

# The first step

---

## Outcomes:

*By the end of this chapter you should be able to:*

- *explain the meaning of the terms **software**, **program**, **source code**, **program code**;*
  - *distinguish between **application software** and **system software**;*
  - *explain how Java programs are compiled and run;*
  - *provide examples of different types of java applications.*
- 

## 1.1 Introduction

Like any student starting out on a first programming module, you will be itching to do just one thing – get started on your first program. We can well understand that, and you won't be disappointed, because you will be writing programs in this very first chapter. Designing and writing computer programs can be one of the most enjoyable and satisfying things you can do, although it can seem a little daunting at first because it is like nothing else you have ever done. But, with a bit of perseverance, you will not only start to get a real taste for it but you may well find yourself sitting up till two o'clock in the morning trying to solve a problem. And just when you have given up and you are dropping off to sleep, the answer pops into your head and you are at the computer again until you notice it is getting light outside! So if this is happening to you, then don't worry – it's normal!

However, before you start writing programs we need to make sure that you understand what we mean by important terms such as *program*, *software*, *code* and *programming languages*.

## 1.2 Software

A computer is not very useful unless we give it some instructions that tell it what to do. This set of instructions is called a **program**. Programs that the computer can use can be stored on electronic chips that form part of the computer, or can be stored on devices like hard disks, CDs, DVDs, and USB drives (sometimes called memory sticks), and can often be downloaded via the Internet.

The word **software** is the name given to a single program or a set of programs. There are two main kinds of software:

- **Application software.** This is name given to useful programs that a user might need; for example, word-processors, spreadsheets, accounts programs, games and so on. Such programs are often referred to simply as **applications**.
- **System software.** This is the name given to special programs that help the computer to do its job; for example, operating systems (such as UNIX™ or Windows™, which help us to use the computer) and network software (which helps computers to communicate with each other).

Of course software is not restricted simply to computers themselves. Many of today's devices - from mobile phones to microwave ovens to games consoles - rely on computer programs that are built into the device. Such software is referred to as **embedded software**.

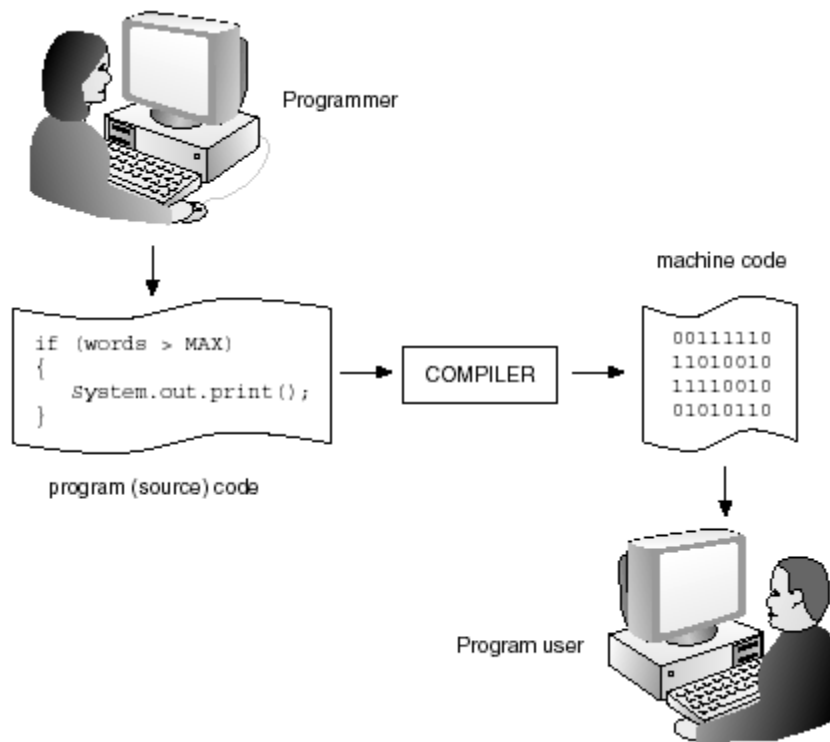
Both application and system software are built by writing a set of instructions for the computer to obey. **Programming**, or **coding**, is the task of writing these instructions. These instructions have to be written in a language specially designed for this purpose. These **programming languages** include C++, Visual Basic, Python and many more. The language we are going to use in this book is Java. Java is an example of an **object-oriented** programming language. Right now, that phrase might not mean anything to you, but you will find out all about its meaning as we progress through this book.

