

JAVA PROGRAMMING – EXERCISE 1

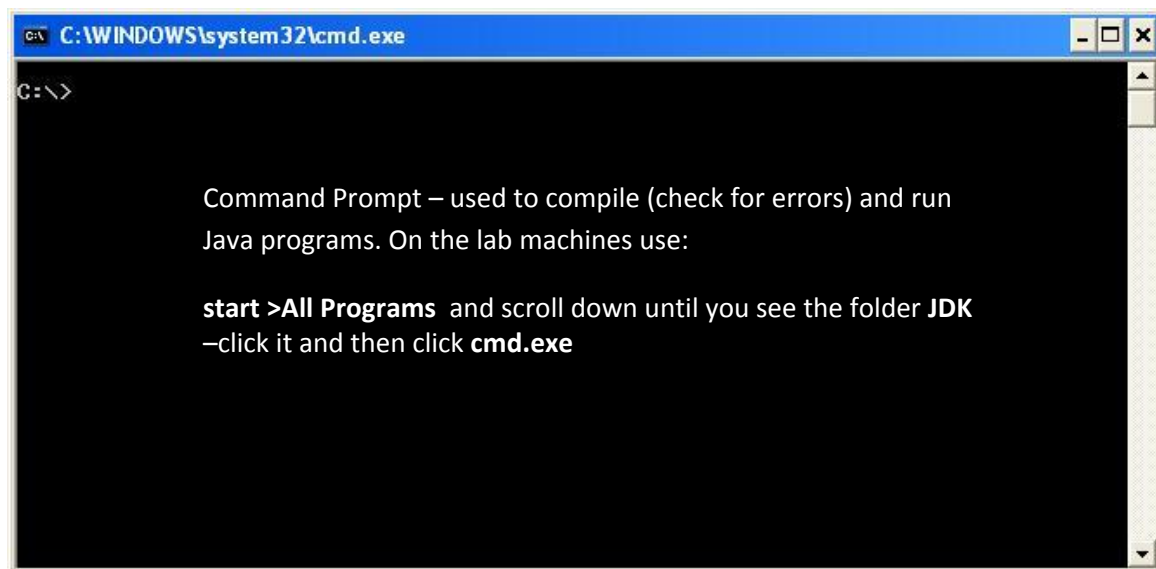
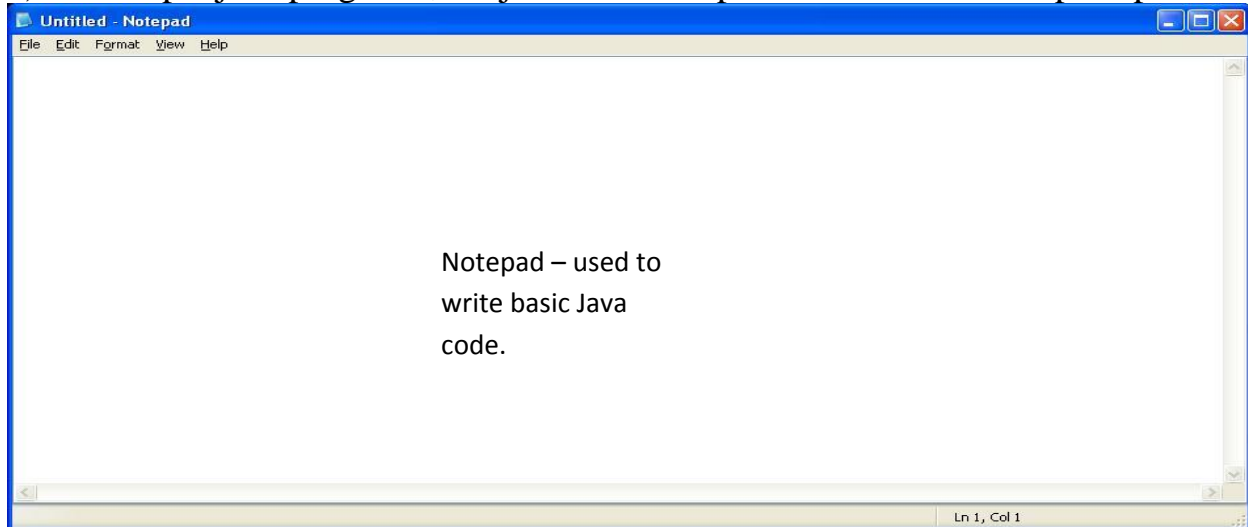
Welcome! Make sure you have completed the Java pre-Lab exercise first.

