TOBB University of Economics and Technology
Department of Computer Engineering

BİL141 Computer Programming (Java)
Spring 2016

Homework 4

Due by Feb 17 (Wednesday) 24:00

Subject: Class, method, constructor, overloading, arrays, loops, strings

1) (50p) Write a **Temperature** class that represents temperatures in degrees in both Celsius and Fahrenheit. Use a floating-point number for the temperature and a character for the scale: either 'C' for Celsius or 'F' for Fahrenheit. The class should have:

• **Four constructors**: one for the number of degrees, one for the scale, one for both the degrees and the scale, and a default constructor. For each of these constructors, assume zero degrees if no value is specified and Celsius if no scale is given.

• **Two accessor methods**: one to return the temperature in degrees Celsius, the other to return it in degrees Fahrenheit. Use the formulas, C=(F-32)x(5/9) and F=Cx(9/5)+32, and round to the nearest tenth of a degree.

• **Three set methods**: one to set the number of degrees, one to set the scale, and one to set both.

• **Three comparison methods**: one to test whether two temperatures are equal, one to test whether one temperature is greater than another, and one to test whether one temperature is less than another.

The driver program (**TemperatureTest**.java) is given in the course piazza site. Make sure that it works with the class you wrote, and all method calls and constructors work without error.

Notes:

Getting the equals method to work properly highlights the problem of comparing two floating point values. One consequence of using a fixed number of bits to encode floating point values is that they cannot always be encoded precisely. Two mathematical expressions that have the same result when done by hand may actually have slightly different values (in the last decimal place or so) when stored in memory. For example the calculation

```
double a = 51.8 /10;
```

is likely to give a slightly different value than

```
double a = 0.518 * 10;
```

because each number is stored as an approximate value. Because of this comparisons of floating point values in conditional expressions do not always return the expected results. A way to get around it is to decide how many decimal place accuracy we want to compare, multiply the floating point numbers by the appropriate power of 10, then round the results to get integers. This is the approach taken in the comparison methods, `equals`, `isGreaterThan`, and `isLessThan`: First the methods make sure both temperatures are in the same units (degrees C, but degrees F would be equally valid) using the `getC()` method. Since `getC()` returns a value with one decimal place, the temperatures are then multiplied by 10 and rounded using `Math.round()`, which returns an integer value (you can think of it as comparing an integer number of tenths of degrees)

2) (50p) Write Primes.java program that reads an integer *n* (>0) and prints out the first *n* prime numbers on the screen.

3) **Bonus** (30p) Write Sort.java program that reads a line of words, sorts them alphabetically and prints line by line. Here is a sample execution:

```
Enter words: one Two  three      fiVE   SiX
Five
One
Six
Three
Two

Enter words: zeynep Ali VELİ ZEKİ
Ali
Veli
Zeki
Zeynep

Enter words:
Bye
```

İpucu: Sıralama için kelimeleri bir diziye ekleyebilir, sonra "kabarcık sıralama" algoritması ile sıralayabilirsiniz.

In each program have the following comments at the top.

```
/*
 * BİL141 Spring 2016
 * HW#
 *
 * NameOfTheProgram (ex: GradeCalculator)
 * Explanation (ex: Given a number of grades (0-100),
 * Counts the number of letter grades)
 *
 * @author      Firstname Lastname (Student number)
 * @email  ....@etu.edu.tr
 * @date        yyyy/mm/dd
*/
```