## 1.3 Compiling programs

Like most modern programming languages, the Java language consists of instructions that look a bit like English. For example, words such as **while** and **if** are part of the Java language. The set of instructions written in a programming language is called the **program code** or **source code**.

Ultimately these instructions have to be translated into a language that can be understood by the computer. The computer understands only **binary** instructions – that means instructions written as a series of 0's and 1's. So, for example, the machine might understand 01100111 to mean add. The language of the computer is often referred to as **machine code**. A special piece of system software called a **compiler** translates the instructions written in a programming language into machine instructions consisting of 0's and 1's. This process is known as **compiling**. Figure 1.1 illustrates how this process works for many programming languages.

Programming languages have a very strict set of rules that you must follow. Just as with natural languages, this set of rules is called the **syntax** of the language. A program containing syntax errors will not compile. You will see when you start writing programs that the sort of things that can cause compiler errors are the incorrect use of special Java keywords, missing brackets or semi-colons, and many others. If, however, the source code is free of such errors the compiler will successfully produce a machine code program that can be run on a computer, as illustrated.



**Fig 1.1** The compilation process

Once a program has been compiled and the machine code program saved, it can be run on the target machine as many times as necessary. When you buy a piece of software such as a game or a word processor, it is this machine code program that you are buying.

## 1.4 Programming in Java

Before the advent of Java, most programs were compiled as illustrated in figure 1.1. The only problem with this approach is that the final compiled program is suitable only for a particular type of computer. For example, a program that is compiled for a PC will not run on a Mac™ or a UNIX™ machine.

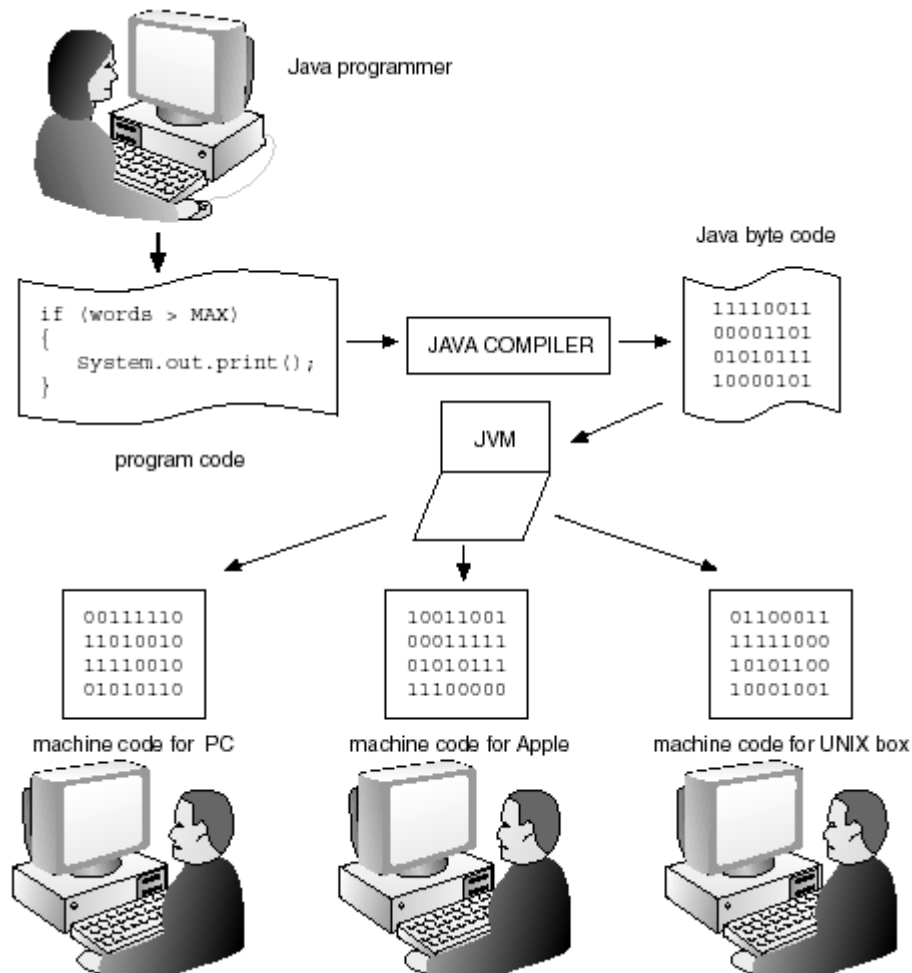
But this is not the case with Java. Java – and nowadays many other languages - is **platform-independent**. A Java program will run on any type of computer.

