Lab Practice Week 9

You need to show working versions of your answers to all questions to your tutor. Your tutor will expect to see them by your next session.

What to submit: your answers to exercises 2.

Note: even though you only need to submit those exercises mentioned above, you should attempt all exercises in each lab practice to help broaden your understanding; this may also help with your assignment work.

Do all the programs in NetBeans IDE.

NOTE: Include internal documentation in your code.

Before starting the exercises, make sure you have read the relevant lecture material.

1. Write a Java program to perform the following tasks:

   The program should ask the user for the name of an input file and the name of an output file. It should then open the input file as a text file (if the input file does not exist it should throw an exception) and read the contents line by line. It should also open the output file as a text file and write to it the lines in the input file, prefixed by line numbers starting at 1.

   So if an input file named "fred.txt" contains:

```
Hello,

I am Fred.
I am enrolled in ICT167.

Bye
```

   and the user enters "fred.txt" and "fredNum.txt", then after running the program "fredNum.txt" will contain something like:

```
1 Hello,
2
3 I am Fred.
4 I am enrolled in ICT167.
5
6 Bye
```

Finally, the program should display on the screen your name, the name of the output file, followed by a count of the total number of lines, the total number of words, and the total number of characters in the input file.
For the above input file, the following will be displayed to the screen:

```
My name = Joe Bloggs
Name of Output file = fredNum.txt
Total number of lines in fred.txt = 6
Total number of words in fred.txt = 10
Total number of characters in fred.txt = 43
```

2. Write a Java program to perform the following tasks:

The program should ask the user for the name of an input file (scores.txt). The input file should contain 10 records. Each record in the input file should consist of a name and score between 0 and 100 (inclusive). Eg: fred 95

The program should open the input file as a text file (if the input file does not exist it should throw an exception) and read the contents line by line.

Store each record in an array of a user-defined Score class. The Score class can be kept very simple with 2 instance variables, a default constructor, get and set methods for the instance variables.

Now process the array of Score class objects to determine:

- The average of all scores
- The largest score
- The smallest score

Now output the above stats and all records to the file *output.csv*. The first line written to this file should consist of the number of records, the average score, the largest score and the smallest score; these values should be comma separated. Finally, the contents of the array of Score objects should be written to *output.csv* starting from the second line (the first line being taken by the above stats). The format to output the array contents is the same as the input file, with the exception that the fields should be comma separated instead of

s
p
a
c
e

s
e
p
a
r
a
t
e