Please read the following instructions carefully.

Video help? <http://stream.manchester.ac.uk/Play.aspx?VideoId=9574>

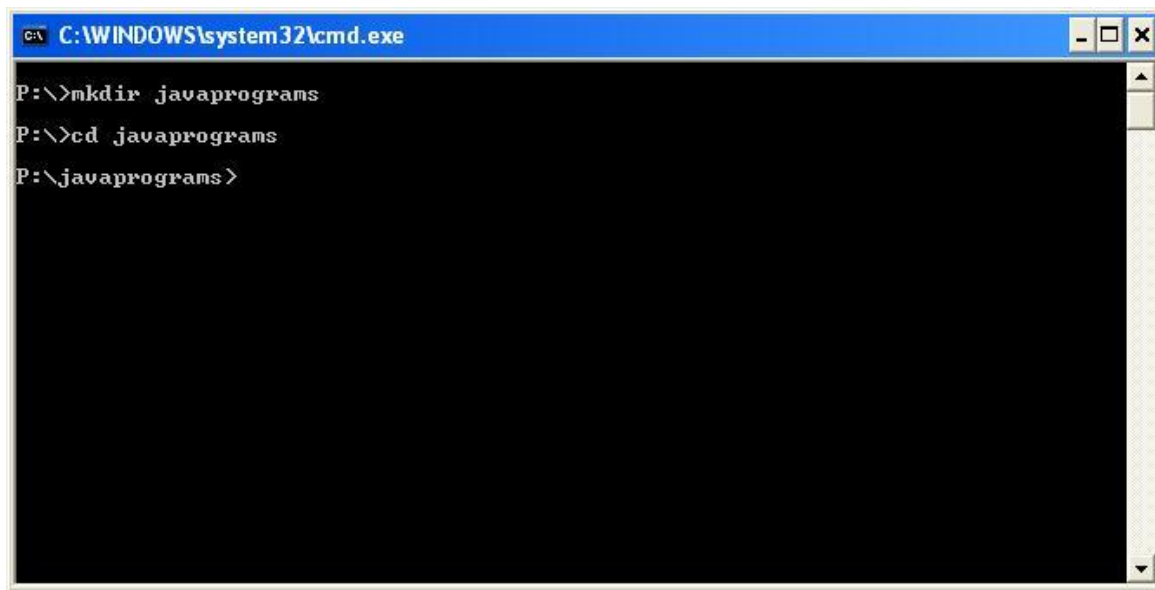
-you may have to type your username and password into the page that loads.

1) For simple java programs, we just need Notepad and the command prompt:



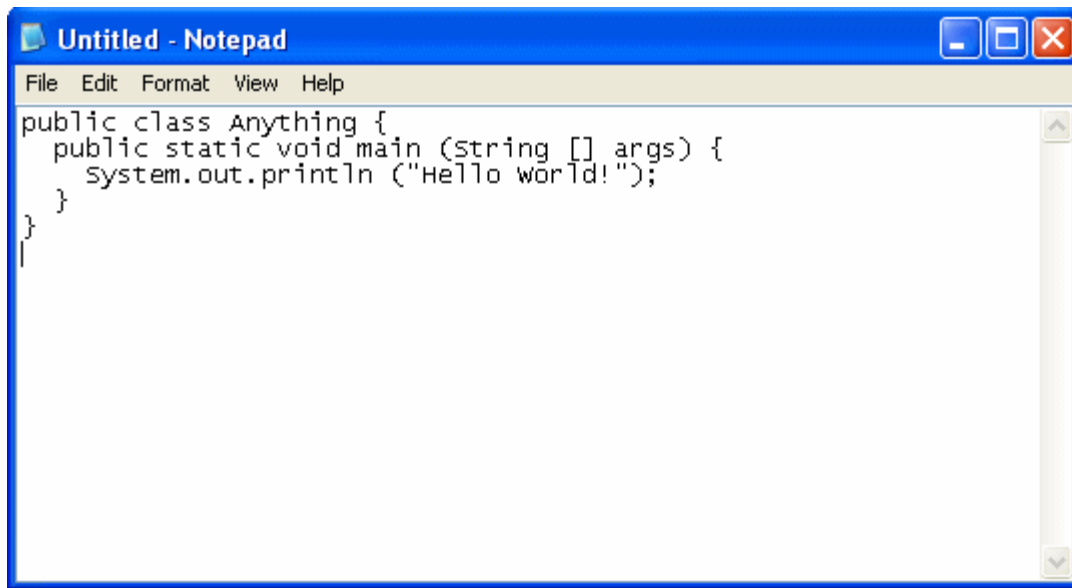
2) On the lab machines make sure you always access the Command Prompt with: **start >All Programs** and scroll down until you see the folder **JDK** –click it and then click **cmd.exe**

Navigate to your P: drive and create a folder called javaprograms as in the following example, and cd into it:

A screenshot of a Windows Command Prompt window. The title bar at the top is blue and contains the text "C:\WINDOWS\system32\cmd.exe" along with standard window control buttons (minimize, maximize, close). The main area of the window is black with white text. The text shows the following commands and their execution: "P:\>mkdir javaprograms", "P:\>cd javaprograms", and "P:\javaprograms>". The cursor is positioned at the end of the last line, ready for the next command.

```
C:\WINDOWS\system32\cmd.exe
P:\>mkdir javaprograms
P:\>cd javaprograms
P:\javaprograms>
```

3) To begin, type **notepad** to start the Notepad editor program, and type in the following Java code exactly as it is in the Notepad editor window:



```
public class Anything {  
    public static void main (String [] args) {  
        System.out.println ("Hello world!");  
    }  
}
```

Java is *case sensitive*, that means words like ‘class’, ‘String’, ‘public’, ‘static’, ‘System’, etc must be typed out exactly as they appear. If you type in ‘system’ instead of ‘System’, it’s an error. Notice how some lines are indented.

4) Before we save or make any changes to the code lets first consider what it means.

public class Anything

Every Java program needs a class. This serves as a container for your Java code, and is absolutely essential. *Nothing works without a class*. We’ve chosen to name it ‘Anything’ (it is convention to capitalize the first letter of the class name).

public static void main(String [] args)

You need this line whenever you write a Java program, so learn it! This defines the start of the main method. Whenever a Java program is run, your computer checks for the presence of main and runs the rest of the program from there.

As a quick explanation of the other parts of the definition for main:

- ‘public’ essentially means that this method is visible and usable by all parts of the program.
- ‘static’ means it belongs to a class (in this case, it belongs to class ‘Anything’).
- ‘void’ means the main method does not return a value.
- ‘String [] args’ is used by the command prompt.

These will be explained more fully in the lectures.

System.out.println("Hello world!");

This line is the only part of the whole code which isn't compulsory.