How is this achieved? The answer lies in the fact that any Java program requires the computer it is running on to also be running a special program called a **Java Virtual Machine**, or **JVM** for short. This JVM is able to run a Java program for the particular computer on which it is running.

For example, you can get a JVM for a PC running Windows™; there is a JVM for a MAC™, and one for a Unix™ or Linux™ box. There is a special kind of JVM for mobile phones; and there are JVMs built into machines where the embedded software is written in Java.

We saw earlier that conventional compilers translate our program code into machine code. This machine

code would contain the particular instructions appropriate to the type of computer it was meant for. Java compilers do not translate the program into machine code – they translate it into special instructions called **Java byte code**. Java byte code, which, like machine code, consists of 0's and 1's, contains instructions that are exactly the same irrespective of the type of computer – it is *universal*, whereas machine code is specific to a particular type of computer. The job of the JVM is to translate each byte code instruction for the computer it is running on, before the instruction is performed. See figure 1.2.



**Fig 1.2.** Compiling Java programs

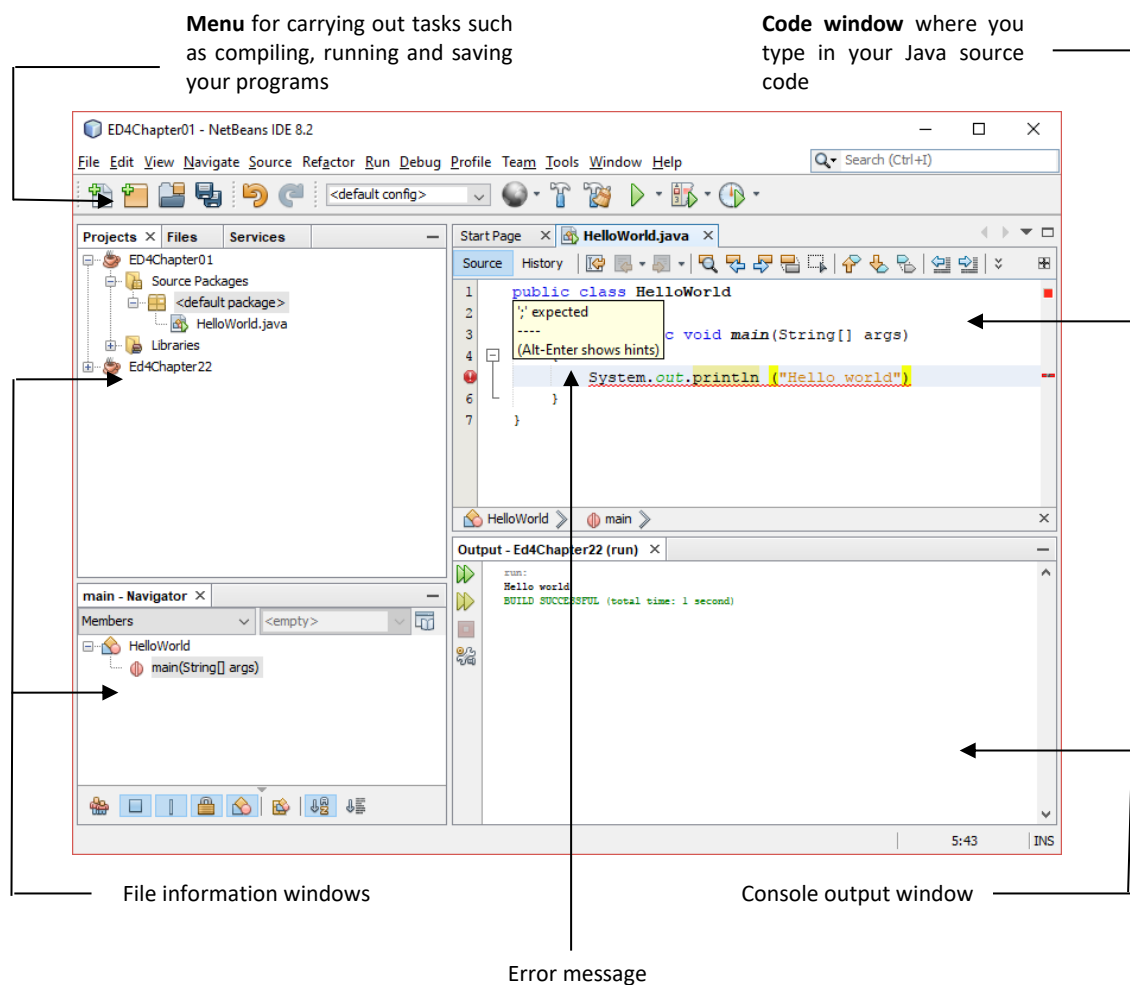
There are various ways in which a JVM can be installed on a computer. In the case of some operating systems a JVM comes packaged with the system, along with the Java **libraries**, or **packages**, (pre-compiled Java modules that can be integrated with the programs you create) and a compiler. Together the JVM and the libraries are known as the **Java Runtime Environment (JRE)**. If you do not have a JRE on your computer (as will be the case with any Windows™ operating system), then the entire Java Development Kit (JDK), comprising the JRE, compiler and other tools, can be downloaded from Oracle™, the owners of the Java platform<sup>1</sup>.

