Worksheet One

"You know", said Arthur, "It's at times like this, when I'm trapped in a Vogon airlock and about to die of asphyxiation in deep space that I really wish I'd listened to what my mother told me when I was young". "Why, what did she tell you?" replied Ford. "I don't know I didn't listen!"
Douglas Adams, The Hitch Hikers Guide to the Galaxy 1979

Objectives: to write simple pseudo code algorithms using modularity.
to create, compile and run simple Java programs.
to use submodules in Java.

It is unlikely you will finish this worksheet in the time allocated, so you will need to do so in your own time. See the handout "Setting up for other O/Ss" so you can work on your own computer.

Exercise One (Setting up Linux)

Login to the computer

Open a terminal window

Create a directory called OOPDWorksheets on your flash drive.

Create 7 subdirectories, P01 to P07

Check the CLASSPATH

To be able to compile and run Java programs you may need to modify a special file in your home directory. The file is called `.bash_profile`. (Because its name starts with a dot, you will not see it using the command for listing files (i.e. `ls`) unless you use the –a option (i.e. `ls –a`)).

in the terminal window type `echo $CLASSPATH`

```
/usr/java/latest/lib:/usr/units/st151/classes:.
```

If not set, use the editor to add this line at the bottom of the file - do not edit anything else!

```
export CLASSPATH="/usr/java/latest/lib:/usr/units/st151/classes:."
```

Save and close the file, then at the command prompt, type:

```
source .bash_profile
```

Find out how to run gedit (or vim) - this will be used to create and edit the source code.

Exercise Two (your first Java program)

Translate the algorithm below into a Java program. Use the examples in lecture one as a guide. Write your java program in your exercise book and then use the editor to create a file called MyFirstJavaApplication.java in your P01 directory.

```
INPUT integerOne and integerTwo
sum = integerOne + integerTwo
diff = | integerOne – integerTwo |
OUTPUT sum and diff
```

Your tutor will explain how to do absolute value in Java, other than that the lecture one examples have all the information you need. Once you have created the java program, try to compile it using the command:
`javac MyFirstApplication.java`

If it does not compile successfully then ask your tutor for help in correcting the errors. Once you have corrected the error then try compiling it again.

Once it has been successfully compiled then you can try running it using the command:

```
java MyFirstApplication
```

Exercises continue on the next page

Exercise Three (Developing ground truth data)
The formula below is for converting a temperature from Fahrenheit to Celsius.

```
Celsius=((Fahrenheit-32.0)×5.0)/9.0
```

Using a calculator, generate 5 Fahrenheit/Celsius pairs by hand. These values will be used to test your solution to exercise four.

Exercise Four (Algorithm design)
Design, in pseudo code, an algorithm which converts degrees Fahrenheit to degrees Celsius. An amount in Fahrenheit is input, converted to Celsius and then output to the user.

Exercise Five (practice with Java implementation).
Translate your algorithm into Java and then compile and test it. What data type should you use? It is probable that your program will not compile. Examine any compiler error messages carefully. Keep correcting any errors until you successfully manage to compile your Java application. Run it using the Fahrenheit/Celsius pairs that you generated for exercise three. Does you Java application generate the same answer? If it does not then seek guidance from your tutor.

Exercise Six (practice with using sub modules).
Finish the algorithm below by designing the sub modules and adding the IMPORT/EXPORT information to the sub module calls in the main algorithm below. The algorithm inputs a measurement in pounds and converts it to a measurement in kilograms. Please note that the MAJOR point in this exercise is to gain some familiarity with the use of sub modules. Use the inches to centimetres example from the lecture notes to guide you. MAKE SURE your MAIN algorithm matches the one below including putting the output in a submodule. Note: there are 2.204 pounds to 1 kg**.**

```
MAIN
   INPUT pounds
   convertPoundsToKilos
   outputPoundsAndKilos
END
```

Exercise Seven (practice with using sub modules in Java).
Convert your pseudo code from exercise four into Java. Place it in a file called PoundsAndKilos.java. What does the name of the Java file imply in terms of the name of the class inside the file?

Exercises continue on the next page

Exercise Eight (Another algorithm with sub modules)
Electric powered radio controlled model aircraft these days make use of Lithium Polymer (Lipo) batteries for their power source. While light and powerful compared to other types of batteries they are also renowned for exploding if not properly used. One sure way to get a Lipo battery to ignite is to attempt to draw current from it at a rate that exceeds its capacity. The maximum rate that a Lipo battery can supply power is called a C-Rating. A C-rating is calculated using the formula below:

$$CRating = \frac{amps \times 1000}{mAh}$$

Where amps is the maximum draw on the battery and mAh is the battery size.

Another sure way to set fire to a Lipo is to drain it completely. In fact the voltage levels in each Lipo battery cell should never be allowed to fall below 3.2 volts.

The flight time of a model aircraft (powered at full throttle for the entire flight) is calculated using the formula below:

$$Time = \frac{60}{CRating}$$

Where time is the flight time in minutes.

Given that the model will not be run at full throttle all the time, this time limit will include a reasonable safety factor. Lipo batteries are described by the number of cells, the mAh capacity and the C-Rating. Hence knowing the flight time and purchasing a battery with a high enough C-Rating will ensure no exploding model planes.

Design, in pseudo code, an algorithm which will input amps and mAh from the user and output the CRating and flight time to the user. Think carefully about the best way to employ sub modules in your algorithm. Your main algorithm should consist of only sub modules calls and, other than the main sub module, you should have at least three other sub modules. Note you may assume that the user input is always valid.

You can test your algorithm on the following inputs:

Amps	mAh
15	1800
8	800
27	2250
35	4000

Exercise Nine (Another Java implementation)

Convert your pseudo code design into a complete Java application.