Dear student: how should you use this book?

If you have not programmed before you should start at Chapter 1 and work through. If you have done a lot of previous programming, but not in Java, you might try skipping to Chapter 9 which contains a consolidation of all the concepts introduced in the preceding chapters.

Just in time learning

Each chapter focuses on a particular topic of programming, and is divided into a number of sections, most of which contain an example program. Programming and Java concepts are introduced and explained in appropriate detail just as they are needed for each example. This is a key principle of the **just in time** learning approach, making it easy to pick up the complexities in 'layers'.

Chapter aims

The start of each chapter contains a discussion of its aims. You should read this so that you know what to expect from the chapter. This is followed by a table of the chapter sections, with their particular aims and a summary of their associated coursework.

The parts in each example

The text in each section contains a mixture of parts.

- General discussion, introducing and explaining the example, and acting as a glue around the other parts.
- Introduction and explanation of programming and java concepts, such as this.

> *Concept* **`System.out.println().`** The simplest way to print a message is to use:
>
> ```
> System.out.println("This text will appear on the screen");
> ```

- Code listing of the example, which appears like this.

```
001: public class HelloWorld
002: {
003:   public static void main(String[] args)
004:   {
005:     System.out.println("Hello world!");
006:   }
007: }
```

- Thought provoking questions like the following. You should stop and ponder these for a while as you get to them, so that they might help deepen your understanding.

Coffee time: Soon you will know what all the code listed above means, but right now, what do you think that program will do?

0.0.1

- Runs of commands, especially runs of the current example.

Console Input / Output

```
$ java HelloWorld
Hello world!
$ _
```

These might appear anywhere, but in particular after we have developed each example, we show it being run with appropriate test data. If you take notice of this, you will also gradually and naturally pick up the skill of designing good test data for your programs.

Revisits to previous examples, or similar ones

Two key observations about programming are that there are many ways to solve a particular task, and that many programs are in fact similar to others. In this book when you meet later versions of previous programs, or new programs which are similar to previous ones, you will be invited to focus more on the different parts of the code, by us showing the similar parts in a smaller font.

```
001: public class HelloMum
 002: {
 003:   public static void main(String[] args)
 004:   {
005:     System.out.println("Hello mum!");
 006:   }
 007: }
```

This is better than showing you the new code only, which would make you have to flick back to previous pages in order to see the context.

Coursework

At the end of most sections you will find a piece of coursework. If you want to get the most from the book then you ought to undertake these as you get to them. They have been carefully created for you to try out the ideas covered in the section, get a deeper understanding of them, and thus become able to *write* programs, rather than just read them!

Please organize your work carefully. For example, you will find some pieces of coursework ask you to make a new version of a program you have written before, and yet these will have the same file name. Clearly you should keep all versions – especially if they are being used as part of some formal assessment of your progress. So, you should undertake the coursework for each section in a separate folder or directory. A good structure is to have a folder for each chapter, within which there is a folder for each section, and you can download this from the book's website, `http://www.cs.man.ac.uk/~jtl/JJIT`.

Getting the most from the coursework

Some coursework tasks, especially early on, are deliberately similar to their associated example and thus the simplest way to tackle them is to start with the example and make changes. *Don't do this!!!* Resist the temptation as much as you can. If you can do the coursework without needing to look at the corresponding example, it will prove to you that the ideas and the way you need to express them are *in your head*, or not. . . . Whereas if you always look at the example while doing the coursework, you will become dependent on that, learn less and have more difficulty when you get to the parts for which the coursework is *not* similar to the example!

So, I recommend you read the section containing the example just before undertaking the associated coursework, and then try your best to not need to look at it while you do the coursework.

Maybe you should keep a logbook?

You will learn a lot from this book, but from time to time you will want to keep your own notes, or listings of things you don't properly understand when you first meet them. In addition, when you undertake the coursework you may need to scribble your thoughts somewhere before you start typing. Perhaps this won't be necessary for the early tasks, but sooner or later you'll need to. And some coursework asks you to design test data for the program – where should you scribble that as you think it through? You could use a scrap of paper each time you need to write something, but then you'd probably end up throwing them away. This book invites you to be a little more professional than that by buying yourself a (cheap) hardback bound notebook, and calling it your logbook. Many of the coursework descriptions remind you of this idea by suggesting specific things you should record, but even where they don't, the suggestion is implicitly there. You should record the date and time you start each logbook entry, and perhaps give it a title too.

End of chapter concepts list

As you get to the end of each chapter, you will find a handy list of all the concepts that were introduced in it. Each of these has a quick 'self assessment' question to help you decide whether you should go back and reread that bit. You are strongly encouraged to spend the time going through these lists when you get to them: it will either be a short time, in which case no harm is done, or it will be time very well spent avoiding misunderstanding in the following chapters.

Finally, supporting reference tools

As you read through the book you may from time to time wish to look back at something you have previously met. To help you with this there are three supporting tools.

- On the book's website, `http://www.cs.man.ac.uk/~jtl/JJIT`, you will find reference documents containing the concepts that have been introduced. There is a version of this for each chapter, containing the concepts that have been covered by all chapters up to the end of that one.
- The table of contents in the book lists major concept names, as well as chapter and section names.
- There is a comprehensive index at the back of the book.