<sup>1</sup> The original developers of Java were Sun Microsystems™. This company was acquired by Oracle™ in 2010.

## 1.5 Integrated Development Environments (IDEs)

It is very common to compile and run your programs by using a special program called an **Integrated Development Environment** or **IDE**. An IDE provides you with an easy-to-use window into which you can type your code; other windows will provide information about the files you are using; and a separate window will be provided to tell you of your errors.

Not only does an IDE do all these things, it also lets you run your programs as soon as you have compiled them. Depending on the IDE you are using, your screen will look something like that in figure 1.3.



**Fig 1.3.** A typical Java IDE screen

The IDE shown in figure 1.3 is NetBeans™, a very commonly used compiler for Java – another widely used

IDE is Eclipse™. Instructions for installing and using an IDE are on the website (see preface for details).

It is perfectly possible to compile and run Java programs without the use of an IDE - but not nearly so convenient. You would do this from a command line in a console window. The source code that you write is saved in the form of a simple text file which has a `.java` extension. The compiler that comes as part of the JDK is called `javac.exe`, and to compile a file called, for example, `MyProgram.java`, you would write at the command prompt:

```
javac MyProgram.java
```

This would create a file called `MyProgram.class`, which is the compiled file in Java byte code. The name of the JVM is `java.exe` and to run the program you would type:

```
java MyProgram
```

To start off with however, we strongly recommend that you use an IDE such as NetBeans™ or Eclipse™.

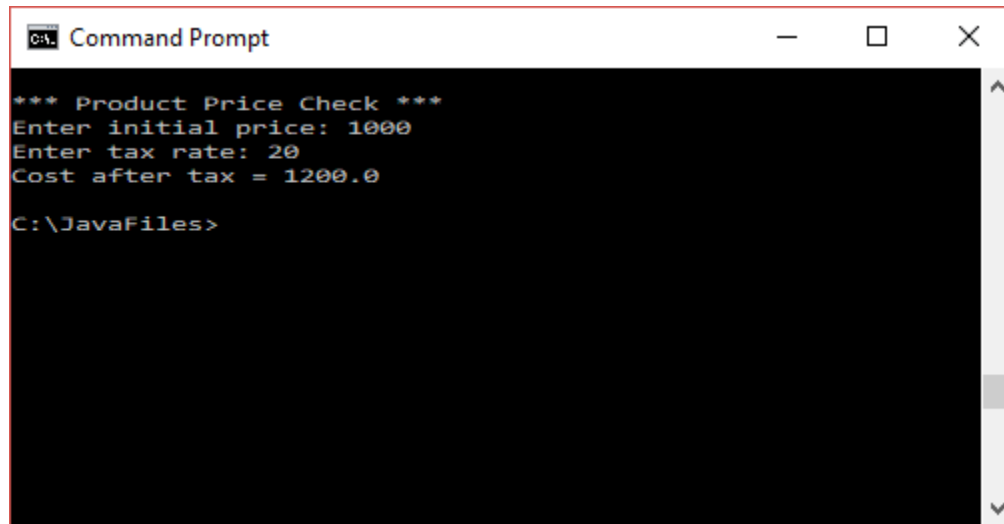
## 1.6 Java applications

As we explained in section 1.2, Java applications can run on a computer, on such devices as mobile phones and games consoles, or sometimes can be embedded into an electronic device. In the last case you would probably be unaware of the fact that the software is running at all, whereas in the former cases you would be seeing output from your program on a screen and providing information to your program via a keyboard and mouse, via a touch screen, or via a joystick or game controller.

