TK1114 Computer Programming

Lab 9

Classes

Simple Problem Solving

29, 30 Nov & 1 Dec 2016

Name : _____

Matric Number : _____

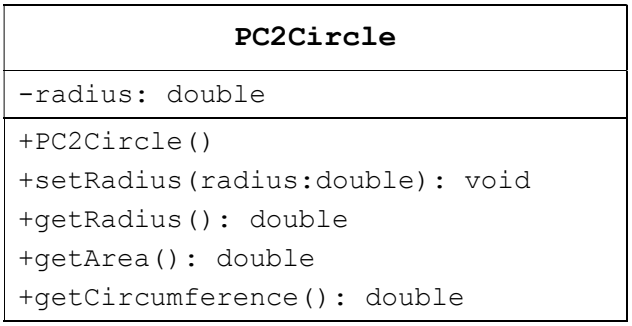
9#	PC2 CIRCLE CLASS	
	Input	Standard input
	Output	Standard output
	Topic	Classes : **Sample Solution**

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.



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` ($1 \le n \le 50$) which determines the number of test cases. Each of the following `n` lines contains a positive real number `r` ($0.0 \le r \le 500.00$) 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;
   }
}
```

9C	MY R-TRIANGLE	
	Input	Standard input
	Output	Standard output
	Topic	Classes : Simple problem solving

Problem Description

A class named `RTriangle` that represents a right angled triangle is defined as follows.

- Private instance variables named `width` and `height` that represents the width and height of the triangle.
- The constructor `RTriangle()` that initialized the `width` and `height` to 0.0.
- `setSide(double)` method to set a new side value
- `getSide()` method that returns the side value
- `getArea()` method that returns the area of the triangle.
- `getPerimeter()` method to returns the perimeter of the triangle.

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

RTriangle
-width: double -height: double
+RTriangle() +setWidth(width:double): void +setHeight(height:double): void +getWidth(): double +getHeight(): double +getArea(): double +getPerimeter(): double

Write the code for `RTriangle` class.

Write a program to test the `RTriangle` 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 two positive real number `w` and `h` ($0.0 \leq w, h \leq 500.00$) which represents the width and height of the right 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 width, height, area and perimeter of the triangle. (format the output in 2 decimal places).

Sample Input Output

Sample Input	Sample Output
2 9.0 7.33 5.5 8.88	Case #1: 9.00 7.33 32.98 27.94 Case #2: 5.50 8.88 24.42 24.83

Use the following program structure:

```
// File name: TestRTriangle.java

// import statements

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

      // code for test class
   }
}

class RTriangle {    // NOT a public class

   // code for RTriangle class
}
```

9D	MY TIME	
	Input	Standard input
	Output	Standard output
	Topic	Classes : Problem solving

Problem Description

A class called `MyTime`, which represent time in 24-hour notation are described below.

`MyTime` contains the following private instance variables that represent a valid time in 24-hour notation:

- `hour`
- `minute`

Constructor methods

- `MyTime();` // initialised all instance variable to 0
- `MyTime(int h, int m)` // initialised instance variables with the values

that invoke the `setTime()` method (described below) to set the instance variables.

`MyTime` contains the following public methods:

- `setTime(int h, int m)`: sets the instance variables.
- Setter methods: `setHour(int h)`, `setMinute(int m)`
- Getter methods: `getHour()`, `getMinute()`.
- `incrementMinutes(int min)` : add `min` to the instance variable `minute`. Update the instance variables to a valid range.
- `incrementHours(int hrs)` : add `hrs` to the instance variable `hour`. Update the instance variables to a valid range.
- `toString()`: returns "HH:MM".

Write the code for the `MyTime` class.

Write a test program named `TestMyTime` that reads two time values that represent the start time and duration for an event. Your program should calculate the end time of the event.

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 the start time (valid time in 24-hour notation) and a duration (in hour and minute) as shown in the sample input.

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 end time of the event in a valid 24-hour time notation.

Sample Input Output

Sample Input	Sample Output
2 8 10 1 01 12 20 5 50	Case #1: 09:11 Case #2: 18:10