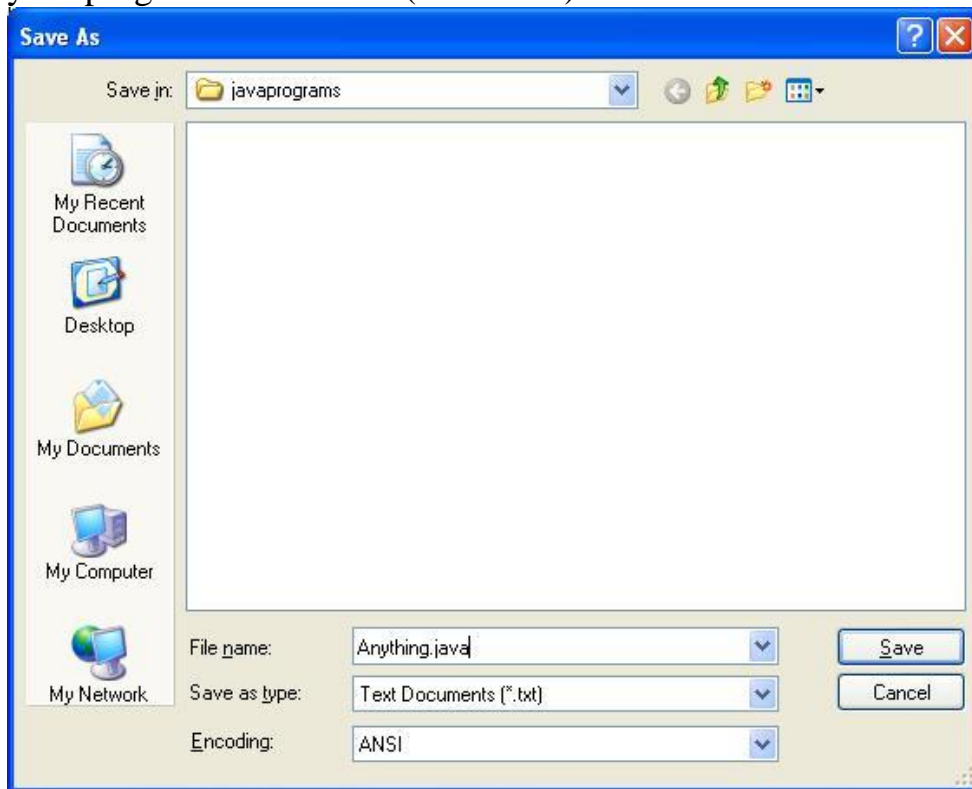
It translates to print the line "Hello world!". You can type in anything you want inside those double quotes. The semi colon at the end indicates a complete line of code.

The curly brackets { }

The curly brackets enclose sections of the code in the class.

If you look at the code, you can see 'class Anything' encloses the whole program, and the 'main' method encloses just the 'System.out.println("Hello world!");' line.

5) Once you've typed in the program, from within Notepad click on File -> Save As, and, making sure you save it in the same javaprograms folder, save it as 'Anything.java'. The filename must be the same as your class name, otherwise your program will not run (see below):

