**Name:** Shradha Mallikarjun Patil

**USN:** 2GI20CS144

**Termwork 6:**

**Practice Problem 6.2:**

**Implement the following inheritance hierarchy.**

Rectangle

(Concrete class)

length, width: float

Circle

(Concrete class)

radius: float

Triangle

(Concrete class)

base, height: float

Shape (Abstract class)

area,perimeter: float

type: String

abstract computeArea()

abstract computePerimeter()

**Program:**

import java.util.Scanner;

abstract class Shape {

double area,perimeter;

String type;

public abstract void computeArea();

public abstract void computePerimeter();

public void display() {

     System.*out*.println(type);

     System.*out*.println("Area : "+area); System.*out*.printf("Perimeter : %.3f\n",perimeter);

}

}

class SixCircle extends Shape{

double radius;

public SixCircle() {

     Scanner sc = new Scanner(System.*in*);

     System.*out*.print("Enter the radius of the circle : ");

     radius = sc.nextDouble();

     super.type = "circle";

}

public void computeArea() {

     area = Math.*PI*\*radius\*radius;

}

public void computePerimeter() { perimeter = 2\*Math.*PI*\*radius;

}

}

class SixRectangle extends Shape {

double length,breadth;

public SixRectangle() {

     Scanner sc = new Scanner(System.*in*);

     System.*out*.print("Enter the length of the rectangle : ");

     length = sc.nextDouble();

     System.*out*.print("Enter the breadth of the rectangle : ");

     breadth = sc.nextDouble();

     super.type = "rectangle";

}

public void computeArea() {

     area = length\*breadth;

}

public void computePerimeter() { perimeter = 2\*(length+breadth);

}

}

class SixTriangle extends Shape {

double base,height;

public SixTriangle() {

     Scanner sc = new Scanner(System.*in*);

     System.*out*.print("Enter the base of the triangle : ");

     base = sc.nextDouble();

     System.*out*.print("Enter the height of the triangle : ");

     height = sc.nextDouble();

     super.type = "triangle";

}

public void computeArea() {

     area = 0.5\*base\*height;

}

public void computePerimeter() { perimeter = 2\*height+base;

}

}

public class tw6practice {

public static void main(String[] args) {

     Shape s;

     SixCircle c = new SixCircle();

     SixRectangle r = new SixRectangle();

     SixTriangle t = new SixTriangle();

     s = c;

     s.computeArea();

     s.computePerimeter();

     s.display();

     System.*out*.println();

     s = r;

     s.computeArea();

     s.computePerimeter();

     s.display();

     System.*out*.println();

     s = t;

     s.computeArea();

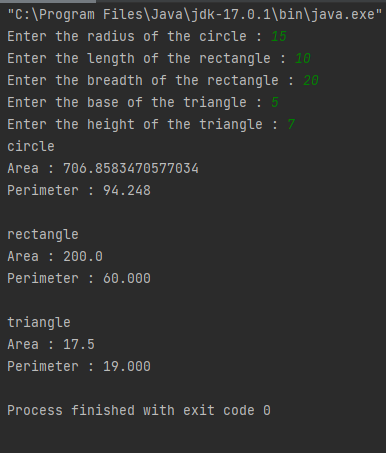
     s.computePerimeter();

     s.display();

}

}

**Output:**

****

**Practice Problem 6.3:**

Write a Java program that defines an abstract class called Account and accepts the following customer account information:  **1) Customer Name   
2) Account Number   
3) Balance**and provides below operations on customer account:   
 **1) Deposit   
2) Withdraw   
3) Display Balance   
4) Display full account details**

­There are two types of accounts – Savings and Current. The Current account has an overdraft facility limited to Rs. 75,000 per account. The following constraints hold on these accounts:

Savings Account:

1. The total number of deposits for a Savings account cannot exceed three per day.
2. The amount deposited into a savings account cannot exceed Rs.5000 in each transaction.
3. The maximum withdrawal amount is Rs.25,000 per transaction.
4. The minimum balance to be maintained is Rs. 10,000.

Current account:

1. The amount withdrawn cannot exceed the overdraft limit once the account balance is zero.
2. Maximum number of withdrawals is two.
3. No limit on the number of deposits.
4. Each deposit cannot exceed Rs. 25,000.

Test the program by creating objects of the Savings and Current Accounts.

**Program:**

abstract class Account {  
 protected String cusName,acNo;  
 protected float bal;  
 public abstract void Deposit(float dep);  
 public abstract void Withdraw(float wdw);  
 public void dispBal() {  
 System.out.println("Balance="+bal);  
 }  
 public void disp() {  
 System.out.println("Customer details are:");  
 System.out.println("Customer Name:"+cusName+"\nAccount No.:"+acNo);  
 System.out.println("Balance:"+bal);  
 }  
}  
  
class Savings extends Account {  
 private int n=0;  
 public Savings(String c,String ac,float b) {  
 cusName=c;  
 acNo=ac;  
 bal=b;  
 }  
 public void Deposit(float dep) {  
 n++;  
 if(n<=3) {  
 if(dep<=5000) {  
 bal=bal+dep;  
 System.out.println("Amount has been deposited");  
 dispBal();  
 }  
 else  
 System.out.println("Amount exceeds the limit");  
  
 }  
 else  
 System.out.println("You've reached maximum limit of deposition");  
  
 }  
 public void Withdraw(float wdw) {  
 if((bal-wdw)<10000)  
 System.out.println("Insufficient balance");  
 else {  
 if(wdw>25000)  
 System.out.println("Amount exceeds maximum withdrawal limit");  
 else {  
 bal=bal-wdw;  
 System.out.println("Amount successfully withdrawn");  
 dispBal();  
 }  
 }  
 }  
}  
  
class Current extends Account {  
 private int n=0;  
 public Current(String c,String ac,float b) {  
 cusName=c;  
 acNo=ac;  
 bal=b;  
 }  
 public void Deposit(float dep) {  
 bal=bal+dep;  
 System.out.println("Amount has been deposited");  
 dispBal();  
  
 }  
 public void Withdraw(float wdw) {  
 n++;  
 if(n<=2) {  
 if((bal-wdw)<-75000) {  
 n--;  
 System.out.println("Amount exceeds overdraft amount");  
 }  
 else {  
 bal=bal-wdw;  
 System.out.println("Amount successfully withdrawn");  
 dispBal();  
  
 }  
 }  
 else  
 System.out.println("You've exceeded withdrawal limit");  
 }  
}  
public class tw6point3 {  
 public static void main(String[] args) {  
 Savings s=new Savings("Finneas","221B",28000);  
 s.disp();  
 System.out.println("Account Type:Savings");  
 s.Deposit(2000);  
 s.Deposit(1000);  
 s.Deposit(2500);  
 s.Deposit(3000);  
 s.Withdraw(1000);  
 s.disp();  
  
 System.out.println();  
  
 Current c=new Current("Elton","241B",0);  
 c.disp();  
 System.out.println("Account Type:Current");  
 c.Withdraw(20000);  
 c.Withdraw(56000);  
 c.Withdraw(1000);  
 c.Withdraw(1000);  
 c.Deposit(30000);  
 c.disp();  
 }  
}

**Output :**

