla05] Chapter 1 and 2

The codes like "[Dow03]" refer to books in the Reading List on page 197.

1.1.2 Suggested Schedule for Volume 1

This schedule is an approximate indication of how much time to spend on each chapter. It assumes that all the material is to be covered in ten weeks. This is a minimum. If you have a longer period of study you can adjust these times proportionally.

Week 1: Chapters 2, 3 and 4

Week 2: Chapters 5 and 6

Week 3: Chapter 7

Week 4: Chapter 8

Week 5: Chapter 8
Week 6: Chapter 9
Week 7: Chapter 10
Week 8: Chapter 11
Week 9: Chapter 12
Week 10: Chapter 12

1.1.3 Practice, Practice, Practice!

Learning to program is a bit like learning a musical instrument. Although theory is important, practice is much, much more important. **The only way to learn to program is to write lots and lots of programs!** The way we judge a good musician is by listening to her playing a piece of music. Similarly we judge a programmer by running her programs. We can also, of course, admire the technique of a musician, but really the technique is just a means to an end. We don't really care how the violinist makes the sound as long as it sounds good to our ear.

Unfortunately, the musical analogy breaks down here. It is not enough that our computer programs work. Although computer programs are primarily meant to be understood by a computer, they also need to be understood by other humans who need to adapt them and improve them. Programs must be easy for humans to understand. Simplicity in programming is the key. The simpler your program, the better it is. Never show off by doing something in a complicated way. Always keep it simple.

1.1.4 The Challenging Problems

The challenging problems in Appendix A, page 135 are central to the course. By attempting to solve these problems you will learn an enormous amount about how to program. Each challenging problem has two numbers, for example [1,5] associated with it. This means you need to have read as far as Volume 1 Chapter 5 before you attempt this problem.

To return to the musical analogy, these problems are equivalent to the pieces you would be expected to perform as a new musician. The problems range from very easy to very difficult. Do not worry if you can't master them all as quickly as your colleagues. Different people learn at different speeds. Just because someone gets there first, it does not mean that they will end up being a better programmer than you.

1.1.5 The Examination

In Volume 2 there is a sample exam paper with no solutions and further past exam questions with solutions. You should start attempting these questions at least two months before your real exam. Try to attempt the sample exam paper in real exam conditions. Give yourself three hours and then mark your exam yourself by referring to the subject guides.

All the example programs given in the text, exercises and solutions, and other useful information will be provided on the accompanying CD and on the course website.

Details of how to access this website will be posted on:

http://www.londonexternal.ac.uk/current_students/programme_resources/index.shtml

1.1.6 Multiple Choice Questions

In the appendix of both Volumes 1 and 2, there are some multiple choice questions with solutions.

1.2 The Course CD

The course is accompanied by a CD containing the following useful material:

1.2.1 Course Material

Clickable CIS109 Subject Guide Volume 1
Clickable CIS109 Subject Guide Volume 2
CIS109 Java Programs and Solutions to Exercises
2006 Exam
2005 Exam

1.2.2 Books and Documentation

Java Documentation From Sun
Free Book: *How to Think Like a Computer Scientist* by Allen B. Downey
Free Book: *Thinking in Java* by Bruce Eckel
Free Book: *Introduction to Programming Using Java* by David J. Eck
Java Elements Documentation

1.2.3 Essential Software

Windows

TextPad Editor for Microsoft Windows
Java Install for Microsoft Windows
Acrobat Reader for Microsoft Windows

Linux

Java Install for Linux
Acrobat Reader for Linux

1.2.4 Extra Software

For Microsoft Windows

bluej for Microsoft Windows
NetBeans for Microsoft Windows

Linux

bluej for Linux
NetBeans for Linux
Eclipse for Linux
Eclipse for Linux

1.3 Topics

The first volume of the Java Subject Guide considers many of the basic concepts of programming. These include:

- Arithmetic and Boolean Expressions
- Variables and Types, Declarations and Assignments
- Input and Output
- Conditional Statements
- Loops: Simple and Nested
- Useful Built-in Methods
- Arrays
- Defining and Using Methods

In the second volume, we cover more advanced, but essential topics in Object Oriented Programming. These include:

- Command-line Arguments
- Recursion
- Packaging Programs
- More about Variables
- Bits, Types, Characters and Type Casting
- Files and Streams
- Sorting Arrays and Searching
- Defining Your Own Classes
- Inheritance
- Exception Handling
- Vectors

1.4 Books

I refer to a number of books throughout the text, specifically at the beginning of each chapter. Details of these books can be found in the bibliography on the last page of this volume (page 197). A good book to get started with is *How to Think Like a Computer Scientist* by Allen B. Downey. It is free and can be found on the course CD and at http://greenteapress.com/thinkapjava/ under the Gnu Free Documentation Licence. Thank you very much Allen B. Downey. I strongly recommend that you read chapters 1 to 13 of the book and do all its exercises.

1.5 Installing Java

Before you can usefully study this course, you need Java installed on your computer. The course CD contains an installable version of Java and instructions on how to install it.

Alternatively, go to

1. http://java.sun.com/javase/downloads/index.jsp
   Click on the **JDK 6** download button.
2. Java SE APIs and Documentation from http://java.sun.com/javase/reference/api.jsp.

If you are using Microsoft Windows you may wish to download and install TextPad Programmer's Text Editor from http://www.TextPad.com for editing, compiling, and running your Java programs (this is also provided on the course CD). You may prefer to use BlueJ from http://www.bluej.org/. Alternative programming environments include Netbeans which can be downloaded from http://java.sun.com/javase/downloads/index.jsp and Eclipse which can be downloaded from http://www.eclipse.org/.

1.6 Need Help Installing Java?

There is plenty of online help for installing Java. Try searching for "installing Java" using your favourite Internet search engine. See, for example, http://www.jibble.org/settingupjava.php.

1.7 Preliminaries

Before starting to learn Java, you need to know a few things about using a computer:

- You need some familiarity with a computer operating system. The operating system that you are using is probably one of the following:
  - Microsoft Windows
  - Unix (or Linux)
- You need to know how to create files using a text editor. In Microsoft Windows, we recommend that you use TextPad (Download from www.TextPad.com and on CD)

  In Unix, popular text editors that you might use include:

- vi
- emacs
- xedit
- nedit

It is important that you know how to create directories and subdirectories, copy, delete and move files.