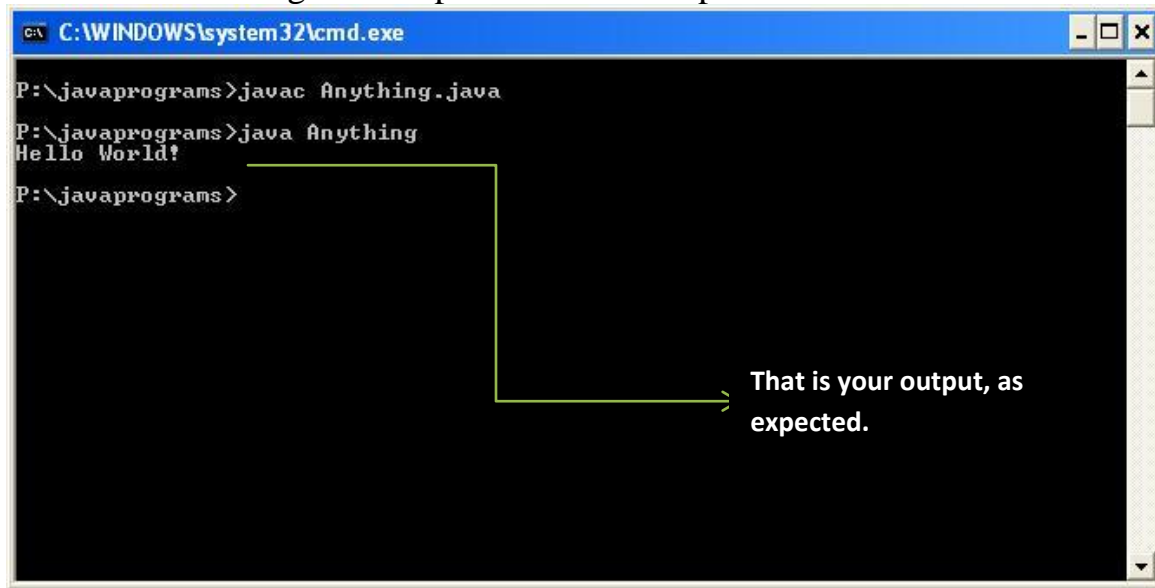


The class name and the filename should be identical.

6) Go back to the command prompt and type dir. You should see Anything.java in the list.

7) Next, we have to compile the program. Type in "javac Anything.java" without the quotes and press Enter. If nothing happens, you have no errors and you can proceed. If an error does occur, please ask a demonstrator for help.

8) If there are no errors, you can assume that it has compiled properly. To run the program, simply type in “java Anything” (again without the quotes) and press Enter. You should get an output similar to the picture below:



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The prompt is at "P:\javaprograms>". The user has entered "javac Anything.java" and pressed Enter. The prompt is now "P:\javaprograms>". The user has entered "java Anything" and pressed Enter. The output "Hello World!" is displayed on the next line. A green arrow points from the output "Hello World!" to the text "That is your output, as expected." which is written in a white font on the right side of the window.

```
C:\WINDOWS\system32\cmd.exe
P:\javaprograms>javac Anything.java
P:\javaprograms>java Anything
Hello World!
P:\javaprograms>
```

That is your output, as expected.

Congratulations on running your first Java program. Now let's change our code a bit.

9. Next type notepad “Anything.java” to open up the file again. Below the “System.out.println(“Hello world!”);” add another line saying System.out.println(“My name is Adam.”);

Now your code should look like,

```
public class Anything {

    public static void main(String [ ] args) {
        System.out.println("Hello world!");
        System.out.println("My name is Adam.");
    }

}
```

Do File>Save to save the changes, and as before, compile using “javac Anything.java” and run by typing out “java Anything”. What does the output look like?

Expected output: Hello world!
My name is Adam.

Go back to notepad, and change the “println” part of both lines to just “print” as given below :

```
System.out.print("Hello world!");  
System.out.print("My name is Adam.");
```

Save your file, compile and run it again.

Expected output: Hello world!My name is Adam.
Notice the difference?

Similarly, try the following changes and make sure you understand each one of them:

JAVA STATEMENTS	EXPECTED OUTPUT
System.out.println("Hello world!"); System.out.print ("My name is Adam.");	Hello world! My name is Adam.
System.out.print ("Hello world!"); System.out.println ("My name is Adam.");	Hello world! My name is Adam.
System.out.print ("Hello world!"); System.out.print ("\nMy name is Adam.");	Hello world! My name is Adam.
System.out.print ("Hello world!"+"My name is Adam.");	Hello world!My name is Adam.
System.out.print ("Hello world!"+" \nMy name is Adam. ");	Hello world! My name is Adam.
System.out.print ("Hello world! \nMy name is Adam. ");	Hello world! My name is Adam.
System.out.print (5+2);	7
System.out.print ("Hello world!"+"My name is Adam and I'm "+22);	Hello world!My name is Adam and I'm 22
System.out.print ("Hello world!"+"My name is Adam and I'm "+22+2);	Hello world!My name is Adam and I'm 222
System.out.print ("Hello world!"+"My name is Adam and I'm "+(22+2));	Hello world!My name is Adam and I'm 24
System.out.print (5*2);	10
System.out.print ("Hello world!"+"My name is Adam and I'm "+(5*4));	Hello world!My name is Adam and I'm 20
System.out.print ((2*3)+(7*2));	20
System.out.print ((2+3)*(7+2));	45

