Java Just in Time:
Collected concepts after chapter 01

John Latham, School of Computer Science, Manchester University, UK.

April 15, 2011

Contents

1 Computer basics **1000**

- 1.1 Computer basics: hardware (page 3) 1000
- 1.2 Computer basics: hardware: processor (page 3) 1000
- 1.3 Computer basics: hardware: memory (page 3) 1000
- 1.4 Computer basics: hardware: persistent storage (page 3) 1001
- 1.5 Computer basics: hardware: input and output devices (page 3) 1001
- 1.6 Computer basics: software (page 3) 1001
- 1.7 Computer basics: software: machine code (page 3) 1001
- 1.8 Computer basics: software: operating system (page 4) 1001
- 1.9 Computer basics: software: application program (page 4) 1002
- 1.10 Computer basics: data (page 3) 1002
- 1.11 Computer basics: data: files (page 5) 1002
- 1.12 Computer basics: data: files: text files (page 5) 1002
- 1.13 Computer basics: data: files: binary files (page 5) 1003

2 Java tools **1003**

- 2.1 Java tools: text editor (page 5) 1003
- 2.2 Java tools: javac compiler (page 9) 1003
- 2.3 Java tools: java interpreter (page 9) 1004

3 Operating environment **1004**

- 3.1 Operating environment: programs are commands (page 7) 1004
- 3.2 Operating environment: standard output (page 7) 1004
- 3.3 Operating environment: command line arguments (page 8) 1004

1 Computer basics

1.1 Computer basics: hardware (page 3)

The physical parts of a computer are known as **hardware**. You can see them, and touch them.

1.2 Computer basics: hardware: processor (page 3)

The **central processing unit** (**CPU**) is the part of the **hardware** that actually obeys instructions. It does this dumbly – computers are not inherently intelligent.

1.3 Computer basics: hardware: memory (page 3)

The **computer memory** is part of the computer which is capable of storing and retrieving **data** for short term use. This includes the **machine code** instructions that the **central processing unit** is obeying, and any other data that the computer is currently working with. For example, it is likely that an image from a digital camera is stored in the computer memory while you are editing or displaying it, as are the machine code instructions for the image editing program.

The computer memory requires electrical power in order to remember its data – it is **volatile memory** and will forget its contents when the power is turned off.

An important feature of computer memory is that its contents can be accessed and changed in any order required. This is known as **random access** and such memory is called **random access memory** or just **RAM**.

1.4 Computer basics: hardware: persistent storage (page 3)

For longer term storage of **data**, computers use **persistent storage** devices such as **hard disc**s and **DVD ROM**s. These are capable of holding much more information than **computer memory**, and are persistent in that they do not need power to remember the information stored on them. However, the time taken to store and retrieve data is *much* longer than for computer memory. Also, these devices cannot as easily be accessed in a random order.

1.5 Computer basics: hardware: input and output devices (page 3)

Some parts of the **hardware** are dedicated to receiving input from or producing output to the outside world. Keyboards and mice are examples of **input device**s. Displays and printers are examples of **output device**s.

1.6 Computer basics: software (page 3)

One part of a computer you cannot see is its **software**. This is stored on **computer media**, such as **DVD ROM**s, and ultimately inside the computer, as lots of numbers. It is the instructions that the computer will obey. The closest you get to seeing it might be if you look at the silver surface of a DVD ROM with a powerful magnifying glass!

1.7 Computer basics: software: machine code (page 3)

The instructions that the **central processing unit** obeys are expressed in a language known as **machine code**. This is a very **low level language**, meaning that each instruction gets the computer to do only a very simple thing, such as the **addition** of two numbers, or sending a **byte** to a printer.

1.8 Computer basics: software: operating system (page 4)

A collection of **software** which is dedicated to making the computer generally usable, rather than being able to solve a *particular* task, is known as an **operating system**. The most popular examples for modern personal computers are Microsoft Windows, Mac OS X and Linux. The latter two are implementations of Unix, which was first conceived in the early 1970s. The fact it is still in widespread use today, especially by computer professionals, is proof that it is a thoroughly stable and well **design**ed and integrated platform for the expert (or budding expert) computer scientist.

1.9 Computer basics: software: application program (page 4)

A piece of **software** which is dedicated to solving a particular task, or application, is known as an **application program**. For example, an image editing program.