1.8 Learning Outcomes

Having completed this subject guide you will understand programming concepts sufficiently to be able to write Java applications to solve simple programming problems. Topics covered include:

- Simple Output (Chapter 2, page 7)
- Arithmetic Expressions (Chapter 3, page 17)
- Variables (Chapter 4, page 27)
- Calling Methods (Chapters 5 and 9, pages 39 and 83)
- Keyboard Input (Chapter 6, page 47)
- Conditional Statements (Chapter 7, page 55)
- Simple `For` Loops (Chapter 8, page 69)
- One-Dimensional Arrays (Chapter 10, page 93)
- Nested Loops (Chapter 11, page 103)
- Defining Static Methods (Chapter 12, page 113)

Chapter 2

Your First Java Program

2.1 Learning Objectives

Chapter 2 explains:

- how to write Java programs that output messages to the terminal.
- about directory structure and where to put the programs you write during this course.
- about the `CLASSPATH` system variable.
- about the use of comments in a program.
- that Java is case-sensitive.
- about the purpose and syntax of the main method in a Java application.
- how to define `String` constants.
- how to compile and run Java programs.
- how to interpret some common compiler error messages.
- about the difference between `print` and `println`.

2.2 Reading

2.2.1 Main Reading

- Do all the exercises in Chapter 1 [Dow03] after you have read both this chapter and Chapter 1 of [Dow03].

2.2.2 Other Reading

- [Hub04] pages 1-13
- [CK06] pages 4-9

2.3 Directory Structure for the Course

I recommend that you create a directory (folder) for each chapter in the book. See Figure 2.1.

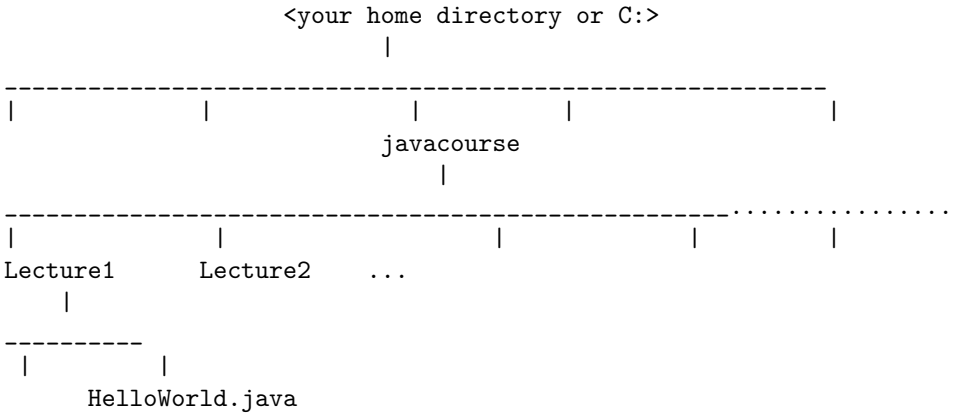

Figure 2.1: Directory Structure

2.4 Task

Read Pages 1-10 of [Dow03].

2.5 Your First Program

Throughout the text, we give suggested file names for each program. We put these file names in square brackets. For example, we write `[Lecture1/HelloWorld.java]`. This means that on the course CD the program can be found in a file called `HelloWorld.java` in a directory(folder) called `Lecture1`. I suggest that you also put your first Java program `HelloWorld` in a file called:

```
HelloWorld.java
```

in a directory(folder) called:

```
Lecture1
```

in the directory called

```
javacourse
```

If you do not put the programs where we suggest you may end up with problems since other programs may be looking in a particular place for another program.

2.5.1 CLASSPATH

There is a system variable called CLASSPATH that causes problems to beginners in Java. If you need to, please ask your tutor to help you with this.

This variable contains the set of directories (folders) where the Java system looks for classes (you will learn about classes later in the course).

2.5.2 Setting the CLASSPATH on Windows XP

In order to make everything in the course work smoothly you need to set the CLASSPATH system variable.

1. click on start → control panel.
2. click on performance and maintenance
3. click on system
4. click on advanced
5. click on Environment Variables
6. click on new
7. for the variable name write CLASSPATH and for the value write `c:\cis109\element.jar;c:\javacourse;.\`

If you have trouble with this, I suggest you do an Internet search using Google or some other search engine with *CLASSPATH java XP* as your search term.

2.5.3 Setting the CLASSPATH on Unix or Mac

Unix users should type:

```
export CLASSPATH=$HOME/element.jar:$HOME/javacourse:$CLASSPATH
```

Read the first chapter of [Hub04] for more details.

2.6 Editing, Compiling and Running your First Program

First, into TextPad (or the programming environment of your choice) type the example [Lecture1/HelloWorld.java]

```
//Our First Program
class HelloWorld
{
        public static void main(String[] args)
        {
          System.out.println("Hello World");
        }
}
```

Having typed it in, save it and compile it. To do this using TextPad, you click on `compile Java` under the `tools` menu. If you have typed it in correctly, nothing will happen. If you have not typed it correctly you will get some error messages from the compiler. If you get error messages, then check that every character you have typed is exactly as it appears in the text. If you still get errors, then try reading Section 2.8. This may help you to find your errors. When you have done this, then compile your program again. Repeat this process until you have no errors and then run your program.

To run your program using TextPad, you click on `Run Java Application` under the `tools` menu.

If you are using Unix or MS Windows and do not have TextPad, you can compile and run your Java programs from the command line. Type `javac HelloWorld.java` at the command line to compile your program and type `java HelloWorld` to run it.

What happens when you run the program? (See page 165 Number 1 for the answer.)

2.6.1 Summary

There are three phases in writing programs:

1. Editing the program
   (a) In Windows, I suggest that you use `TextPad`.
   (b) In Unix use your favourite text editor. I use `nedit`. Other people prefer `vi` or `emacs`.
2. Compiling the program
   (a) In Windows, click on `tools` followed by `compile` in `TextPad`.
   (b) In Unix (or DOS) type `javac` followed by file name, e.g. `javac HelloWorld.java`.
3. Running the program
   (a) In Windows, click on `tools` followed by `run Java application` in `TextPad`.
   (b) in Unix (or Dos) type `java` followed by class name, e.g. `java HelloWorld`.

