Software Workshop

Welcome Week

Exercise 1

Log in on one of the lab machines with your login details.

Open a terminal window by clicking 'Activities' and then by clicking the icon that looks like a computer terminal.

In the terminal enter the following command:

```
module load java
```

This configures your environment for Java. You will need to type this every time you open a new terminal.

Exercise 2

In the terminal type '`pwd`'.[1][2] The terminal will respond with your 'present working directory' - the folder (directory) you are currently in. This will be your 'starting' folder whenever you log in or open new terminal. It is called your *home* folder.

Type '`ls`'. This command will list the folders in your current folder.

You should see a folder called 'work'. This is the folder in which you should place your university work (when storing it on the university system). It is easier when working with commands to change to the folder in which you want to work. To change into the 'work' folder type:

```
cd work
```

If you now type '`ls`' again, this folder should be empty (no documents or other folders will be listed). Type '`pwd`' again to see where you are in the folder hierarchy.

If you want to return to the folder 'above' the current folder (one step above the current folder in the hierarchy) then type '`cd ..`'. If you want to return to your home folder then simply type: '`cd`'.

You should now be in the 'work' folder. Create a new folder called 'welcomeweek' in that folder by typing:

```
mkdir welcomeweek
```

[1] Unless otherwise stated, quotation marks should not be typed in. They are used in the text to distinguish commands from the rest of the text.

[2] Press enter after each command.

This will create a new folder called 'welcomeweek'[3]. If you don't like the capitalisation of that folder name then you may change it but **be aware** that Linux is *case sensitive* so 'welcomeweek' and 'welcomeWeek' (for example) are two different folder names.

Finally, change into the new folder that you have just created using the commands you have learned so far.

Tip: when using `cd` you only need to type enough of the folder name to distinguish it from other folders in the folder you are in. When you have done that, you can press the tab key to automatically complete the rest of the command.

Exercise 3

We are now going to create the traditional 'Hello, World' program. This exercise assumes you are in the folder just created for Exercise 2.

At the terminal, type:

```
gedit HelloWorld.java
```

This will start a text editing program called `gedit` and open a new file called `HelloWorld.java` (you may see an error in the terminal window. You can ignore this).

In the text editor of `gedit`, type in the following program exactly as you see it here between the lines

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

Listing 1: HelloWorld.java

Save your program.

Note that the name of the file **must** be the same as the name of the *class* (which is the string of text after `public class` on the first line of the program), with the extension `.java`.

Exercise 4

We have so far created a text file containing some Java *source code*. This is not yet a program that the Java system will understand. We need to *compile* the program and then *run* it.

Firstly, open a **new terminal**. In that terminal, navigate to the folder that contains your HelloWorld.java file.

To *compile* the program, type the following in **your new terminal**:

[3]It is rarely, if ever, necessary to include spaces in a folder name. I counsel against it, always.

```
javac HelloWorld.java
```

`javac` is a program that compiles Java source code. It produces a file with the same name as the class but with a `.class` extension. In this case it will produce a file called HelloWorld.class.

If you have typed the program correctly, the terminal will respond by simply waiting for further input. You will not be congratulated by the compiler ...

If you have not typed the program correctly you will be told what the errors are. There's a chance these errors won't mean much to you right now so just look again at where you may have made a mistake in typing the program. You will also get an error if the file name is incorrect, as discussed in Exercise 3. You will also get an error (but not from the compiler) if you are not in the folder you think you are and `javac` cannot find your file.

To run your program, in the terminal type

```
java HelloWorld
```

If you have done everything correctly, your terminal should respond with

```
Hello, World.
```

Exercise 5

Create a new program called `HelloMe` in which instead of printing 'Hello, World' the program prints 'Hello, a.' where 'a' is your student ID. The instructions for this are the same as above except for the change of class name, change of file name and change of output. This will, of course, change the command arguments too (for example, you will need to type '`javac HelloMe.java`' to compile the program).

Exercise 6

Create a new program that prints the words "I will do LOTS of programming practice." to the screen five times. It is a known fact that you only need to read these words five times in order to comply. You can run this program when you feel you are flagging at any point.

Note that whatever you call your class, you must follow the rule that the file must have the same name but with the extension `.java`. Note also that you are not allowed spaces in Java class names (hence file names).

Exercise 7

Create a new program called `HelloYou` that accepts a command line argument, which will be the user's name. The program will greet the user. If the user types

```
java HelloYou Terrence
```

The program will respond with

```
Hello, Terrence.
```

To do this you need to know how command line arguments are passed to Java programs. Command line arguments are passed as *strings* and are placed in the *array* called `args` that you can see in any of the programs written so far. If only one argument is passed to the program, it will appear in `args[0]`. The only other thing that you need to know is how to concatenate strings in Java (see if you can look that up).

Exercise 8

(Advanced)

Create a command line program that converts between Fahrenheit and Celsius and vice versa. The program should accept two command line arguments. The first argument will be either `f` or `c` and indicates which temperature scale the *output* temperature should be in. The second argument will be an integer which will be the temperature to convert to the output scale.

For example, if the user types `f 20` they wish to convert 20 degrees Celsius to Fahrenheit. If they type `c 100` they wish to convert 100 degress Fahrenheit to Celsius.

The result of the conversion should be printed to the screen with a suitable message.

This program requires the use of a few more techniques. It requires the use of some programming constructs that may be new to you so, if necessary, you can come back to this question when you have learned more. If you are attempting it straight away, you will need some kind of `if else` construct; a method for converting a string to an integer; and a conversion formula for the temperature scales.