Lab 1
Classes and Object

A	CIRCLE	
	Input	Standard input
	Output	Standard output
	Topic	Classes

Problem Description

A class named `Circle` is defined as follows.

- Private instance variable named `radius` that represents the radius of the circle.
- The constructor `Circle()` that initialized 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 `Circle` is as the following.

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

Write the code for `Circle` class and a test program named `CircleApp` 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: 9.00 254.47 56.55 Case #2: 5.50 95.03 34.56

Use the following program structure:

```
// File name:   Circle.java

class Circle{

    // code for Rectangle class
}
```

```
// File name: CircleApp.java

// import statements

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

      // code for test class
    }
}
```

Notes: Please refer Tutorial Classes and Object Section C. You already discussed this in the tutorial session.

B	RECTANGLE	
	Input	Standard input
	Output	Standard output
	Topic	Classes

Problem Description

A class named `Rectangle` is defined as follows.

- Private instance variables named `width` and `height` that represents the width and height of the rectangle.
- The constructor `Rectangle()` that initialised `width` and `height` to 0.0 respectively.
- `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 rectangle.
- `getPerimeter()` method to returns the perimeter of the rectangle.

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

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

Write the code for `Rectangle` class and a test program named `RectangleApp` 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 rectangle.

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 rectangle. (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 65.97 32.66 Case #2: 5.50 8.88 48.84 28.76

Use the following program structure:

```
// File name:   Rectangle.java

class Rectangle{

    // code for Rectangle class
}
```

```
// File name: RectangleApp.java

// import statements

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

      // code for test class
    }
}
```

Notes: Please refer to Tutorial Classes and Object Section B. Get some references and ideas there.

C	SAVING ACCOUNT	
	Input	Standard input
	Output	Standard output
	Topic	Classes

Problem Description

A savings account enables you to save or deposit your money into an account and receive certain interest with no stated maturity. Available at all the banking institutions in Malaysia, interest earned on your balance is normally credited to your account every month. The minimum deposit to open a savings account varies from one banking institution to another and can be as low as RM1.

A savings account class called `SavingAccount` is defined as follows.

- Private instance variable called `balance` represents the account balance.
- The constructor `SavingAccount(double amount)` that sets the amount it receives as the initial value of the account balance.
- `addInterest()` method that adds a 3% interest to the account balance.
- `getBalance()` method that returns the account balance.

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

SavingAccount
balance:double
+SavingAccount(double amount) +addInterest():void +getBalance():double

Write the code for `SavingAccount` class and a test program to test `SavingAccountApp` class 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 ($1 \leq n \leq 20$) which indicates the number test cases. Each of the following n lines contains a decimal points number represent the amount deposited (in Ringgit Malaysia) when opening a savings account.

Output

For each test case, output a line in the format "Case #x:" where x is the case number (starting from 1), followed by the account balance, which is a number with decimal points that represents an amount in Ringgit Malaysia.

Sample Input Output

Sample input	Sample output
3 1.00 150.00 0	Case #1: 1.03 Case #2: 154.50 Case #3: 0.00

Use the following program structure:

```
// File name: SavingAccount.java

public class SavingAccount{
    // code for SavingAccount class
}
```

```
// File name: SavingAccountApp.java
// import statements

public class SavingAccountApp{
   public static void main(String [] args){
      // code for test class
    }
}
```

Notes: Please refer to Tutorial Classes and Object Section B. Get some references and ideas there.

D	MY PRINTCARD	
	Input	Standard input
	Output	Standard output
	Topic	Classes

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` that 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`.

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

MyPrintCard
-balance: double
+MyPrintCard() +getBalance(): double +topupCard(double amt): void +payService(double amt): boolean

Write the code for `MyPrintCard` class and a test program named `MyPrintCardApp` (provided) 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*** ≤ 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

Use the following program structure:

```
// File name:   MyPrintCard.java

class MyPrintCard{

   // code for MyPrintCard class
}
```

```
// File name:   MyPrintCardApp.java
```

```java
import java.util.Scanner;
import java.text.DecimalFormat;
public class MyPrintCardApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double amt;
        int N = sc.nextInt();
        int x = 1;
        do {
            System.out.println("Case #" + x + ":");
            MyPrintCard card = new MyPrintCard();
            for (int i = 1; i <= N; i++) {
                char ch = sc.next().charAt(0); if (ch == '=')
                    System.out.println(card.toString());
                else if (ch == '-') {
                    amt = sc.nextDouble();
                    if (card.payService(amt))
                        System.out.println(card.toString());
                    else
                        System.out.println("invalid");
                }
                else if (ch == '+') {
                    amt = sc.nextDouble();
                    card.topupCard(amt);
                    System.out.println(card.toString());
                }
            }
            N = sc.nextInt();
            x++;
        }
        while (N != 0);
    }
}
```

Reflection:

After doing some tasks tutorial and lab, what do you understand about Classes and Object? Make some simple discussion with your instructor (*depends on timing and it is not compulsory*).