2.7 Analysis of the `HelloWorld` Program

We will now analyse various aspects of the program: `[Lecture1/HelloWorld.java]` in more detail.

2.7.1 Comments

The very first line `// HelloWorld` is just a comment. After two forward slashes `//` you can write anything you like on that line. It will be ignored by the compiler and have no effect on what your program does when it runs. Comments are very important since when your programs become large the comments help to remind you how and why you wrote your programs the way you did.

2.7.2 The Other Way of Doing Comments

In Section 2.7.1 we saw one way of doing comments. In the program
[Lecture1/HelloWorld2.java]

```
/*
This program prints
Hello World
when we run it.
It was written by Sebastian Danicic
and he's very proud of it
*/
class HelloWorld
{
        public static void main(String[] args) /* the
                                                  main
                                                  method
                                               */
        {
          System.out.println("Hello World");
        /* This is where it does the printing*/
        }
}
```

we have included some text between `/*` and `*/`. This is how we do comments if we want them to last more than one line. We can think of `/*` as meaning "start comment" and `*/` as meaning "end comment". It is essential that from the beginning of your programming "life" you get into the habit of commenting your programs.

2.7.3 The Program Heading

The next line

```
class HelloWorld
```

tells us the name of the program. All Java programs have a `class` statement very near the beginning. Normally, because our program is called `HelloWorld`, we store it in a file called

```
HelloWorld.java
```

This is not essential however. We could have called the file `anything.java` and it would still have worked. Because of the program heading, after we compile it, we will end up with a file in the current directory called `HelloWorld.class`.

2.7.4 Java is Case-Sensitive

This means that it matters whether we use small or capital letters. If we had written `CLASS` instead of `class`, the compiler would give us an error message and we would have to correct it before being allowed to run the program. Try it and see!

2.7.5 The Program Body

The rest of the file is the *body* of the program.

Matching Brackets

It starts with an open curly bracket { and ends with a closing curly bracket }. When you write programs, brackets must always match: for every opening bracket there must be a corresponding closing bracket and vice-versa.

The Main Method

All Java applications have what is called a *main method* which always starts:

```
public static void main( String[ ] args)
```

This line is called the *heading* of the main method. The code inside the next pair of open and closing curly brackets is called the *body* of the main method.

```
{
   System.out.println("Hello World");
}
```

This is where we put what we actually want our Java program to do when we run it. In this example, the body of our main method consists of a single statement. In this case, the statement is a call to a *method*[1] whose name is `System.out.println`. We have passed this method the argument `"Hello World"`. When the `System.out.println` method is executed the `String` passed to it is *printed* on the computer screen.

[1] Methods will be studied in more detail in Chapters 5, 9 and 12.

2.7.6 `String`s

A `String` is a sequence of characters, with a double quote at either end. Examples of `String`s are

- `"sddffhh*^(_sg"`
- `"3253dssfdgg09231138"`
- `"1"`
- `""` (This one is called the *empty* `String`)

2.8 Some Compiler Error Messages

In Java all statements end with a semi-colon `;`. If we leave the semi-colon out the compiler will complain! Try compiling the program: [`Lecture1/bad.java`]

```
class  bad
{
        public static void main(String[] args)
        {
          System.out.println("Henry")
        }
}
```

When we try to compile this program we get an error message:

```
bad.java:6: ';' expected.
        System.out.println("Henry")
                                   ^
1 error
```

The Java compiler tells us that it got to line 6 when it realised that there was an error. In fact the error is on line 5. It puts a little caret `^` pointing at where the error might be. Another common error is to have the class name different from the file name. This is only a problem if we have the word `public`[2] before `class`. If we compile the program [`Lecture1/bad1.java`]

[2] The use of the word `public` will be explained later in the course.

```
public class  Bad1
{
        public static void main(String[] args)
        {
          System.out.println("Henry");
    }
}
```

we will get a compiler error message saying:

```
bad1.java:2: Public class Bad1 must be defined in a file called "Bad1.java".
public class  Bad1
              ^
1 error
```

2.8.1 Correcting Compilation Errors

If your programs do not conform exactly to the rules for the syntax of Java, errors will appear when you try to compile your program. When you start writing programs you will have lots of compilation errors. The best way to correct them is just to correct the first one and then recompile. This is because the first error sometimes makes the compiler think there are lots of other errors which are not really there. Note: **Just because your program has no compilation errors it doesn't mean it will do what you want it to do!**

2.9 `print` vs. `println`

Consider:

```
public class Name
{
 public static void main(String[] args)
 {
   System.out.println("Sebastian Danicic");
   System.out.println("Sebastian Danicic");
   System.out.println("Sebastian Danicic");
 }
}
```

As you have seen, every time we call the `System.out.println` method it prints its actual parameter (the bit in the brackets after the word `System.out.println`) and then goes on to the next line. The output to the program above is

```
Sebastian Danicic
Sebastian Danicic
Sebastian Danicic
```

If we had written:

```
public class Name
{
 public static void main(String[] args)
 {
   System.out.print("Sebastian Danicic");
   System.out.print("Sebastian Danicic");
   System.out.print("Sebastian Danicic");
 }
}
```

The output would have been:

```
Sebastian DanicicSebastian DanicicSebastian Danicic
```

So, as we have seen, `System.out.println` prints its argument followed by a newline character which makes the cursor go onto the next line.

2.10 Exercises on Chapter 2

2.10.1 Printing your Name

Create a new program that prints your own name instead of Henry's. Don't forget to compile and run the new program. (See page 165 Number 2 for the answer.)

Notice that the first program, HelloWorld.java starts with

```
class HelloWorld
```

and the second program, Name.java starts with

```
class Name
```

Notice that there is a file in your directory called `HelloWorld.class`.

Delete the file called `HelloWorld.class`. Now try to run it. What happens? (See page 165 Number 3 for the answer.)

2.10.2 Print your Name Three Times

Write a Program that prints your name 3 times; once per line. (See page 165 Number 4 for the answer.)

