Assignment (BCAC391)

(Basic class, constructors, package concept)

1. Design a class name ShowRoom with the following description :

   Instance variable/ Data members:

   String name – to store the name of the customer
   long mobno – to store the mobile number of the customer
   double cost – to store the cost of items purchased
   double dis – to store the discount amount
   double amount – to store amount to be paid after discount

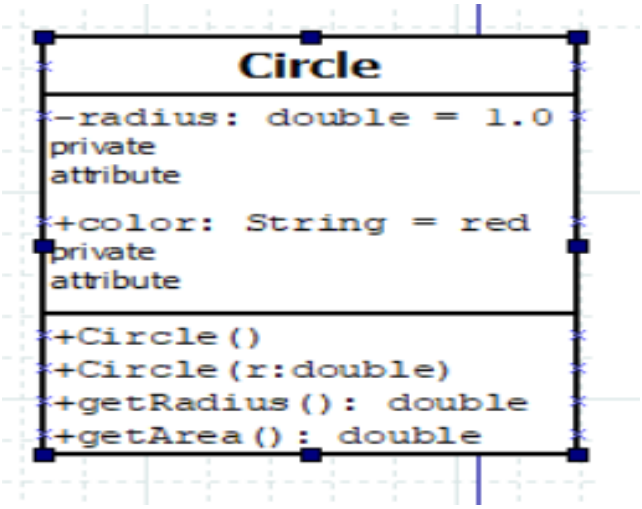
   Member method :
   ShowRoom() – default constructor to initialize data members
   void input() – to input customer name, mobile number, cost
   void calculate() – to calculate discount on the cost of purchased items based on the following criteria

Cost	Discount (in percentage)
Less than or equal to ₹ 10000	5%
More than ₹ 10000 and less than or equal to ₹ 20000	10%
More than ₹ 20000 and less than or equal to ₹ 35000	15%
More than ₹ 35000	20%

   void display() – to display customer name, mobile number, amount to be paid after discount.

Write a main method to create an object of the class the above member's methods.

2. A class called **`circle`** is designed as shown in the following class diagram.



This `Circle` class does not have a `main()` method. Hence, it cannot be run directly. This `Circle` class is a "building block" and is meant to be used in another program.
Let us write a *test program* called `TestCircle` (in another source file called `TestCircle.java`) which uses the `Circle` class, as follows:

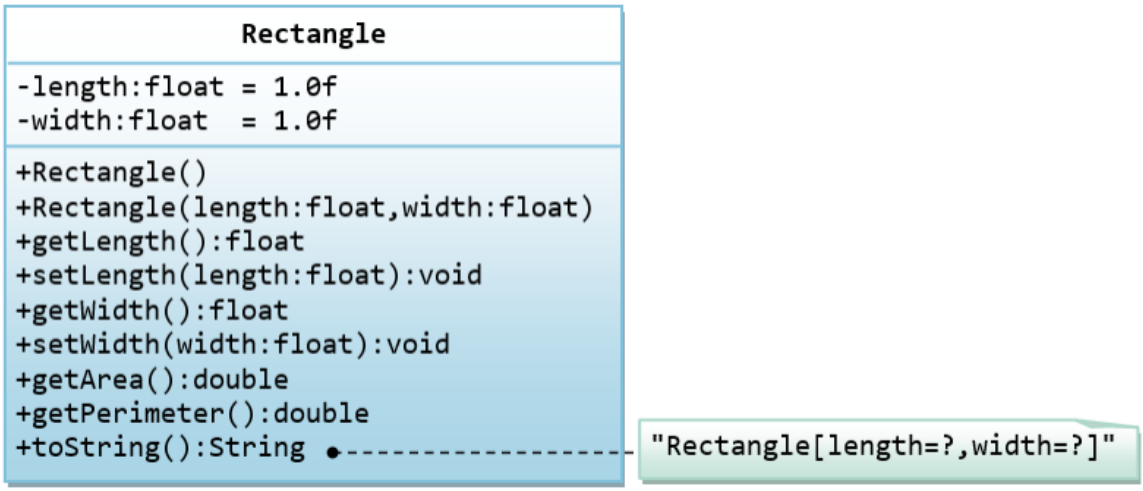
```
/**
 *  A Test Driver for the Circle class
 */
public class TestCircle {  // Save as "TestCircle.java"
   public static void main(String[] args) {
      Circle c1 = new Circle();
      System.out.println("The circle has radius of "
         + c1.getRadius() + " and area of " + c1.getArea());
      Circle c2 = new Circle(2.0);
      System.out.println("The circle has radius of "
         + c2.getRadius() + " and area of " + c2.getArea());
      //The circle has radius of 2.0 and area of 12.566370614359172
   }
}
```

Now, run the `TestCircle`

Sample Output:

```
The circle has radius of 1.0 and area of 3.141592653589793
The circle has radius of 2.0 and area of 12.566370614359172
```

3. A class called `Rectangle`, which models a rectangle with a length and a width (in `float`), is designed as shown in the following class diagram. Write the `Rectangle` class.



Below is a test driver to test the `Rectangle` class:

```
public class TestMain {
   public static void main(String[] args) {
      // Test constructors and toString()
      // You need to append a 'f' or 'F' to a float literal
      Rectangle r1 = new Rectangle(1.2f, 3.4f);
      System.out.println(r1);  // toString()
      Rectangle r2 = new Rectangle();  // default constructor
      System.out.println(r2);

      // Test setters and getters
      r1.setLength(5.6f);
      r1.setWidth(7.8f);
      System.out.println(r1);  // toString()
      System.out.println("length is: " + r1.getLength());
      System.out.println("width is: " + r1.getWidth());

      // Test getArea() and getPerimeter()
      System.out.printf("area is: %.2f%n", r1.getArea());
      System.out.printf("perimeter is: %.2f%n", r1.getPerimeter());
   }
}
```

The expected output is:

```
Rectangle[length=1.2,width=3.4]
Rectangle[length=1.0,width=1.0]
Rectangle[length=5.6,width=7.8]
length is: 5.6
width is: 7.8
area is: 43.68
```

```
perimeter is: 26.80
```

4.

Account
+balance : double
<<constructors>> +Account (initBalance : double)

Figure1.

1. Figure1 shows the UML class diagram of the Account class that you are going to create. It will have one public data member (or instance variable) called balance that maintains the monitory value of the customer's bank account. Initialize the balance instance variable with the parameter of the constructor. Create another class TestAccount which acts as a programmed to create an Account object with an initial balance of hundred. The test programmed will then add 47 and then subtract 150.Finally the test programmed must print the balance of the object to the standard output string.

   [The output should be similar to the following:

   Final account balance is -3.0]

2.

Account
-balance : double
<<constructors>> +Account (initBalance: double) <<methods>> +getBalance (): double +deposit (amt: double): void +withdraw(amt:double):void

Figure2.

Modify the Account class source file according to the Figure2 UML class diagram. The deposit method adds money to the account, the withdraw method removes money from the account and the getBalance method returns the current value of the current instance variable. The withdraw method should be implemented in such a way that the balance of the bank account should never go below zero.
Modify the TestAccount class by changing the amount in the call to the deposit method to 47 and the amount in the call to the withdraw method to 150.

[The output should be similar to the following:

Final account balance is 147.0]