1.10 Computer basics: data (page 3)

Another part of the computer that you cannot see is its **data**. Like **software** it is stored as lots of numbers. Computers are processing and producing data all the time. For example, an image from a digital camera is data. You can only see the picture when you display it using some image displaying or editing software, but even this isn't showing you the actual data that makes up the picture. The names and addresses of your friends is another example of data.

1.11 Computer basics: data: files (page 5)

When **data** is stored in **persistent storage**, such as on a **hard disc**, it is organized into chunks of related information known as **file**s. Files have names and can be accessed by the computer through the **operating system**. For example, the image from a digital camera would probably be stored in a jpeg file, which is a particular type of image file, and the name of this file would probably end in `.jpg` or `.jpeg`.

1.12 Computer basics: data: files: text files (page 5)

A **text file** is a type of **file** that contains **data** stored directly as **character**s in a human readable form. This means if you were to send the raw contents directly to the printer, you would (for most printers) be immediately able to read it. Examples of text files include `README.txt` that sometimes comes with **software** you are installing, or source text for a document to be processed by the LaTeX[6] document processing system, such as the ones used to produce this book (prior to publication). As you will see shortly, a more interesting example for you, is computer program **source code** files.

1.13 Computer basics: data: files: binary files (page 5)

A **binary file** is another kind of **file** in which **data** is stored as **binary** (base 2) numbers, and so is not human readable. For example, the image from a digital camera is probably stored as a jpeg file, and if you were to look directly at its contents, rather than use some **application program** to display it, you would see what appears to be nonsense! An interesting example of a binary file is the **machine code** instructions of a program.

2 Java tools

2.1 Java tools: text editor (page 5)

A **text editor** is a program that allows the user to type and edit **text file**s. You may well have used `notepad` under Microsoft Windows; that is a text editor. More likely you have used `Microsoft Word`. If you have, you should note that it is not a text editor, it is a **word processor**. Although you can save your documents as text files, it is more common to save them as `.doc` **file**s, which is actually a **binary file** format. `Microsoft Word` is not a good tool to use for creating program **source code** files.

If you are using an **integrated development environment** to support your programming, then the text editor will be built in to it. If not, there are a plethora of text editors available which are suited to Java programming.

2.2 Java tools: javac compiler (page 9)

The Java **compiler** is called `javac`. Java program source is saved by the programmer in a **text file** that has the suffix `.java`. For example, the text file `HelloWorld.java` might contain the source text of a program that prints `Hello world!` on the **standard output**. This text file can then be **compile**d by the Java compiler, by giving its name as a **command line argument**. Thus the command

```
  javac HelloWorld.java
```

will produce the **byte code** version of it in the **file** `HelloWorld.class`. Like **machine code** files, byte code is stored in **binary file**s as numbers, and so is not human readable.

2.3 Java tools: java interpreter (page 9)

When the end user wants to run a Java program, he or she invokes the `java` **interpreter** with the name of the program as its **command line argument**. The program must, of course, have been **compile**d first! For example, to run the `HelloWorld` program we would issue the following command.

```
  java HelloWorld
```

This makes the **central processing unit** run the interpreter or **virtual machine** `java`, which itself then **execute**s the program named as its first argument. Notice that the suffix `.java` is needed when compiling the program, but no suffix is used when **run**ning it. In our example here, the virtual machine finds the **byte code** for the program in the **file** `HelloWorld.class` which must have been previously produced by the **compiler**.

3 Operating environment

3.1 Operating environment: programs are commands (page 7)

When a program is **execute**d, the name of it is passed to the **operating system** which finds and loads the **file** of that name, and then starts the program. This might be hidden from you if you are used to starting programs from a menu or browser interface, but it happens nevertheless.

3.2 Operating environment: standard output (page 7)

When programs **execute**, they have something called the **standard output** in which they can produce text results. If they are **run** from some kind of **command line interface**, such as a Unix **shell** or a Microsoft Windows **Command Prompt**, then this output appears in that interface while the program is running. (If they are invoked through some **integrated development environment**, browser, or menu, then this output might get displayed in some pop-up box, or special console window.)

3.3 Operating environment: command line arguments (page 8)

Programs can be, and often are, given **command line argument**s to vary their behaviour.