2.10.3 Print your Name Ten Times

Write a program that prints your name 10 times. (See page 165 Number 5 for the answer.)

2.10.4 Print your Name a Hundred Times

Write a program that prints your name 100 times.

2.10.5 Print your Name a Thousand Times

Write a program that prints your name 1000 times. In Chapter 8, on `For` loops, you will learn a shorter way of programming this!

2.11 Summary

Having worked on Chapter 2 you will have:

- Written Java programs that output messages to the terminal.
- Understood about directory structure and where to put the programs you write during this course.
- Been introduced to the `CLASSPATH` system variable.
- Learned about the use of comments in a program.
- Learned that Java is case-sensitive.
- Understood the purpose and syntax of the main method in a Java application.
- Learned how to define `String` Constants.
- Learned how to compile and run Java programs.
- Understood how to interpret some common compiler error messages.
- Understood the difference between `print` and `println`.

Chapter 3

Arithmetic Expressions

3.1 Learning Objectives

Chapter 3 explains:

- how arithmetic expressions are used in programming to perform calculations.
- an alternative way of writing comments.
- how to use the integer and real types in programming.
- how the division operator gives different types of result depending on its operands.
- how to concatenate `Strings` using `+`.
- about the use of operator precedence in expressions.
- about the use of brackets in computing expressions.

3.2 Reading

- [Dow03] Chapter 2
- [DD07] pages 53-57
- [Fla05] Chapter 1 and 2

3.3 Introduction

Arithmetic expressions are a way of telling a computer to do calculations. Compile and run the program [`Lecture1/OnePlusOne.java`]

```
class OnePlusOne
{
    public static void main(String[] args)
    {
      System.out.println(1+1);
    }
}
```

`1+1` is an example of an arithmetic expression. When we call `System.out.println(1+1)` the arithmetic expression 1+1 is first evaluated to produce 2. When we run this program it prints 2.

3.4 Quotes Make All the Difference

What is the output of [Lecture1/QuoteOnePlusOne.java]

```
class QuoteOnePlusOne
{
    public static void main(String[] args)
    {
      System.out.println("1+1");
    }
}
```

(See page 166 Number 6 for the answer.)

The quotes round 1+1 make it into a String. Without the quotes 1+1 is an integer. As we will see in Chapter 4, an integer is called an int in Java.

3.5 Multiplication is Written with an Asterisk *

I am going to America and taking 250 pounds sterling with me. I want to know how much this is in US Dollars. There are 1.51 dollars in each pound. [Lecture1/PoundstoDollars.java]

```
/*
I am going to America and taking 250 pounds with me.
I want to know how much this is in US Dollars.
There are 1.51. Dollars in each pound.
*/
class PoundstoDollars
{
    public static void main(String[] args)
    {
      System.out.println(250*1.51);
    }
}
```

As you can see, 1*2 means 1 times 2.

3.6 Division is Written with a Forward Slash /

Write a program which prints out how many pounds there are in one dollar assuming there are 1.51 dollars in a pound. [Lecture1/DollarsToPounds.java]

```
/*
There are 1.51. Dollars in each pound.
How many Pounds are there in one dollar?
*/
class DollarsToPounds
```

```
{
    public static void main(String[] args)
    {
     System.out.println("Assuming 1.51 dollars per pound");
     System.out.print("There are ");
     System.out.print(1/1.51);
     System.out.println(" pounds in one dollar.");
    }
}
```

As you can see, `1/2` means 1 divided by 2.

3.7 Converting Centigrade to Fahrenheit

Here is a program to print a Centigrade to Fahrenheit conversion table where x degrees centigrade is $32 + 9x/5$ degrees Fahrenheit. [`Lecture1/CentigradeToFahrenheit.java`]

```
/*
A Temperature conversion table
*/
class CentigradeToFahrenheit
{
    public static void main(String[] args)
    {
     System.out.println("centigrade  fahrenheit ");
     System.out.print(10);
     System.out.print("            ");
     System.out.println(32 + 9.0*10/5.0);
     System.out.print(20);
     System.out.print("            ");
     System.out.println(32 + 9.0*20/5.0);
     System.out.print(30);
     System.out.print("            ");
     System.out.println(32 + 9.0*30/5.0);
     System.out.print(40);
     System.out.print("            ");
     System.out.println(32 + 9.0*40/5.0);
    }
}
```

The output of this program is

```
centigrade  fahrenheit
10            50.0
20            68.0
30            86.0
40            104.0
```

3.8 More About Division

3.8.1 Integer Division

If both the numerator and denominator are integers then Java does **integer division**.
[Lecture1/div1.java]

```
class div1
{
    public static void main(String[] args)
    {
      System.out.println(3/2);//int divided by int
    }
}
```

prints 1 when we run it. This is because

- 3 and 2 are both `int`s
- The largest integer which is less than or equal to 3/2 is one.

Notice also that 3/(-2) would give -1.

The general rule for integer division is to work out the largest integer which is less than the absolute value of the expression.

3.8.2 Non-Integer Division

To represent real numbers we simply include a decimal point and at least one digit to the right of the decimal point, for example 1.0 or 1.51. If either the numerator or denominator is real the division is 'what we would expect'. The following programs all print `1.5`:

[Lecture1/div2.java]

```
class div2
{
    public static void main(String[] args)
    {
      System.out.println(3/2.0);//int divided by real
    }
}
```

[Lecture1/div3.java]

```
class div3
{
    public static void main(String[] args)
    {
      System.out.println(3.0/2);//real divided by int
```

```
    }
}
```

[Lecture1/div4.java]

```
class div4
{
    public static void main(String[] args)
    {
      System.out.println(3.0/2.0);//real divided by real
    }
}
```

3.8.3 Concatenating `String`s

As well as for adding numbers, the plus sign can be used for concatenating `String`s. For example `"Hello" + "fred"` gives `"Hellofred"`
and `"Hello " + "fred"` gives `"Hello fred"` (Note the space at the end of the first `String`).
The program DollarsToPounds.java in Section 3.6 could have been written in a neater way as:
[Lecture1/BetterDollarsToPounds.java]

```
/*
There are 1.51 Dollars in each pound.
How many Pounds are there in one dollar?
*/
class BetterDollarsToPounds
{
    public static void main(String[] args)
    {
      System.out.println("Assuming 1.51 dollars per pound");
      System.out.println("There are " + 1/1.51 +" pounds in one dollar.");
    }
}
```

3.9 Operator Precedence

What is the output of [Lecture1/Precedence.java]

```
class Precedence
{
        public static void main(String[] args)
        {
          System.out.println(5*1+1);
        }
}
```

The answer is 6. This is because when the system works out `5*1+1` it does the multiplication before it does the addition. We write

"times binds more tightly than plus"

or

"* binds more tightly than +".

