CSCI 1101- Computer Science II
Laboratory No. 1
Week of January 16-20, 2017
Deadline for submission: 11.55 p.m. (five minutes to midnight) on Saturday, January 21, 2017

Welcome to your first lab in CSCI 1101. This lab introduces you to the basics of object-oriented programming that was discussed in the lectures. For each exercise, your task is to first write the class that defines the given object. Compile it and check it for syntax errors. Then write the "demo class" with the main program that creates and uses the object.

You may use Eclipse, NetBeans, JGrasp, JCreator or any other IDE.

***Submission**: All submissions are through Brightspace. Log on dal.ca/brightspace using your Dal NetId. Submissions are pretty straightforward. Instructions will be also be given in the first lab.*
Deadline for submission: Saturday, January 21, 2017 at 11.55 p.m. (five minutes before midnight).

***Marking Scheme**:*

Each exercise carries 10 points. Your final score will be scaled down to a value out of 10.
Working code, Outputs included, Efficient, Good basic comments included: 10/10
No comments: subtract one point
Unnecessarily long code and inefficient program, improper use of variables: subtract one point
No outputs: subtract two points
Code not working: subtract up to six points depending upon the extent to which the program is incorrect.

***Error checking**: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.*

***Testing your code and generating the outputs:** If the test data is included in the question, use that to test your classes. In addition, test it with two more input sets. Otherwise, create your own test data and run your program for at least 3 input sets such that they cover the range of results expected.*

The first two exercises will help you to get started with the basic concepts of object-oriented programming. They are a review the Rectangle and the Circle classes that were discussed in the lectures.

Exercise 1 (Review from lecture notes) [0 points]
Write a java class definition for a rectangle object. The object should be capable of setting its width and height and computing its area and perimeter. Use this to create two rectangle objects of dimensions 10 X 30 and 20 X 25, respectively. Display their areas and perimeters.

Here is the Java class file (Rectangle.java). Save it and compile it.

```
public class Rectangle
{
   private int width;
   private int height;

   public void setWidth(int w)
   {
      width = w;
   }
   public void setHeight(int h)
   {
      height = h;
```

```
   }
   public int getWidth()
   {
      return width;
   }
   public int getHeight()
   {
      return height;
   }
   public int findArea()
   {
      return height*width;
   }
   public int findPerimeter()
   {
      return 2*width +2*height;
   }
}
```

Now use the following tester program (RectangleDemo.java) to create and use Rectangle objects. Compile and run it.

```
public class RectangleDemo
{
   public static void main(String[] args)
   {
      Rectangle rect1, rect2;
      rect1 = new Rectangle();
      rect2 = new Rectangle();
      rect1.setWidth(10);
        rect1.setHeight(30);
      rect2.setWidth(20);
      rect2.setHeight(25);
      System.out.println("Area (rectangle1): " + rect1.findArea() + " square units");
      System.out.println("Perimeter (rectangle1): " + rect1.findPerimeter() + " units");
      System.out.println("Area (rectangle2): " + rect2.findArea() + " square units");
      System.out.println("Perimeter (rectangle2): " + rect2.findPerimeter() + " units");
   }
}
```

Warning! If you cut and paste this code, it may result in errors (for example, the double quotes may not compile properly. Please enter the code on your own).

Exercise 2 (Review from lecture notes) [0 points]

Write a class called Circle that defines a Circle object. It has one attribute, namely, radius (of type double). Provide a constructor that sets the radius to the supplied value, get and set methods for the radius, a method that computes and returns the area, and a method that computes the circumference.

Test it in a demo program that accepts a radius input by the user, creates the Circle, and displays its area and circumference.

```
//Circle.java
public class Circle
{
   //instance variables
   private double radius;

   //methods
   public Circle(double r)
   {
      radius = r;
   }
   public void setRadius(double r)
   {
      radius = r;
   }
   public double getRadius()
   {
```

```
      return radius;
   }
   public double findArea()
   {
      return Math.PI*radius*radius;
   }
   public double findCirc()
   {
      return 2*Math.PI*radius;
   }
}

//CircleDemo.java
import java.util.Scanner;
public class CircleDemo
{
   public static void main(String[] args)
   {
      Circle circ1;
      double r;
      Scanner input = new Scanner(System.in);
      System.out.print("Enter the radius: ");
      r = input.nextDouble();
      circ1 = new Circle(r);
      System.out.println("Area of the circle is: " + circ1.findArea() + " square units");
      System.out.println("Circumference of the circle is: " + circ1.findCirc() + " units");
  }
}
```

No submission required for the above exercise. Just test it and make sure it runs properly.

Exercise 3 (also review from lecture notes) [10 points]
Make the following changes to Rectangle.java and RectangleDemo.java:

- **In Rectangle.java, change the constructor so that it accepts the width and the height as input parameters and sets those attributes.**
- **In Rectangle.java, add a toString method to display the length and width of the rectangle like this:**
  `Rectangle: [Width:10, Height:30]`
- **Modify the RectangleDemo.java program so that it reads the width and height from the keyboard input by the user, creates the Rectangle object and displays its dimensions (using the toString method), the area and the perimeter. Here's a sample screen dialog:**

```
Enter width and height of the rectangle: 10 13

Rectangle: [Width: 10, Height:13]
Area:      130 square units
Perimeter: 46 units
```

Test your RectangleDemo.java for at least three different input sets.
Keep the following files ready for submission: Rectangle.java and RectangleDemo.java

Exercise 4: [10 points]
Body Mass Index (BMI) is a measure of health based on height and weight. Given a person's weight in pounds and height in inches, the BMI can be calculated using the formula:
BMI = 703 X Weight in pounds/ ((Height in inches)X(Height in inches))

The interpretation of BMI for people 20 years or older is as follows:

BMI	Interpretation
BMI< 18.5	Underweight
18.5<=BMI<25.0	Normal
25.0<=BMI<30.0	Overweight
BMI>=30.0	Obese

Your task is to write an object-oriented program to determine the BMI of a person. Write a class called **Person** that has the following attributes:

- name (a String)
- age (an int)
- weight in pounds (a double)
- height in inches (a double)

and the following methods:

- A constructor that creates a Person object with the user specified name, age, weight and height.
- Get and set methods for name, age, weight and height.
- calcBMI() that calculates the body mass index and returns that value.

