COM1003 Java Programming - Autumn Semester 2017-8

Programming Exercise Sheet 2

Dr Siobhán North

Department of Computer Science, University of Sheffield

Learning outcomes

Having completed these exercises you will:

- Have downloaded the **`sheffield`** package;
- Understand how to use the **`sheffield`** package for input and output;
- Understand how to display numbers in a various formats using the **`sheffield`** package;
- Have used GUIs for both input and output;
- Have some experience of manipulating text strings and characters in Java.

Exercise A

Download the **`sheffield`** package (from the same page as this exercise sheet amongst other places). Install the package as instructed, and download and run one of the programs from the week 2 lecture.

Exercise B

Write a program that reads a distance as an integer number of miles and then an integer number of yards and converts it into kilometres. A mile is 1,760 yards and 1.60934 kilometres. The result should be written to two decimal places. Here is a typical run of the program:

```
Enter a distance in miles and yards
How many miles? : 123
How many yards? : 567
123 miles and 567 yards is equivalent to 198.47km
```

Don't forget to make good use of constants. If you are not sure you have used them correctly ask a demonstrator to check your program.

Exercise C

Write a program that reads two decimal numbers from a text file, representing amounts in pounds sterling. You should write the text file in a simple text editor such as Notepad (although jEdit will also work) and save it with an appropriate name and extension (e.g., data.txt). After reading the numbers, your program should convert them to amounts in Euros (assume £1 = €1.11 which it probably isn't but was when I prepared this exercise) and output them directly to another text file (e.g., call it out.txt). Your program should also output the sum of the two Euro amounts to the file. For example, if the input file contained

```
12.45
14.23
```

Turn over

The output file generated by the program is

```
13.82
15.80
29.62
```

Report all currency values to two decimal places. Make appropriate use of constants in your program.

Exercise D

Write a program that uses a **JOptionPane** to read two **String**s from the user, first the user's given name and then their family name. Then display their full name in the terminal window by concatenating (joining) the two **String**s together.

This program could be improved by writing the result to a graphical dialog rather than the terminal window. You can do this using the **showMessageDialog** method of the **JOptionPane** class. One version appears in the notes but there is also a version which has the following syntax:

```
JOptionPane.showMessageDialog(null,message,title,
   JOptionPane.INFORMATION_MESSAGE);
```

Here, **message** and **title** represent the **String** variables containing the message you want to display and the title at the top of the dialog window. Write a second version of the program you wrote above which uses this version of the **showMessageDialog** method.

Exercise E

Telephone numbers in the UK consist of a four or five digit area code, followed by a six or seven digit number. For example:

(0114) 239 3124
(01226) 783 215

Write a Java program that uses a **JOptionPane** to input a telephone number in the above form as a string, and then extracts the area code and number. For the second example shown above, the output of the program should be something like this:

```
The area code is: 01226
The telephone number is: 783 215
```

Your program should remove spaces between the area code and the telephone number (but spaces within the telephone number should be retained). Hence the following input:

```
 (01226)     783 215
```

will give the same result as above.

N.B. The **indexOf(..)** method of the **String** class works with a single **char** as its parameter as well as with another **String**.

Continued

Exercise F

Write a program which asks the user to type in a sentence, reads it in (this time using **`EasyReader`**), works out what the 4th character is and prints the sentence back out but this time replacing every occurrence of the 4th character, whatever it is, by an asterisk. By 4th I mean the 4th counting from conventionally from 1. So the program's input and output could look like this.

```
U…>\java ExerciseF
Please type in a sentence : I like working in The Diamond
I l*ke work*ng *n The D*amond
```

To do this you will need to use two **`String`** methods we didn't cover in the lecture. The method **`charAt(i)`** where **`i`** is an integer when applied to a **`String`** returns a **`char`** which is the ith character in the **`String`** counting from 0 and the method **`replace (oldChar, newChar)`** when applied to a **`String`** will replace every occurrence of the **`char oldChar`** by the **`char newChar`**.

Note that in the example above the capital letter I is not converted to an asterisk. That is because upper and lower case letters are different.

Exercise G

In the lecture we established that the decimal Unicode value for the upper case alphabetic characters starts at 65 for the letter A. The rest of the uppercase alphabet runs consecutively to Z at 90. Write a program to find where the lower case alphabet starts and also find out if the decimal code for the character 0 is nine less than that for the character 9.