3.9.1 Brackets

How would we make the system do the plus first? Answer: Use brackets! `5*(1+1)` would give 10.

You never need to remember operator precedence. Just use brackets to get the expression you want. Expressions inside brackets are always calculated first. For example (3+5)*2 evaluates to 16.

See [Fla05] page 29 for a list of all operators. or [DD07] page 53.

3.10 Exercises on Chapter 3

3.10.1 Pence to Dollars

Look up, on the internet or elsewhere, the exchange rate between UK Sterling and US Dollars. Write a program that works out how many pence in 250 dollars. (See page 166 Number 7 for the answer.)

3.10.2 Ten Times Table

Write a program that prints out the 10 times table. (See page 166 Number 8 for the answer.)

3.10.3 One Hundred and Thirty Seven Times Table

Write a program that prints out the 137 times table. (See page 166 Number 9 for the answer.)

3.10.4 Operator Precedence

Write some programs to test the order in which expressions are evaluated in Java.

Note

To make the following programs work, you have to write the numbers as real numbers with a decimal point. That is for two, write 2.0. For one million write 1000000.0 and so on. This will be explained in Volume 2.

3.10.5 Seconds in a Year

Write a program to work out the number of seconds in 365 days.

3.10.6 Months in a Millennium

Write a program to work out the number of months in a millennium (1000.0 years).

3.10.7 Bits in a Megabyte

Write a program to work out the number of bits in a megabyte. (A byte is 8 bits and a megabyte is 2.0^{20} bytes) To work out 2.0^{10} for now simply write

$$2.0 * 2.0 * 2.0 * 2.0 * 2.0 * 2.0 * 2.0 * 2.0 * 2.0 * 2.0$$

Eventually you will learn a better way of achieving this!

3.10.8 Bits in a Gigabyte

Write a program to work out the number of bits in a gigabyte. (A gigabyte is 2.0^{10} megabytes.)

3.10.9 My Snail

Assume light travels at 299,792,458 metres per second, and the star Proxima Centauri is 4.2 light years away. My snail travels at 48 centimetres an hour. How many years will it take my snail to get to Proxima Centauri and back? Write a Java program to work it out.

3.10.10 Feeding my Snail

My snail eats two grams of lettuce a day. Write a program that works out how many metric tons of lettuce it will have to take with it to Proxima Centauri. There are a million grams in a metric ton.

3.11 Summary

Having worked on Chapter 3 you will have:

- Understood how arithmetic expressions are used in programming to perform calculations.
- Learned an alternative way of writing comments.
- Been introduced to the integer and real types in programming.
- Understood how the division operator gives different types of result depending on its operands.
- Learned how to concatenate `Strings` using `+`.
- Understood operator precedence in expressions.
- Understood the use of brackets in computing expressions.

Chapter 4

Variables

4.1 Learning Objectives

Chapter 4 explains:

- the purpose of variables.
- about the primitive types of Java.
- about the allowable `Strings` used for variable names.
- how to declare variables.
- how to use assignment statements.

4.2 Reading

- [Dow03] Chapter 2
- [DD07] pages 48-49
- [Hub04] pages 19-23
- [CK06] pages 11-21

4.3 Introduction

Variables are very important in all programming languages. Variables are used to store values that we need later on in a computation. Each variable represents some memory inside the computer. Into this memory, values can be stored. In order to use a variable, we first **declare** it with a **variable declaration** and then store a value in it using an **assignment statement**.

Consider [`LectureVariables/Hello1.java`]

```
class Hello1
{
        public static void main(String[] args)
        {
          int s;             //Declaration of variable s
          s = 124;          // assignment statement
          System.out.println(s);
        }
}
```

4.4 Declaring Variables

In `Hello1.java`, first we declare the variable called `s`. The value `124` is then stored in this variable `s`. The **contents** of the variable `s` (in this case, 124) will be printed. When we run this program we will see

```
124
```

on the screen of our computer. We will not see `s` appearing on the computer screen. `s` is the **name** of the variable, not its contents.

Whenever we declare a variable we must give its **type**. The type of `s`, in this case, is `int`. This means that the only sorts of thing we can store in `s` are integers.

4.4.1 Other Types

Other basic types (usually called *primitive* types) in Java include

```
boolean
char
byte
short
long
float
double
```

Variables of different types are for holding different sorts of values.

Examples of legal declarations are:

```
boolean b; //A boolean variable called b.
char c,d; //Two char variables called c and d.
byte k; //A byte variable called k.
short silly; // A short variable called silly.
int m,n,p; //Three int variables called m, d and p.
long lilliput; // A long variable called lilliput.
float f1,g1,h; //Three float variables called f1, g1 and h.
double q,r; //Two double variables called q and r.
```

4.5 Variable Names

Any sequence of letters and digits that starts with a letter is a legal variable name. Examples of legal variable names are

- `x`

- x1
- banana
- Kilimanjaro
- Y2K
- t3x4y666minush4
- ZuZuZu11

There is absolutely no difference in the behaviour of

```
int x = 1543;
System.out.print(x);
```

and

```
int bananasplit = 1543;
System.out.print(bananasplit);
```

and

```
int BorisYeltsin54 = 1543;
System.out.print(BorisYeltsin54);
```

In each of the three program fragments we store the integer 1543 in a variable and then print out the contents of the variable. In all three

```
1543
```

will be printed out.

4.5.1 Important Fact about Replacing Variable Names

If we replace every occurrence of a variable name in a program by another that doesn't occur already in the program then the program will behave exactly the same.