Write a demo class BMIDemo.java. It should prompt the user to enter the name, age, weight in pounds and height in inches. With these values, it should create a Person object. If the age is under 20, it should simply display "The minimum age should be 20 in order to calculate the BMI". If the age is >=20, then it should call the method calcBMI() to find the bmi and display the BMI. It should also display the BMI status based on the chart given above.

A sample screen dialog is given below:

```
Enter the name: John Doe
Enter the age: 45
Enter the weight in pounds: 140.5
Enter the height in inches: 67

The BMI of John Doe is 22.003
The BMI status is Normal
```

Another sample screen dialog is given below:

```
Enter the name: Ichabod Crane
Enter the age: 25
Enter the weight in pounds: 102.5
Enter the height in inches: 70

The BMI of Ichabod Crane is 14.7056122
The BMI status is Underweight
```

Another screen dialog (in this example, the program will not calculate the BMI since the age is less than 20):

```
Enter the name: Jane Doe
Enter the age: 18
Enter the weight in pounds: 125
```

```
Enter the height in inches: 65

The minimum age should be 20 in order to calculate the BMI.
```

The solution templates for Person.java and BMIDemo.java are given below. Fill in the missing code pieces in the places marked TO DO.

```
//Person.java
public class Person
{
   //TO DO: complete this class

}
//end of Person.java


//BMIDemo.java
import java.util.Scanner;
public class BMIDemo
{
   public static void main(String[] args)
   {
      Scanner keyboard = new Scanner(System.in);
      System.out.print("Enter the name: ");
      String n = keyboard.nextLine();
      System.out.print("Enter the age: ");
      int a = keyboard.nextInt();
      System.out.print("Enter the weight (in pounds): ");
      double w = keyboard.nextDouble();
      System.out.print("Enter the height (in inches): ");
      double h = keyboard.nextDouble();

     //TO DO: rest of the code here

    }

}
//end of BMIDemo.java
```

Test BMIDemo.java for at least three different inputs.
Keep the files Person.java and BMIDemo.java ready for submission.

Exercise 5: [10 points]

Write a class called Car that models the amount of gas required by the car to drive a certain distance. The attributes of the Car object are:

- tank capacity (in liters) (a double)
- amount of gas in tank (in liters) (a double)
- fuel consumption rate (in liters/km) (a double)

It has the following methods:

- a constructor
  public Car(double c, double f)
  that sets the tank capacity to c, fuel consumption to f. It should also set the amount of gas in tank to 0.
- Get and set methods for all the attributes.

- A method `public void fill(double g)`
  This method should add g to the amount of gas in the tank if the total doesn't exceed the capacity. If it exceeds the capacity, it should display a message "Cannot fill. Exceeds capacity".
- A method `public void drive(double d)`
  This method should calculate the amount of gas required to drive the distance d. If the car has sufficient gas, it should decrement the required amount of gas from the amount of gas in the tank. If not, it should display a message "Cannot drive. Not enough gas".

Write a demo program CarTester.java that creates a car with input values for tank capacity and fuel consumption. The program then prompts the user to enter the amount of gas to fill and the distance to drive. It should display the gas remaining in the tank after the distance is covered (or it should display the error message). Note that the amount of gas required to drive a distance d is given by
d X fuelConsumptionRate.

A sample screen dialog is given below:

```
----jGRASP exec: java -ea CarTester
Enter the capacity (in liters): 50.0
Enter the fuel consumption rate (in lt/km): 0.2
Enter the amount of gas to fill: 30.0
Enter the distance to drive: 100.0

Gas remaining in tank: 10.0
----jGRASP: operation complete.
```

Another sample screen dialog:

```
----jGRASP exec: java -ea CarTester
Enter the capacity (in liters): 50.0
Enter the fuel consumption rate (in lt/km): 0.3
Enter the amount of gas to fill: 45.0
Enter the distance to drive: 300.0
Cannot drive. Not enough gas

Gas remaining in tank: 45.0
----jGRASP: operation complete.
```

A third example:

```
----jGRASP exec: java -ea CarTester
Enter the capacity (in liters): 20.0
Enter the fuel consumption rate (in lt/km): 0.1
Enter the amount of gas to fill: 30.0
Enter the distance to drive: 100.0
Cannot fill. Exceeds capacity
Cannot drive. Not enough gas

Gas remaining in tank: 0.0
----jGRASP: operation complete.
```

Run CarTester.java for at least three different inputs.
Keep the files Car.java and CarTester.java ready for submission.

What to submit:
Create ONE zip file containing the following:

1. Rectangle.java

2. RectangleDemo.java
3. Person.java
4. BMIDemo.java
5. Car.java
6. CarTester.java
7. One text file that has the outputs (cut and pasted) for all the programs that you have tested.

Submit the zip file on Brightspace.