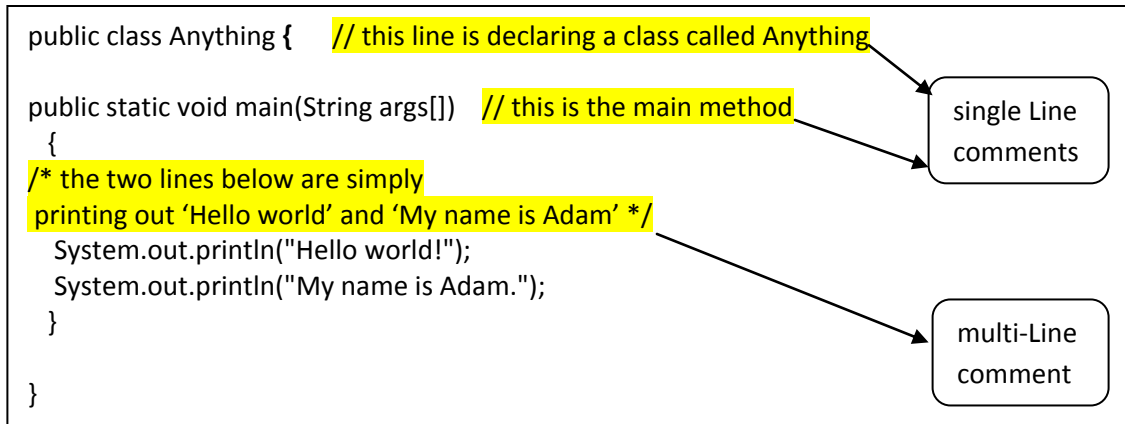
Explanatory Notes:

- The '+' sign is used to combine two or more pieces of text and/or numbers.
- Anything inside those double quotes is treated as a piece of text. So if you type in "2"+"2", it will just be put together and printed out as 22. But if you type in 2+2 (without any double quotes), the output will be 4.
- You can perform various mathematical operations on two or more numbers by utilizing the signs:
+ (Addition), -(Subtraction), / (Division), and * (Multiplication).
- The '\n' character is an example of an escape sequence. There are quite a few escape sequences, and each one has a special meaning to the java compiler. Here are the major escape sequences,

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

- You can add notes to your java code by using // (for a single line comment) or /* and */ (for opening and closing a multi-line comment.)

Example:



When writing programs it is important to realise that someone else might read your code, and to help them understand it comments are used. Comments allow you to include ‘notes’ in your program. They are used to document the code, to explain the purpose of each section.

Comments are ignored by the compiler.

Task 1:

Add statements to output the following table

Hint: use tabs (\t) to get those big spaces between columns, and print out quotes using \"

Name	Age	Course
J.Smith	19	“EEE”
P.Jones	20	“EEE”
S.Carey	18	“EE”
F.Hogg	19	“MTE”

Task 2:

Add a *single* println statement that displays the following output:

Hint: use the escape sequence '\n'

**I
study
in
Manchester**

Task 3:

Add statements to print out the results of

18*3

-21-6

12*-5

Task 4:

What do you expect the output to be from the following line (test with your program)?

System.out.println("The number is: " + 4 + 6);

Task 5:

Modify the above println statement to force the output to print

The number is: 10

Task 6:

Add a line to print out the average of the following numbers: 12, 8, 19, 5, 17, 14. Make sure the output begins with "The average is: ".

Task 7:

Add a line to print out the following statement exactly as it appears with quotes:

The path to my Java folder is:

"C:\users\fred\javaprograms"

END