TK1114 Computer Programming

Lab 9

Classes

Simple Problem Solving

29, 30 Nov & 1 Dec 2016

Name : _____

Matric Number : _____

9#	PC2 CIRCLE CLASS	
	Input	Standard input
	Output	Standard output
	Topic	Classes : **Sample Solution**

Problem Description

A class named `PC2Circle` is defined as follows.

- Private instance variable named `radius` that represents the radius of the circle.
- The constructor `PC2Circle()` that initialised radius to 0.0.
- `setRadius(double)` method to set a new radius value
- `getRadius()` method that returns the radius
- `getArea()` method that returns the area of the circle.
- `getCircumference()` method to returns the circumference of the circle.

The UML Class Diagram for class `PC2Circle` is as the following.

PC2Circle
-radius: double
+PC2Circle() +setRadius(radius:double): void +getRadius(): double +getArea(): double +getCircumference(): double

Write the code for `PC2Circle` class.

Write a program to test the `PC2Circle` class that reads the input and print the output as described below.

Input

The first line contains an integer `n` ($1 \leq n \leq 50$) which determines the number of test cases. Each of the following `n` lines contains a positive real number `r` ($0.0 \leq r \leq 500.00$) which represents the radius of the circle.

Output

For each test case, the output contains a line in the format "`Case #x: `", where x is the case number (starting from 1) follows by the radius, area and circumference of the circle. Format the output in 4 decimal places.

Sample Input Output

Sample Input	Sample Output
2 9.0 5.5	Case #1: 9.000 254.4690 56.5487 Case #2: 5.500 95.0332 34.5575

Solution for this problem.

```
// File name: TestPC2Circle.java

import java.util.Scanner;
import java.text.DecimalFormat;

public class TestPC2Circle {
   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);
      DecimalFormat  df  = new DecimalFormat("0.0000");
      double radius, area, circumference;

      int N = sc.nextInt();
      for (int i = 1; i <= N; i++) {
         PC2Circle myCircle = new PC2Circle(); // create PC2Circle object
         radius = sc.nextDouble();

         myCircle.setRadius(radius);   // sets radius
         area = myCircle.getArea();    // returns area
         circumference = myCircle.getCircumference(); // returns

         System.out.println("Case #" + i + ": " +
               df.format(myCircle.getRadius()) + " " +
               df.format(area) + " " + df.format(circumference));
      }
   }
}

class PC2Circle {                 // NOT a public class
   private double radius;         // private instance variable

   public PC2Circle() {           // constructor without parameter
      radius = 0.0;
   }
   void setRadius(double rad) {
      radius = rad;
   }
   double getRadius() {
      return radius;
   }
   double getArea() {
      return radius * radius * Math.PI;
   }
   double getCircumference() {
      return 2 * Math.PI * radius;
   }
}
```

9B	MY EQ-TRIANGLE	
	Input	Standard input
	Output	Standard output
	Topic	Classes : Simple problem solving

Problem Description

A class named `EqTriangle` that represents an equilateral triangle is defined as follows.

- Private instance variable named `side` that represents the sides of the rectangle.
- The constructor `EqTriangle()` that initialized the `side` to 0.0.
- `setWidth(double)` method to set a new width
- `setHeight(double)` method to set a new height
- `getWidth()` method that returns the width
- `getHeight()` method that returns the height
- `getArea()` method that returns the area of the triangle.
- `getPerimeter()` method to returns the perimeter of the triangle.

The UML Class Diagram for class `EqTriangle` is as the following.

EqTriangle
-side: double
+EqTriangle() +setSide(side:double): void +getSide(): double +getArea(): double +getPerimeter(): double

Write the code for `EqTriangle` class.

Write a program to test the `EqTriangle` class that reads the input and print the output as described below.

Input

The first line contains an integer `n` ($1 \leq n \leq 50$) which determines the number of test cases. Each of the following `n` lines contains a positive real number `side` ($0.0 \leq side \leq 500.00$) which represents the side of the equilateral triangle.

Output

For each test case, the output contains a line in the format "`Case #x: `", where `x` is the case number (starting from 1) follows by the side, area and perimeter of the triangle. (format the output in 4 decimal places).

Sample Input Output

Sample Input	Sample Output
2 9.0 5.5	Case #1: 9.0000 35.0740 27.0000 Case #2: 5.5000 13.0986 16.5000

Use the following program structure:

```
// File name: TestEqTriangle.java

// import statements

public class TestEqTriangle {
   public static void main(String [] args) {

      // code for test class
    }
}

class EqTriangle {    // NOT a public class

   // code for EqTriangle class
}
```

9E	MY PRINTCARD	
	Input	Standard input
	Output	Standard output
	Topic	Classes : Problem solving

Problem Description

A class called `MyPrintCard` which represent a prepaid card that can be used at all Print & Copy Center in campus are described below.

- Private instance variables named `balance` that represents the amount of money in the card.
- Constructor method `MyPrintCard()` that initialised the `balance` to **RM10.00**.
- There is no setter method for the class
- Getter method `getBalance()` that returns the current balance.
- `topupCard(double amt)` : top up the card, add `amt` to `balance` .
- `payService(double amt): boolean` : pay the print and copy services and update the `balance`. Payments are only valid if the balance after payment is not less than **RM5.00**, otherwise it returns `false`.
- `toString()`: returns the balance amount in Malaysian currency format. For example `"RM55.55"` .

Write the code for `MyPrintCard` class.

Write a test program named `TestMyPrintCard` that reads the input and print the output as described below.

Input

The input consists of a few test cases. For each test case, the first line of input is a positive integer N ($N \leq 50$) which indicates the number of transaction for the card. Each of the following N lines represents a transaction. It starts with character "+", "-" or "=" which indicates topup the card, pay the service or print the balance. Input is terminated by a test case where N is 0.

Output

For each test case, output a line in the format `"Case #x:"` where `x` is the case number (starting from 1), follow by the balance for each valid transaction, otherwise print `"invalid"`.

Sample Input Output

Sample Input	Sample Output
5 = - 0.70 - 2.40 - 1.20 + 20.00 3 - 5.00 - 0.50 = 0	Case #1: RM10.00 RM9.30 RM6.90 RM5.70 RM25.70 Case #2: RM5.00 invalid RM5.00