University of Windsor

SCHOOL OF COMPUTER SCIENCE
03-60-212 – OOP USING JAVA
SUMMER 2014

LAB – 1

[To get the **full marks**, complete and show your works to the Lab Instructor before the end of the Lab period. Lab works submitted in the next week's Lab will get a maximum of **75% marks**, if you attend this Lab, and a maximum of **50% marks**, if you do not attend this Lab.]

Objective:

The objective of this Lab is to be familier with the ***Eclipse*** IDE, using which you are going to develop most, if not all, of the Lab works and the home Assignments. You will also work on developing some simple Java programs in this Lab.

In all the machines you will be using in the Labs, as well as most of the Unix machines in the university, ***Eclipse*** is already installed. For your home computers or Laptops, you can download and install ***Eclipse*** from the website: ***www.eclipse.org/downloads/***

You can find tutorials on how to use ***Eclipse*** at various websites including the followings: ***http://agile.csc.ncsu.edu/SEMaterials/tutorials/eclipse/eclipse_tutorial_3.5.html*** or at ***www.cs.umanitoba.ca/~eclipse/***. We suggest that you should review some of those tutorials before starting the Lab – 1. You can also watch an introductory video on how to use ***Eclipse*** at this YouTube site: ***www.youtube.com/watch?v=DRBrv3EVTfQ***

Once your sufficently familier with the ***Eclipse*** environment, work on the follwings problems:

Although Java is an ***Object Oriented Programming*** language, it is possible to develop ***C*** like simple ***structural programs*** using ***Java***. The simplest way to experiment that is to write your whole program within a single class. The name of the class should be the same as the name of the program itself. Write the words ***static*** (or ***public static***, or ***private static***, depending upon context. For simple cases all will work.) in front of the name of **all the methods** you use, including the ***main().*** The reason for using these words will be explained to you later, when we talk about classes. You do not need any function prototypes, and Java has no concept of global variables. Note the skeleton of an structural program ***MyProg.java*** as given below:

```
public class MyProg {
   public static void main( String[] args ) {
      //statements
      int x = 5;
      method_1();
      int a = method_2( x );
      method_3();
      //statements
   }//end of main
```

```
   static void method_1(){
      //do something
   }//end of method_1

   private static int method_2( int num ){
      //do something
      //return an int value
   }//end of method_2

   public static void method_3(){
      //do something
   }//end of method_3
}//end of class MyProg
```

Now, using the techniques discussed above, write two Java programs to achieve the following goals. For these two trivial problems, it is not mandetory to use any additional methods. If you prefer, you can solve the problems using only the ***main()*** method.

Problem A:

Prime numbers are those numbers that can be divided by only two **distinct** numbers, 1 and the number itself. For example, the numbers 2 (the only even number), 3, 5, 7, 11, 13, 17, 19... are all prime numbers.

Write a java program (call it ***Lab1a.java***) that continuously asks the user to enter a positive integer greater than 1 and then finds out and declares whether the number is a prime number or not. The program terminates as soon as the user enters 1 or any number less than 1.

Use an object of the **Scanner** class to read the integers from the keyboard as entered by the user. You have to import into your program the class ***java.util.Scanner***, for that.

Sample Output from Lab1a.java:

```
Enter a number (1 to quit):
357
357 is not a prime number
Enter a number (1 to quit):
271
271 is a prime number
Enter a number (1 to quit):
1
Bye...
```

Problem B:

Java ***String*** class has many useful methods to manipulate strings and characters.

Write a java program (call it ***Lab1b.java***) that continuously asks the user to enter a string (sentence) and does the followings:

1. Counts and prints the number of vowels (independent of upper or lower cases) in the sentence.
2. Using any appropriate methods from the ***StringTokenizer*** class, the program separates and prints individual words in the sentence each in a separate line. You may want to import the class ***java.util.StringTokenizer***.
3. The program should also find and print the shortest and the longest words in the sentence.
4. To get rid of any leading or traling spaces in the input string, you can use the method ***trim()*** of the ***String*** class.

As before, use an object of the **Scanner** class to read the sentence from the keyboard as entered by the user. The program should terminate as soon as the user enters the word ***Quit*** (or ***quit***).

Sample Output from Lab1b.java:

```
Enter a sentence (“Quit” to quit):
This is a test
The number of vowels = 4
The words are:
This
is
a
test
The shortest word: a
The longest word: This
Enter another sentence (“Quit” to quit):
Programming in Java is easy
The number of vowels = 9
The words are:
Programming
in
Java
is
easy
The shortest word: in
The longest word: Programming
Enter another sentence (“Quit” to quit):
quit
Bye...
```