300581 & 300903 Programming Techniques

Week 7 Practical Exercises

During the Week 7 practical complete the following exercises using the **Eclipse IDE** to produce working Java code.

Coding Standards: When writing code in this unit you are required to adhere to the **Coding Standards** and General Principles as defined in the FAQ in vUWS. Your code for the following exercises should adopt these standards and general principles.

1) Class name: **NumberFile.java**

   a) Write a method to create a text file named **numbers.txt** that is to contain the whole numbers 101 to 10000 (use a for loop to write the output to the file). Write a second method that will read the data from **numbers.txt** and display the content to the screen (one value per line of output). Write a third method that can be used to add (sum) all of the numbers from the file and display their total on the screen – **do not use an array**. Write a program that includes all of the above methods. Your program must allow the user to choose if they want to create the file (first method), read & display the file content (second method), read and calculate the total from the file (third method), or exit the program. The program should continue to run until the user chooses to exit the program.
   Hint: When writing code for the first method you will need to consider how you will want to read the data back from the file in in the second and third method.

   Make sure that your Java program checks that the file exists before attempting to read from it. If the file does not exist then handle this situation by informing the user that the file does not exist and then create the file by executing the first method.

   b) Modify your program (and possibly the methods) so that
      i) the user can choose the starting and ending values to be stored in the text file
      ii) the user can enter the name of the file that is created by the first method
      iii) the user can enter the name of the file that is to be read by the second method.

2) This is a continuation of Exercise 2 from week 6 practical exercises. Import a copy of the two java files, **Expenses.java** and **ExpensesDemo.java** that you developed in Exercise 2 Week 6 into your week 7 project in Eclipse.

Kevin Malone has realised that it would be easier for him to be able to enter the weekly sales and expenses figures for each month into a text file that your program could then process.

   a) Modify your program (ExpensesDemo.java) to read the data from the file **MalonesCones.txt** instead of from the keyboard (ie, your program will no longer need to ask the user for the month name, number of weeks in the month or the weekly sales and expenses values since these are now in the text file) instantiating an Expenses object for each month that is read from the file. **Note:** the code in the Expenses class should not need to be modified.

An example of the file **MalonesCones.txt** has been included in this week's practical zip file. MalonesCones.txt contains data in the following format (**suggestion**: open the file in a text editor to confirm that you understand the structure and organisation of the data in the file):

```
February 4                    month name<space>number of weeks in month<eoln>
200 300                       sales week 1<space>expenses week 1<eoln>
300 450                       sales week 2<space>expenses week 2<eoln>
350 250                       sales week 3<space>expenses week 3<eoln>
275 275                       sales week 4<space>expenses week 4<eoln>
<repeat pattern for subsequent months>
```

Use the example input file to test your code. Make sure that your Java program checks that the file exists before attempting to read from it. If the file does not exist then handle this situation by informing the user that the file does not exist and then obtain a new file name/location from the user. You should also ensure that your code works correctly for different length files (ie, the input files used for marking your program may contain more or less lines of text than the example file provided).

b) The program will need to generate the on-screen report for Kevin as per the following example output:

```
For Month of February:
Balance:    $-150.00
Total Sales:        $1125.00
Total Expenses:     $1275.00
Keleven:    $150.00

For Month of March:
Balance:    $25.00
Total Sales:        $1325.00
Total Expenses:     $1300.00
Keleven:    $0.00

For Month of April:
Balance:    $-256.80
Total Sales:        $1103.05
Total Expenses:     $1359.85
Keleven:    $256.80
```

c) The program will need to generate a text file (**expenseSummary.txt**) containing the following information for each month that was processed from the MalonesCones.txt file:

- Month Name
- Month Balance
- Total Sales
- Total Expenses
- Keleven

Each month's data is to be on a separate line with each data value separated by a comma. Hence, the expenseSummary.txt file should contain data in the following format:

```
February,-150.00,1125.00,1275.00,150.00
March,25.00,1325.00,1300.00,0.00
April,-256.80,1103.05,1359.85,256.80
```

This file format is commonly known as a CSV file (comma separated value).