**Institute of Technology Tralee**

**Ord/Hons BSc. in Computing with Specialism (Group D) - Year 1**

**Continuous Assessment #2**

**Date: 14/3/13**

**Time: 1 – 3 p.m.**

**Object Oriented Programming 1**

**Instructions:** Attempt the following question. You should use the JCreator IDE. When you are finished coding, print out your code in **landscape** format.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Q1.**

A program is required that will read in an *arbitrary* amount of **positive** numeric values. End of input is signaled by the user entering the empty string for the number (hitting return on the keyboard). The numbers can contain a single decimal point (or they could be whole numbers). The program must first of all validate the value entered to ensure it is a valid positive number. If the number turns out to be invalid, the user is given a suitable warning message and given the opportunity to re-enter the value, and this process repeats until a valid number has been supplied.

Once a particular value has found to be valid, the program should keep track of the amount of whole numbers entered as well as the amount of values that contain a decimal point. In order to determine these quickly you may find the following Java API method especially useful:

public int **indexOf**(int ch)

This method returns the index (position) within the string of the first occurrence of the specified character. If a character with value ch occurs in the String object, then the index of the first such occurrence is returned, otherwise the method will return the value -1 to indicate the particular character ch did not occur in the string. The method is non-static and so must be called on a string object reference (variable) e.g.

String test = “Hamster”;

int outcome;

outcome = test.indexOf(‘m’);

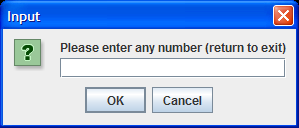
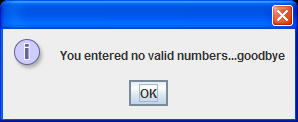
System.out.println(“The character m was first found at position ” + outcome);

It should also keep track of the largest number entered as well as determining the average of the numbers entered to **3 decimal places**.

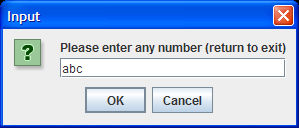
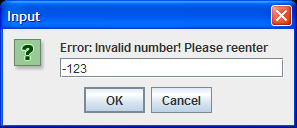
For full marks your program should, along with a logically correct solution for the problem above, include **comments** and **meaningful variable names** and you must ensure your program is **terminated correctly**.

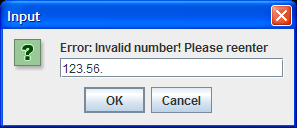
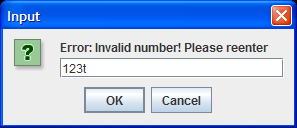
**Sample Screenshots**

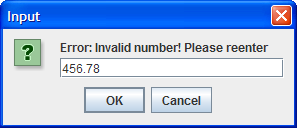
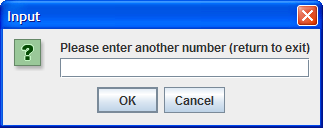
**Run 1 – the user hits return immediately when the program starts**

**Run 2 – the user begins by entering a number of invalid values, all of these are rejected until the user finally enters a valid one (456.78 in this case)**

**…… having input a number of values, the program finally produces the following ouput (yours will differ depending on the values you supply)**

