CSCI 1101 – Winter 2013
Laboratory No. 2

This lab is a continuation of the concepts of object-oriented programming. It focuses on method overloading and passing objects into methods.
Note:

1. *All submissions must be made on http://moodle.cs.dal.ca with your CS login id. Submission deadline is 12 noon on Saturday, January 26th.*
2. *Run each demo program for at least three output sets.*
3. *Put the java source code files and the text outputs for each exercise in separate folders. Zip all the folders into one file and submit the zip file.*

Exercise 1: Write a class called MotorBoat with the following attributes:

- The amount of fuel in the tank (in liters)
- The speed of the boat (in km per hour)
- The fuel consumption rate of the boat (in liters per km).

and the following methods:

- Constructor that sets the fuel in the tank and rate. The constructor should set the speed to zero.
- Get and set methods.
- A toString method that returns the attribute values.
- A method to change the speed of the boat by a given value (the method should accept the change value which is a positive or negative number and increase or decrease the speed, respectively).
- Operate the boat for a given amount of time in minutes at the current speed (this method should reduce the amount of fuel in the tank by the appropriate amount). This method should also print the distance travelled by the boat.
  Note the formulas: If the boat has consumption rate r, current speed is s and the time in minutes is t, then the amount of fuel used is = r * s * t/60
  Distance travelled in km = s*t/60
- Refuel the boat with some amount of fuel.

Implement the class and test it using a demo program. Here's the skeletal outline of the demo program:
Warning: Do not cut and paste the code. It may result in errors. Write it yourself.

```
import java.util.Scanner;
public class MotorBoatDemo
{
      public static void main(String[] args)
      {
            MotorBoat myBoat;
            Scanner keyboard = new Scanner(System.in);
            System.out.print("Enter the fuel in tank: ");
            double f = keyboard.nextDouble();

            System.out.print("Enter the consumption rate: ");
            double r = keyboard.nextDouble();
            myBoat = new MotorBoat(f,r);

            //complete the rest of the code

      }
}
```

Here's a sample screen dialog and output:

```
Enter the fuel in tank: 10
Enter the consumption rate: 0.1


Fuel in tank: 10.0 liters
Current speed: 0.0 km per hour
Rate: 0.1 liters per km

Change speed by(enter a value, positive or negative): +6
Current speed: 6.0
Operate for (enter time in minutes): 6

Distance travelled: 0.6


Fuel in tank: 9.94 liters
Current speed: 6.0 km per hour
Rate: 0.1 liters per km


Process completed.
```

Exercise 2: Implement a Stock class that has the following attributes:
symbol (String), price (double), number of shares (int)
and the following methods:
Constructor (that sets the symbol, price and number of shares to user defined values).
Get and set methods.
toString method returns the symbol, price and number of shares in a nice format
method `compare(Stock s)` compares the share price of this stock with another stock and returns:
-1 if the value of this stock is lower than the other stock.
+1 if the value of this stock is higher than the other stock.
0 if both the values are the same.
(Value of the stock = price * number of shares)

For example, let's say IBM has a price of $105.23 and you have bought 45 shares and MOS has a price of $89.88 and you have bought 60 shares. Then the value of IBM in your portfolio is 105.23*45= $4735.35 and the value of MOS in your portfolio is 89.88*60 = $5392.80. Comparing this stock(IBM) with another stock(MOS) will return a -1 since the value of MOS is greater in your portfolio.

Test the Stock class with a client program that asks the user to enter the symbols, share prices, and number of shares for two stocks, prints their values, determines which stock is higher than the other and by how much and prints the total value. You must use the methods in the Stock class wherever appropriate.

A portion of the client program is given below for your programming convenience.

```
import java.util.Scanner;
```

```
public class StockDemo
{
      public static void main(String[] args)
      {
            Scanner keyboard = new Scanner(System.in);
            String sym1, sym2;
            double prc1, prc2;
            int sh1, sh2;
            //get the values for two stocks
            System.out.print("Enter the symbols for the two stocks: ");
            sym1 = keyboard.next();
            sym2 = keyboard.next();
            System.out.print("Enter their prices: ");
            prc1 = keyboard.nextDouble();
            prc2 = keyboard.nextDouble();
            System.out.print("Enter the number of shares for the two
stocks: ");
            sh1 = keyboard.nextInt();
            sh2 = keyboard.nextInt();

            //create the first Stock
            Stock s1 = new Stock(sym1,prc1,sh1);


            //create the second Stock
            Stock s2 = new Stock(sym2,prc2,sh2);

            // continue the rest of the code here
```

The following is a sample screen dialog:

```
Enter the symbols for the two stocks: IBM MOS
Enter their prices: 105.23 89.88
Enter the number of shares for the two stocks: 45 60
I have the following stocks:
Stock: IBM
Price: 105.23
Shares: 45
Stock: MOS
Price: 89.88
Shares: 60
The value of MOS is higher than IBM by $ 657.45
The total value of my portfolio is $ 10,128.15

Process completed.
```

Exercise 3: Implement the Point class (basic Point class was discussed in the lecture). The class has the following instance variables:

- x coordinate
- y coordinate

and the following methods:

- Constructor that sets the x and y coordinates
- Get and set methods
- Method equals if this point is equal to another point object (if the x and y coordinates are the same).
- Method isHigher if this point is higher than another point object (if the y coordinate of this point is greater than the y coordinate of the other point)
- Method findLength that calculates the length of the line connecting this point to another point object. (Formula for length between two points (x1,y1) and (x2,y2) is square root of $((x2-x1)^2 + (y2-y1)^2)$

Test the class. In the demo program, construct four points p1, p2, p3 and p4 using user-defined values. Determine

- Which point is the highest. (If two points are at the same height, you may report either point).
- Whether line p1→ p2 is longer than line p3→ p4.

A sample screen dialog is given below:

```
Enter the x and y coordinates for point1: -9 7
Enter the x and y coordinates for point2: 5 -7
Enter the x and y coordinates for point3: 4 -3
Enter the x and y coordinates for point4: 0 -2
[-9,7] is the highest point
The length between [-9,7] and [5,-7] is 19.79898987322333
The length between [4,-3] and [0,-2] is 4.123105625617661
Line from [-9,7] to [5,-7] is longer
```