**Institute of Technology Tralee**

**Ord/Hons BSc. in Computing with Specialism (Groups B and C) - Year 1**

**Continuous Assessment #2**

**Date: 15/3/13**

**Time: 9 – 11 a.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 Java program is required that will read in and process **exactly 3 time values** using a **for** loop. Each time value must have the form hh:mm in order to be considered a valid time value, where hh represents the hours part of the time and must be a value between 00 and 23 inclusive and mm represents the minutes part of the time and must be a value between 00 and 59 inclusive.

Each time value entered must be fully validated in order to ensure it corresponds to a valid time. First of all, it must contain **exactly 5 characters**. If it passes this test then the individual characters can be tested. It should have the colon character in the middle of it, separating the hours and minutes values (which have the restrictions mentioned earlier). Otherwise the user will be repeatedly prompted for a valid replacement value until such is provided. You can use a **while** loop for this validation process if you wish. Depending on how you go about things, you may find the **charAt**() method particularly useful for your validation routine.

The main (for) loop should also generate a message each of the 3 times it iterates, after a valid time has been entered. This message should be “Good morning!” if the time happens to be between midnight (inclusive) and before midday (12:00). It should be “Good afternoon” if the time happens to be between midday (inclusive) and before 18:00. Finally, the message should be “Good evening” if the time happens to be between 18:00 (inclusive) and before midnight (00:00). You may find the (non-static) **substring**() method of the Java API String class particularly useful when trying to extract the hours part of the time (there are other ways to do it also however). This method has the following definition:

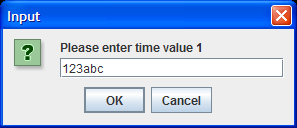
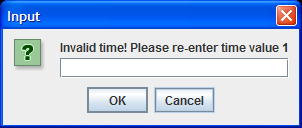
|  |  |
| --- | --- |
| [String](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) | [**substring**](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html#substring%28int,%20int%29)(int beginIndex, int endIndex)            Returns a new string that is a substring of this string e.g. to return the substring that contains the 3rd and 4th characters of the original string testString you would code testString.substring(2,4) |

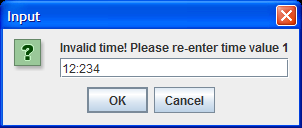
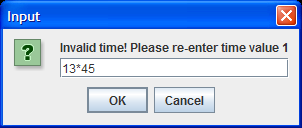
Using the test values as indicated in the screen shots below, the program should give you **exactly** the following output when it runs, including banners, blank lines etc.

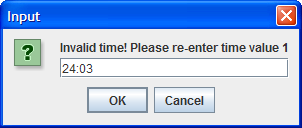
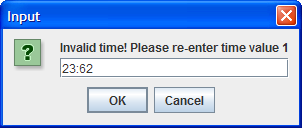
For full marks here your program should, along with a logically correct solution for the problem above, include **comments** and **meaningful variable names**. It should also be **terminated correctly**.

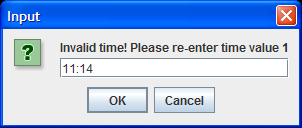
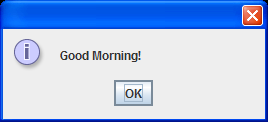
**Sample Screenshots**

**Run 1 – The user begins by entering some invalid times. All of these are rejected until the user enters a valid one (11:14 in this case). When the user enters a valid time, they are presented with an appropriate message (“Good morning!” in this case).**

**….as soon as the user hits return on the message dialog, the next iteration of the for loop occurs and the user is prompted for the next time. This process repeats until 3 valid times are entered, at which time the program terminates.**