4.6 Exercise: Boris Yeltsin's Pet Rabbit

Rewrite all the programs in this chapter that contain a variable in such a way as to not change their behaviour but so they all have a variable called `BorisYesltsinAndHisPetRabbit`.
(See page 167 Number 10 for the answer.)

4.6.1 Exercise

What is the output of [`LectureVariables/Hello1Boris.java`]

```
class Hello1Boris
{
   public static void main(String[] args)
   {
      int BorisYeltsinAndHisPetRabbit;//Declaration of variable
```

```
      BorisYeltsinAndHisPetRabbit = 1543; // assignment statement
      System.out.println(BorisYeltsinAndHisPetRabbit);
   }
}
```

If you think the answer is `BorisYeltsinAndHisPetRabbit` then please re-read this chapter. If you think the answer is `1543`, then carry on reading!

4.7 Wrong Assignments

Consider the program [`LectureVariables/WrongType.java`]

```
class WrongType
{
        public static void main(String[] args)
        {
          String s;       //Declaration of variable s
          s = 1;          // assignment statement
          System.out.println(s);
        }
}
```

When we try to compile this program we get the following error message:

```
WrongType.java:7: Incompatible type for =. Can't convert int to java.lang.String.
          s = 1;          // assignment statement
            ^
1 error
```

This is because we are trying to assign a value of 1 to a `String` variable but 1 is not a `String`, 1 is an integer (called `int` in Java). If we put double quotes round the 1 (i.e. `"1"`) it becomes a `String`.

Now consider Hello2.java:

```
class Hello2
{
        public static void main(String[] args)
        {
          int s;             //Declaration of variable s
          s = 1543;        //Assignment statement
          System.out.println(s + 25);
        }
}
```

This program prints `1568`.

So does Hello3.java:

```
class Hello3
{
        public static void main(String[] args)
        {
          int s,t;         //Declaration of variables s and t
```

```
          s = 1543;        // assignment statement
          t = 25;          // another assignment statement
          System.out.println(s + t);
        }
}
```

So does Hello4.java:

```
class Hello4
{
        public static void main(String[] args)
        {
          int s;             //Declaration of variable s
          s = 1543;          //assignment statement
          s = s + 25;        //another assignment statement
          System.out.println(s);
        }
}
```

4.7.1 Executing Assignment Statements

When an assignment is executed, first the expression on the right hand side is calculated and the result is put into the variable on the left hand side of the assignment. So in `Hello4.java` when executing the assignment statement `s = s + 25;` first the expression `s + 25` is calculated to give `1568`. The result is then stored in the variable `s`.

4.7.2 A Common Mistake

Consider [`LectureVariables/Hello5.java`]

```
class Hello5
{
        public static void main(String[] args)
        {
          int s;               //Declaration of variable s
          s = 15*2;            //assignment statement
          System.out.println("s");
        }
}
```

What does it output? The answer is `s`. It does not print `30` because we are asking the system to print the `String` value `"s"` not the value contained in the `int` variable `s`. It is very important that you understand this! The `"s"` is NOT the same as `s`. Again, the quotes make all the difference.

4.7.3 Another Common Mistake

A common mistake made by beginners is to declare the same variable more than once inside the main method (or as we shall see later, in any method). Java does not allow this.

Consider [LectureVariables/Dec2.java]

```
class Dec2
{
        public static void main(String[] args)
        {
          int s = 124 ;// s declared and assigned value 124
          int s = 53;// s declared and assigned value 53
          System.out.println(s);
        }
}
```

When we try to compile this program we get:

```
Dec2.java:6: Variable 's' is already defined in this method.
     int s = 53;
         ^
1 error
```

4.8 Assigning to the Same Variable More Than Once

It is allowed to assign to the same variable more than once, so the following program compiles with no errors.

[LectureVariables/TwoAssign.java]

```
class TwoAssign
{
        public static void main(String[] args)
        {
          int s;
          s = 1453; // First Assignment
          s = 26;   // Second Assignment
          System.out.println(s);
        }
}
```

The output of the program above is

```
26
```

because the value 1453 stored in variable s has been overwritten by the value 26. A new assignment to the same variable always causes the previous value in that variable to be thrown away and replaced with the new value.

4.9 A Common Mistake - Forgetting to Declare Variables

A very common mistake is to forget to define variables.

See, for example, [LectureVariables/Undeclared.java]

```
class Undeclared
{
        public static void main(String[] args)
        {
          s = 55;
          System.out.println(s);
        }
}
```

The compiler complains with

```
Undeclared.java:5: Undefined variable: s
          s = 55;
          ^
Undeclared.java:6: Undefined variable: s
          System.out.println(s);
                             ^
2 errors
```

The solution is simply to add the declaration `int s;` as in [LectureVariables/Declared.java]

```
class Declared
{
        public static void main(String[] args)
        {
          int s;
          s = 55;
          System.out.println(s);
        }
}
```

and now there are no errors. Another possible solution is:

[LectureVariables/Declared1.java]

```
class Declared
{
        public static void main(String[] args)
        {
          int s =55;
          System.out.println(s);
        }
}
```

4.10 Shorthand

Instead of `int x; int y; x=1;y=1;`

we can write:

```
int x=1; int y=1;
```

or even

```
int x=1,y=1;
```

4.11 Exercises on Chapter 4

4.11.1 Add One

What is the output of [`LectureVariables/AddOne.java`]

```
class AddOne
{
        public static void main(String[] args)
        {
          int x = 1;
          x = x+1;
          System.out.println(x);
        }
}
```

(See page 167 Number 11 for the answer.)

4.11.2 Double

What is the output of [`LectureVariables/DoubleDouble.java`]

```
class DoubleDouble
{
        public static void main(String[] args)
        {
          int x = 1;
          x = 2*x;
          x=2*x;
          System.out.println(x);
        }
}
```

(See page 167 Number 12 for the answer.)

4.11.3 Arithmetic

What is the output of [LectureVariables/p1.java]

```
class p1
{
        public static void main(String[] args)
        {
          int x = 1;
          int y = 3;
          int z;
          z = 2*x +3*y;
          System.out.println(z+y);
        }
}
```

(See page 167 Number 13 for the answer.)

