COM1003 Java Programming - Autumn Semester 2017-8

Programming Exercise Sheet 1

Dr Siobhán North

Department of Computer Science, University of Sheffield

Learning outcomes

Having completed these exercises you will:

- Understand how to type in a simple Java program using the jEdit application;
- Understand how to compile and run your Java program.
- Understand how to correct errors in a small Java program.
- Understand how to use variables, types and expressions in Java, in order to carry out simple calculations.

Writing and saving a Java program

To write and run your Java programs I suggest you use jEdit, which is essentially a text editor with extras. It is an open source program which you can download and install on your own computer at home if you want to. There is a link to it from the web page.

- Click on the Windows icon at the bottom left of the PC screen and select **jEdit** from the alphabetical list of programs then select **jEdit** again at the next level.
- In jEdit select **File → New** to create a new Java file.
- Type in your program (the following assumes that you will type in the **`Simple.java`** program shown on the next page).
- Create a new directory to save it in. Select **File → Save As.** This will bring up a file browser. The first line will start "Path:". Click on the text next to it and replace it by **U:**. Now click on the rightmost icon on the line above, it is a picture of a folder with a little star on it. That creates a new folder on the U: drive. You will be asked to give it a name. For the purposes of these notes I will assume you have chosen to call it **myJava**. The **myJava** folder will appear in the list of files. Double click on it to open it. Next time jEdit will remember this path. Now you can save your file. At the bottom of the screen is a box labelled "File Name:". Here you type in the name of your program ***with the file extension*** **`.java`**, e.g. **`Simple.java`**. It is important that the file name ***exactly*** matches the name of the class including the case of each letter.
- All the programs used in the lectures are available from the course web pages so, to save typing, you could also copy **`Simple.java`** from the web page, paste it into the jEdit screen and edit it or download it to your **myJava** folder and then open it in jEdit (**File → Open).**

Compiling and Running a Java program

- Leaving jEdit open because you will need it again click on the Windows icon at the bottom left of the PC screen again then select **Java JDK** from the alphabetical menu and then **Command Prompt with JDK**. A new black window will open with the

prompt

```
u:\>
```

Type

```
cd myJava
```

That will change the prompt to

```
u:\ myJava>
```

You only need to do this once per session.

- At the prompt compile your java program:

```
u:\myJava> javac Simple.java
```

- And run it:

```
u:\myJava> java Simple
```

The first time you compile and run a Java program it may take some time, since the computer runs a virus check on Java libraries that are copied to your file space. Subsequently it should be very quick. Also, note that you can use the up and down arrow keys to scroll through previous commands in the terminal window. This can save you a lot of typing. Leave the Java JDK window open too because you will be switching between it and the JEdit window for the rest of the session.

Exercise A

Type in the following program, replacing ******Name****** with your name and ******Date****** with today's date, and save it in a file called **Simple.java** as described above.

```
/* A simple Java program
* Written by: ***Name***
* First written: ***Date***
* Last updated: ***Date***
*/
public class Simple {
  public static void main(String[] args) {
    System.out.println("Hello ***Name***.");
  }
}
```

Include comments such as these in every program you write.

Exercise B

The following program contains syntax errors (and is not particularly well laid out).

```
public class Exercise1b
{ public static void main(String[] args) {
System.out.print ( " Hello " )
System.out.println ( world." );
}}
```

Type it in (correcting the layout as you go) and save it in a file called **Exercise1b.java**. The text colouring scheme jEdit uses will help you spot one of the errors without even running it. Correct that error and then run it to see what the compiler does. Now correct the program. Put your hand up to get one of the demonstrators to check you have got the layout right.

Exercise C

Everyone knows that two plus two equals four. Write a program to prove the computer also knows it so get the computer to work out and print the answer to 2 + 2.

Exercise D

Explain why the following program generates an error when compiled. Correct the error in the program (without changing the type of **z**), then compile it and run it.

```
public class Exercise1d {
  public static void main(String[] args) {
    int z = 5/3.0;
    System.out.println(z);
  }
}
```

Exercise E

What is contained in the variable **x** after execution of the following Java statements?

```
int a=4, b=9, c=3, d=2;
int x=a+b/c*d-a*c/2;
```

Write a Java program to verify your answer.

Exercise F

Write a program in which you declare a constant called PI whose value is 22 ÷ 7 and a variable called radius and give it a value. Make your program print out the value of your radius, the value of the circumference of a circle with that radius (twice the radius times ∏) and the area (∏ times the square of the radius).

Don't just print out the numbers concerned, print out messages saying what they are so instead of just using the line

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

Put a message in front like this

```
System.out.print("The radius is ");
System.out.println(radius);
```

All your programs should be like that in future, they should not only print out a value, there should be an explanation too.

You can simplify the program by replacing the two lines above by

```
System.out.println("The radius is " + radius);
```

What does the plus sign do?

Continued

Exercise G

This is a brief introduction to some more advanced material. Change the print statement in the **`Simple.java`** program to:

```
System.out.println("Hello " + args[0]);
```

Compile the program as usual but run it with

```
U:\myjava> java Simple XXXX
```

except replace XXXX with your own name.

Try running the program again in the usual way without your name

```
U:\myjava> java Simple
```

and you will get an error message.

What is the print statement doing? We will return to the notation **`args[0]`** at a later date.

Exercise H

As a new undergraduate student you will soon appreciate that one of the most important equations you will learn is this one, which calculates the monthly payment for an amortized loan (i.e., a loan which is repaid in fixed periodic instalments, each of which includes interest and principal). The equation is:

$$m = \frac{\left(\frac{p}{1200}\right)\left(1+\frac{p}{1200}\right)^{n}}{\left(1+\frac{p}{1200}\right)^{n}-1}c$$

This calculates the monthly payment ***m*** for a loan on a principal sum of ***c*** pounds, paid back over ***n*** months at an annual interest rate of ***p*** percent.

Which of the following is a correct Java implementation of this equation? N.B. **`Math.pow(x,y)`** means x^y

i) `m = c*p/1200*Math.pow(1+p/1200,n)/Math.pow((1+p/1200),n)-1`

ii) `m = (p/1200*Math.pow(1+p/1200,n)/Math.pow(1+p/1200,n)-1)*c`

iii) `m = c*p/1200*Math.pow(1+p/1200,n)/(Math.pow(1+p/1200,n)-1)`

iv) `m = c*(p/1200*Math.pow(1+p/1200,n))/Math.pow(1+p/1200,n)-1`

Write a Java program to verify your answer and test it using some made up numbers.

Your answer will probably be displayed to far too many decimal places. There are ways around this that we will cover next week but if you want an extra task you could try working out how many whole pence in your answer and then print out the number of pence divided by 100 to ensure there are at most two decimal places.