2 HELLO WORLD

Terry Marris 7 April 2001

2.1 OBJECTIVES

By the end of this lesson the student should be able to

- write a Java program to display text on the screen
- understand the importance of layout and naming conventions
- understand the use of the Java keywords *class*, *main* and *new*.

2.2 PREVIEW

In Chapter One we saw that a class represents a set of similar objects. In this chapter we see how to write a class in Java that represents a set of similar greeting messages displayed on computer monitor or screen.

We write our first class and test it out in our first Java program. We take a first look at constructors, objects and strings. We see how to print lines of text on the screen. We see what makes a good layout style and how to make our Java programs easy to read and understand.

2.3 THE FIRST JAVA PROGRAM

The first computer program in any language displays some text on the screen or monitor. Our first Java program, shown below, displays *Hello World* on the screen.

```
/* Greeting.java
   Terry Marris  7 April 2001
*/


public class Greeting {
  public static void main(String[] s)
  {
    System.out.println("Hello World");
  }
}
```

We look at the program bit by bit.

2.4 COMMENTS

Comments are written for the benefit of the person who has to read the Java program. Comments are introduced with /* and terminated with */, as shown below.

```
/* Greeting.java
   Terry Marris  7 April 2001
*/
```

Notice that the slashes are forward slashes (and not backward slashes, \) and that the slashes are both the first and the last part of a comment.

This comment tells the reader that the Java program is saved in a file named *Greeting.java* and that *Terry Marris* wrote it on *7 April 2001*.

Every program you write should begin with a comment stating the name of the file, your name and the date you created it.

2.5 CLASSES

Classes are the essential building blocks of Java programs. Every program you write will contain one or more classes. Our program contains just one class.

```
public class Greeting {
   .
   .
   .
}
```

class is a Java keyword.

A Java keyword is a special word that cannot be used for anything but its intended purpose; it is a reserved word. Java keywords include *public, private, static, main* and *void.* We shall be defining and explaining the Java keywords as we come to them. A full list of Java keywords may be found in Appendix A.

class introduces a class defined by the programmer.

Greeting is the name chosen for the class by the programmer. We choose names which

- are descriptive - so that their intended purpose is conveyed to the reader and
- start with an upper case letter - so that wherever the name appears the reader knows it is a class (and not something else) that is being referred to

Spaces and hyphens are not allowed in class names. Class names are also known as class identifiers.

public is a Java keyword. It means *can be used by anything.*

A public class MUST be saved in a file with the same name. For example

```
public class Greeting
```

must be saved in a file named *Greeting.java.*

```
public class FirstProgram
```

must reside in a file named *FirstProgram.java*. *public* class names never contain a *.java* extension.

2.6 BRACES

Braces, { and }, mark the beginning and end of a class. Every opening brace, {, must have a matching closing brace, }.

```
public class Greeting {
.
.
.
}
```

Notice the positioning of the class braces. We use the following convention:

- the opening brace is on the same line as the class name and
- the closing brace is directly in line below the *p* in *public class*
- both the *class* keyword and its closing brace are hard up against the left hand margin

By consistently using the same layout rules we

- make it harder to inadvertently leave out a brace and
- make it easier to find where we have left out a brace if we have done so

Good programmers write neat and tidy Java as they go along.

2.7 THE MAIN( ) METHOD

A program that runs on a computer is known as an executable program. Every executable Java program must have a method named *main()*. *main* is a Java keyword.

```
public static void main(String[] s)
{
  ...
}
```

A method is responsible for carrying out some task.

main() is a special method because programs start running from this point onwards.

Every *main()* method starts exactly like this

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

public - can be used by any object from any class

static - remains in memory for as long as the program is running

void - the empty type - just like the empty set, it has no members or values.

String[] - a collection of *String* objects. A string is just a sequence of letters, digits or symbols; for example: *007, bond, licensed to score!* is a string.

s - the name of the collection of string objects.

The *main()* method must reside in a public class whose name is exactly the same as its file name (but without the *.java* extension).