4.11.4 `String` Concatenation

What is the output of [LectureVariables/p2.java]

```
class p2
{
        public static void main(String[] args)
        {
          String x = "hello ";
          x=x+x;
          System.out.println(x);
        }
}
```

(See page 167 Number 14 for the answer.)

4.11.5 `String` and `int` Concatenation

What is the output of [LectureVariables/p3.java]

```
class p3
{
        public static void main(String[] args)
        {
          String x = "hello ";
          int y = 5;
          System.out.println(x+y);
        }
}
```

(See page 167 Number 15 for the answer.)

4.11.6 Division by `int`

What is the output of [LectureVariables/p4.java]

```
class p4
{
        public static void main(String[] args)
        {
          int x= 11,y=5;
          int z= x/y;
          System.out.println(z);
        }
}
```

(See page 167 Number 16 for the answer.)

4.11.7 Division by Real

What is the output of [LectureVariables/p5.java]

```
class p5
{
        public static void main(String[] args)
        {
          int x= 11;
          double y=5.0;
          double z= x/y;
          System.out.println(z);
        }
}
```

(See page 167 Number 17 for the answer.)

4.11.8 Division by Zero

What is the output of [LectureVariables/p6.java]

```
class p6
{
        public static void main(String[] args)
        {
          int x= 11;
          System.out.println(x/0);
        }
}
```

(See page 167 Number 18 for the answer.)

4.11.9 Further Exercises (no solutions)

1. What would be the appropriate type for variables that represent each of the following:
   (a) The number of students in your class.
   (b) The average number of students per class in your college.
   (c) The distance from the earth to the moon measured to the nearest centimetre.
   (d) Whether a person has a degree.

4.12 Summary

Having worked on Chapter 4 you will have:

- Understood the purpose of variables.
- Learned the primitive types of Java.
- Learned which `String`s are allowable as variable names.
- Learned how to declare variables.
- Learned how to use assignment statements.

Appendix C

Answers to Exercises

1. [Lecture1/Henry.java]

```
Hello World is displayed on the computer screen!
```

2. [Lecture1/Name.java]

```
class Name
{
    public static void main(String[] args)
    {
        System.out.println("Sebastian Danicic");
    }
}
```

3. [Lecture1/Hello.java]

```
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld
```

4. [Lecture1/answers/three.java]

```
public class three
{
        public static void main(String[] args)
        {
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
        }
}
```

5. [Lecture1/answers/ten.java]

```
public class ten
{
        public static void main(String[] args)
        {
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
                System.out.println("Sebastian Danicic");
        }
```

```
}
```

6. 1+1

7. [Lecture1/DollarsToPence.java]

```
/*
There are 1.51. Dollars in each pound.
How many Pence are there in dollar?
*/
class BetterDollarsToPence
{
    public static void main(String[] args)
    {
    System.out.println("Assuming 1.51 dollars per pound");
    System.out.println("There are " + 100/1.51 +" pence in one dollar.");
    }
}
```

8. [Lecture1/TenTimesTable.java]

```
/*
This program prints out the ten times table
*/
class TenTimesTable
{
    public static void main(String[] args)
        {
         System.out.println(10 + " times " + 1 + " is " + 10*1 );
         System.out.println(10 + " times " + 2 + " is " + 10*2 );
         System.out.println(10 + " times " + 3 + " is " + 10*3 );
         System.out.println(10 + " times " + 4 + " is " + 10*4 );
         System.out.println(10 + " times " + 5 + " is " + 10*5 );
         System.out.println(10 + " times " + 6 + " is " + 10*6 );
         System.out.println(10 + " times " + 7 + " is " + 10*7 );
         System.out.println(10 + " times " + 8 + " is " + 10*8 );
         System.out.println(10 + " times " + 9 + " is " + 10*9 );
        }
}
```

9. [Lecture1/OneThreeSevenTimesTable.java]

```
/*
This program prints out the 137 times table
*/
class OneThreeSevenTimesTable
{
    public static void main(String[] args)
        {
         System.out.println(137 + " times " + 1 + " is " + 137*1 );
         System.out.println(137 + " times " + 2 + " is " + 137*2 );
         System.out.println(137 + " times " + 3 + " is " + 137*3 );
         System.out.println(137 + " times " + 4 + " is " + 137*4 );
         System.out.println(137 + " times " + 5 + " is " + 137*5 );
         System.out.println(137 + " times " + 6 + " is " + 137*6 );
```

```
            System.out.println(137 + " times " + 7 + " is " + 137*7 );
            System.out.println(137 + " times " + 8 + " is " + 137*8 );
            System.out.println(137 + " times " + 9 + " is " + 137*9 );
           }
}
```

10. [LectureVariables/Hello1Boris.java]

```
class Hello1Boris
{
   public static void main(String[] args)
   {
      int BorisYeltsinAndHisPetRabbit;//Declaration of variable
      BorisYeltsinAndHisPetRabbit = 1543; // assignment statement
      System.out.println(BorisYeltsinAndHisPetRabbit);
   }
}
```

11. 2
12. 4
13. 14
14. `hello hello`
15. `hello 5`
16. 2
17. 2.2
18. Answer:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at p6.main(p6.java:6)
```

19.
```
public static int abs(int)
public static long abs(long)
public static float abs(float)
public static double abs(double)
public static double acos(double)
public static double asin(double)
```
Notice I have left out the parameter name. It is acceptable to do this.

20. [Lecture2/answers/DoubleNew.java]