The screen that provides output from your program, and prompts you to enter information, is known as the **user interface**. There are two principal types of user interface:

- **text** based;
- **graphics** based.

With text based user interfaces, information is displayed simply as text – with no pictures. Text based programs make use of the keyboard for user input. Text based programs are known as **console applications**. If you are using an IDE, the console window is usually integrated into the IDE as you saw in figure 1.3. However, if you are running a program from the command prompt you will see a window similar to that shown in figure 1.4.



```
*** Product Price Check ***
Enter initial price: 1000
Enter tax rate: 20
Cost after tax = 1200.0
C:\JavaFiles>
```

Fig 1.4. A Java console application

You are probably more accustomed to running programs that have a **graphical user interface (GUI)**. Such interfaces allow for pictures and shapes to be drawn on the screen (such as text boxes and buttons) and make use of the mouse as well as the keyboard to collect user input. An example of a GUI is given in figure 1.5.

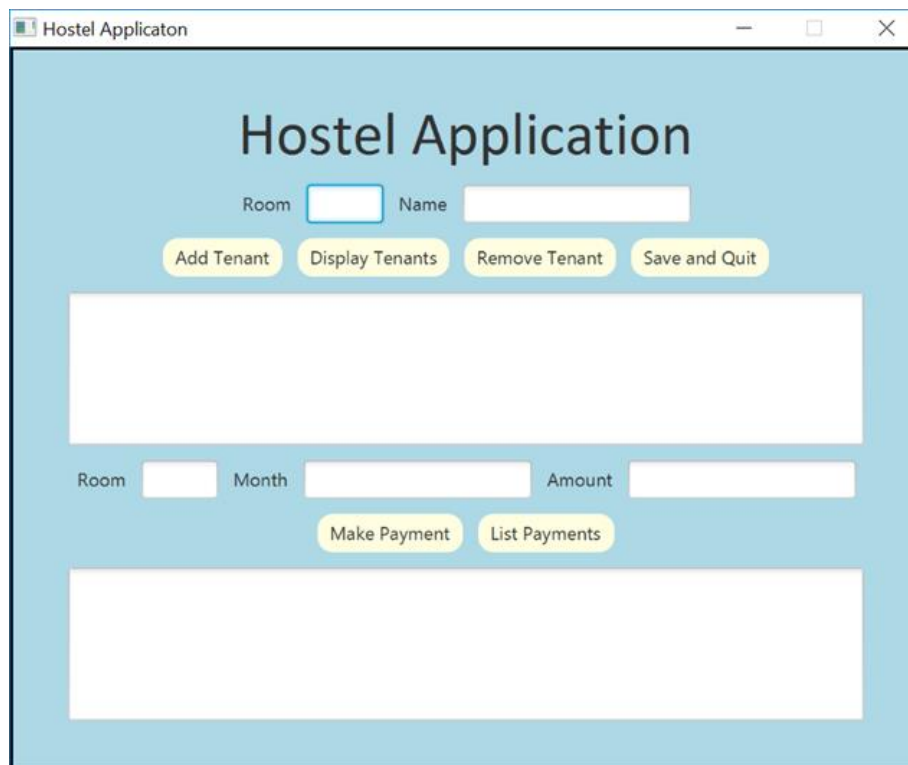


Fig 1.5. A graphical application

Eventually we want all your programs to have graphical interfaces, but these obviously require a lot more programming effort to create than simple console applications. So, for most of the first semester, while we are teaching you the fundamentals of programming in Java, we are going to concentrate on getting the program logic right and we will be sticking to console style applications. Once you have mastered these fundamentals, however, you will be ready to create attractive graphical interfaces before the end of this very first semester.



## Self-test questions

- 1 Explain the meaning of the following terms:
  - program;
  - software;
  - application software;
  - system software;
  - machine code;
  - source code;
  - embedded software
  - compilation;
  - Java byte code;
  - Java virtual machine;
  - integrated development environment;

- 2 Explain how Java programs are compiled and run.

## Programming exercises

- 1 If you do not have access to a Java IDE go to the accompanying website and follow the instructions for installing an IDE. You will also find instructions on the website for compiling and running programs.