Notice how the braces are aligned.

2.8 DISPLAYING TEXT

The line actually responsible for displaying the words *Hello World* on the screen is

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

println() (say printline) prints a string on a new line on the screen.

The text to be printed is enclosed between quotation marks " and " within the brackets (and). The quotes themselves are not printed.

To print a blank line on the screen use

```
System.out.println();
```

And to print quotes themselves, as in *She said: "Hello sailor"* place a backslash before the quotes to be printed like this

```
System.out.println("She said \"Hello Sailor\"");
```

System.out represents the computer's monitor or screen and is part of the standard Java language package. Notice the capital *S* in *System* and the letter ell in *println()*.

2.9 STATEMENTS

Methods usually contain statements. A statement, when executed, usually performs some task, such as displaying text on a screen. Our example main() method contains just one statement, *System.out.println("Hello World");*.

```
public static void main(String[] s)
{
  System.out.println("Hello World");
}
```

We can place as many statements as we like inside a method, as in the example shown below.

```
public static void main(String[] s)
{
  System.out.println("JACK: you're quite perfect, Miss Fairfax");
  System.out.println("GWENDOLIN: Oh!  I hope I am not that.");
  System.out.println("It would leave no room for developments");
  System.out.println("And I intend to develop in many directions");
  System.out.println("Oscar Wilde, The Importance of Being Ernest");
}
```

Each statement must end with a semi-colon. The semi-colon is known as the statement terminator

2.10 INDENTATION

Notice also that whereas classes start hard up against the left hand margin on the page, its methods begin indented by two spaces under the *public class* keywords (just tap the space bar two times, as shown by the ∇ symbol below).

```
public class Greeting {
∇∇public static void main(String[] s)
  {
  ∇∇System.out.println("Hello World");
  }
}
```

And statements are indented by two spaces under the opening brace.

By consistently using the same indentation rules we

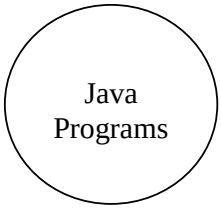
- make it harder to make mistakes
- make it easier to find mistakes

Good programmers write neat and tidy Java as they go along.

2.11 FURTHER READING

HORSTAMANN& CORNEL *Core Java 2 Volume 1 Fundamentals* pp 44

2.12 REVIEW



- classes
  - building blocks of programs
  - class names
    - descriptive
    - begin with upper case letter
  - contain methods
    - a method is responsible for performing a task
    - main() is a special method
      - programs start running with main
    - methods contain statements
      - statement
        - specifies some action
        - terminated with semi-colon
      - text - displayed with System.out.println()
  - comments
    - written for the human reader
    - begin with /* and end with */
  - layout standards
    - alignment
    - indentation
    - minimises errors
    - aids understanding

2.15 EXERCISES

1 Explain the meaning of the following Java words

(a) class
(b) main
(c) new

2 Critically evaluate the following as class names

(a) X
(b) Person
(c) book

3 Critically evaluate the layout of each of the following classes.

(a)
```
class Poem
{  Poem() { System.out.println("Good morning, sunshine");
System.out.println(
"The world says \"Hello\"");System.out.println(
"You shine above us");
System.out.println("We toil here below"); }}
```

(b)
```
class Welcome {
Welcome() {
System.out.println("Welcome to the Billing Program");
System.out.println("By Side Gates");
System.out.println("7 July 2000");
}
}
```

4 Find five errors in each of the two programs shown below, and correct them.

(a)

```
/* Greeting.java
   Terry Marris  7 April 2001



public class Greeting {
  public static void Main(string[] s)
  {
    system.out.println("Hello World")
  }
}
```

(b)

```
/* Greeting.java
   Terry Marris  7 April 2001
/*


Public class Greeting {
  Public static void main(String[] s)
  {
    System.out.println("Hello World);
  }
```

5 Write a Java program that will display the first four lines of a song or poem on the screen. You choose the song or poem.