```
import java.util.Scanner;
public class DoubleNew
{
    public static void main(String[] args)
    {
        Scanner in =new Scanner(System.in);
        System.out.print("Enter a number ");
        int x=in.nextInt();
        System.out.println(2*x);
    }
}
```

21. [Lecture2/answers/AddTwoNew.java]

Appendix D

Reading List

[AW01] David Arnow and Gerald Weiss. *Introduction to Programming using Java*. Addison-Wesley, 2001.

[BB99] Duane A. Bailey and Duane W. Bailey. *Java Elements*. McGraw–Hill International Editions, October 1999. http://www.cs.williams.edu/~bailey/JavaElements/.

[Bis01] Judith Bishop. *Java Gently - Third Edition*. Addison-Wesley, 2001.

[CK06] Quentin Charatan and Aaron Kans. *Java - In Two Semesters - Second Edition*. McGraw-Hill, East London Business School, 2006.

[DD07] Harvey Deitel and Paul Deitel. *Java - How to Program - 7/e*. Prentice Hall International, 2007.

[Dow03] Allen B. Downey. *How to Think Like a Computer Scientist - Java Version*. Green Tea Press, 2003. A free book – see http://greenteapress.com/thinkapjava/.

[Fla05] David Flanagan. *Java in a Nutshell, Fifth Edition*. O'Reilly, 2005.

[Hub04] John R. Hubbard. *Schaums:Outlines - Programming with Java*. McGraw-Hill, University of Richmond, 2004.

[Inc] Sun Microsystems Inc. http://java.sun.com/javase/reference/api.jsp. This is where you can look up information about Java classes and methods.

[LO02] Kenneth A. Lambert and Martin Osborne. *Java - A Framework for Programming and Problem Solving*. Brookes-Cole, 2002.

UNIVERSITY OF LONDON | INTERNATIONAL PROGRAMMES

Introduction to Java and object-oriented programming Volume 2

S. Danicic

CO1109

2007

Undergraduate study in
Computing and related programmes

Goldsmiths
UNIVERSITY OF LONDON

This guide was prepared for the University of London International Programmes by:

S. Danicic

This is one of a series of subject guides published by the University. We regret that due to pressure of work the author is unable to enter into any correspondence relating to, or arising from, the guide. If you have any comments on this subject guide, favourable or unfavourable, please use the form at the back of this guide.

University of London International Programmes
Publications Office
32 Russell Square
London WC1B 5DN
United Kingdom

www.londoninternational.ac.uk

Published by: University of London

Contents

1 Introduction **1**
- 1.1 What We Cover in this Subject Guide . . . 1
  - 1.1.1 Suggested Schedule for Volume 2 . . . 1
- 1.2 Books . . . 2

2 Command-Line Arguments **3**
- 2.1 Learning Objectives . . . 3
- 2.2 Reading . . . 3
- 2.3 Introduction . . . 3
- 2.4 The Number of Command Line Arguments . . . 4
- 2.5 Exercises on Chapter 2 . . . 5
  - 2.5.1 Add One . . . 5
  - 2.5.2 Add . . . 5
  - 2.5.3 Backwards . . . 5
  - 2.5.4 Add All . . . 5
  - 2.5.5 Add All Real . . . 5
  - 2.5.6 Exercises (no solutions) . . . 5
- 2.6 Summary . . . 5

3 Recursion **7**
- 3.1 Learning Objectives . . . 7
- 3.2 Reading . . . 7
- 3.3 Definition of a Recursive Method . . . 7
- 3.4 Examples of Recursive Methods . . . 7
  - 3.4.1 Factorial . . . 7
  - 3.4.2 Greatest Common Divisor . . . 8
- 3.5 Exercises on Chapter 3 . . . 8
  - 3.5.1 Fibonacci Numbers . . . 8
  - 3.5.2 Multiplication . . . 8
  - 3.5.3 Exponentiation . . . 8
  - 3.5.4 Reversing Input . . . 9
  - 3.5.5 The Syracuse Sequence . . . 9
- 3.6 Summary . . . 9

4 Packaging Programs **11**
- 4.1 Learning Objectives . . . 11
- 4.2 Reading . . . 11
- 4.3 Introduction . . . 11
- 4.4 `Public` Classes . . . 12
- 4.5 File Names and Public Classes . . . 12
  - 4.5.1 Running Programs that are Part of Packages . . . 13
- 4.6 The `import` Statement . . . 13
- 4.7 Laborious but Worthwhile Packaging Task . . . 14
- 4.8 Exercises on Chapter 4 . . . 17
  - 4.8.1 Add Comments . . . 17
  - 4.8.2 No Import . . . 17
  - 4.8.3 A Complete Application . . . 17

4.8.4 Add your Own Method . . . 17
4.9 Summary . . . 18

5 More About Variables **19**
5.1 Learning Objectives . . . 19
5.2 Reading . . . 19
5.3 Introduction . . . 19
5.3.1 Local Variables . . . 19
5.4 What's Really in a Variable? . . . 21
5.5 Exercises . . . 21
5.6 Parameters Passed by Value . . . 22
5.7 Exercises on Chapter 5 . . . 23
5.7.1 Arrays (1) . . . 23
5.7.2 Arrays (2) . . . 23
5.7.3 `Strings` . . . 24
5.7.4 Test Array . . . 24
5.7.5 Test Int . . . 24
5.8 Summary . . . 26

6 Bits, Types, Characters and Type Casting **27**
6.1 Learning Outcomes . . . 27
6.2 Reading . . . 27
6.3 Introduction . . . 27
6.3.1 Exercise: Maximum Array Size . . . 28
6.4 Different Types Have Different Sizes . . . 28
6.5 Characters . . . 28
6.5.1 Exercise . . . 28
6.5.2 Exercise: Find All Characters . . . 28
6.6 Type Casting . . . 28
6.6.1 Quick Question . . . 29
6.7 The Method `read()` . . . 29
6.7.1 End of File . . . 30
6.7.2 A Reason why `read()` Returns an `int` . . . 30
6.8 More Type Casting . . . 30
6.8.1 Question . . . 30
6.9 Exercises on Chapter 6 . . . 32
6.9.1 Research . . . 32
6.9.2 `Int` to `Boolean` . . . 32
6.9.3 `Boolean` to `Int` . . . 32
6.9.4 `Float` to `Int` . . . 32
6.9.5 `Int` to `Float` . . . 32
6.9.6 `Double` to `Float` . . . 32
6.9.7 `Float` to `Char` . . . 32
6.9.8 `Int` to `Short` . . . 33
6.9.9 Next Biggest . . . 33
6.9.10 What is the Output? . . . 33
6.9.11 Largest `Int` . . . 33
6.10 Summary . . . 35

7 Files and Streams **37**
7.1 Learning Objectives . . . 37
7.2 Reading . . . 37
7.3 Introduction . . . 37
7.4 Reading Files . . . 37