Task # 1: Write a program which implements a interface of Banking System by having all standard functionalities and will be implemented by branches.

Hint:(Interface Methods)
CreateAccount()
Search Account details()
Update CustInfo()
Cash Withdraw()
Cash Deosit()

Solution:

CLASS INTERFACE

```
interface  BankingSystem
{
  public void CreateAccount();
  public void SearchAccountdetails();
  public void UpdateCustInfo();
  public void CashWithdraw();
  public void CashDeosit();
}
```

CLASS ATM

```
class branch1 implements BankingSystem
{
   Scanner input = new Scanner(System.in);

   float balance,accnum,index=1;
   String name;
   public void CreateAccount()
    {
       System.out.println("Enter Your Name: ");
       name=input.next();
       System.out.println("An bank account has been opened");
       System.out.println("Name: "+name);
       accnum= 4344343+index;
       System.out.println("accnum: "+accnum);
       index += 1;
    }
  public void SearchAccountdetails()
  {
     System.out.println("Enter account number: ");
     float account = input.nextFloat();
     if(account == accnum)
```

```
    {
       System.out.println("Account Found.");
       System.out.println("Name: "+name);
      System.out.println("Account Number: "+accnum);
      System.out.println("Balance: "+balance);
    }}
  public void UpdateCustInfo()
  {
     System.out.println("Enter Your Name: ");
      name=input.next();
       System.out.println("Your name linked with your bank account is updated");
  }
  public void CashWithdraw()
  {
     System.out.println("Enter amount: ");
     float amount = input.nextFloat();
     if(balance>=amount)
     {
        System.out.println(amount+" Rupees has been withdrawned from your account");
        balance -= amount;
     }}
  public void CashDeosit()
  {
     System.out.println("Enter amount: ");
     float amount = input.nextFloat();
     System.out.println(amount+" Rupees has been deposited to your account");
     balance += amount;
  }}
```

MAIN METHOD

```
public class JavaApplication22 {
   public static void main(String[] args) {

       Scanner input = new Scanner(System.in);
       branch1 c1 = new branch1();
       String choice;
       String ans="yes";
       while (ans.equals("yes") || ans.equals("Yes"))
       {
       System.out.println("Enter 'A' to create account\nEnter 'B' to search account\nEnter 'C' to
deposit money\nEnter 'D' to withdraw money\nEnter 'E' to update account info\n");
       choice = input.next();
       switch(choice)
       {
```

```
        case "A":  case "a":
            c1.CreateAccount();
            break;
        case "B": case "b":
            c1.SearchAccountdetails();
            break;
        case "C": case "c":
            c1.CashDeosit();
            break;
        case "D": case "d":
            c1.CashWithdraw();
            break;
        case "E": case "e":
            c1.UpdateCustInfo();
            break;
        default:
            System.out.println("Sorry invalid option selected");
    }
      System.out.println("Do you want to run this again ?");
      ans=input.next();
    }}}
```

Output:

```
run:
Enter 'A' to create account
Enter 'B' to search account
Enter 'C' to deposit money
Enter 'D' to withdraw money
Enter 'E' to update account info

A
Enter Your Name:
FAROOQ
An bank account has been opened
Name: FAROOQ
accnum: 4344344.0
Do you want to run this again ?
yes
Enter 'A' to create account
Enter 'B' to search account
Enter 'C' to deposit money
Enter 'D' to withdraw money
Enter 'E' to update account info

C
Enter amount:
1000
1000.0 Rupees has been deposited to your account
Do you want to run this again ?
no
BUILD SUCCESSFUL (total time: 42 seconds)
```

Task # 2: By looking at the formulae for an ellipse, provide the missing code for all of the methods in the class Ellipse including the toString() method. Test your program using the TestShapes.java class. Your output should look as follows (for an ellipse with a = 10 and b = 7) (values are randomly generated).

Solution:

CLASS INTERFACE

```
interface Eccentric {
    double eccentricity();
}
```

CLASS SHAPE

```
abstract class Shape{

    public String name(){
        return getClass().getName();
    }
    public abstract double area();

    public abstract double perimeter();

    public String toString() {
return "\n" +name() +"\n Area=" +area() +"\nPerimeter=" +perimeter();

   }
}
```

CLASS CIRCLE

```
class Circle extends Shape {
    private double radius;

   public Circle(double r){
        radius = r;
   }
    public double area(){
        return Math.PI * (radius * radius);
    }
    public double perimeter(){
        return 2.0 * Math.PI * radius;
    }
    public double getRadius(){
        return radius;
    }
}
```

CLASS RECTANGLE

```
class Rectangle extends Shape {
    private double length;
    private double width;
    public Rectangle(double length, double width){
        this.length = length;
        this.width = width;
    }
    public double area(){
        return length * width;
    }
    public double perimeter(){
        return 2*(length+width);
    }
    public double getLength(){
        return length;
    }
    public double getWidth(){
        return width;
    }
}
```

CLASS SQUARE

```
class Square extends Rectangle{
    public Square(double length){
        super(length, length);
    }
}
```

CLASS EQUILATERAL TRIANGLE

```
class EquilateralTriangle extends Shape
    {
        private double side;
        public EquilateralTriangle(double side)
        {
            this.side=side;
        }
        public double getSide()
        {
            return side;
        }
        public double area(){
         return ((0.25)*(side*side));
    }
```

```
    public double perimeter(){
        return 3*side;
    }
}
```

CLASS ELLIPSE

```
class Ellipse extends Shape implements Eccentric
{
    double a, b;
    public Ellipse(double s1, double s2){
        if(s1 < s2) {
            a = s2;
            b = s1;
        }
        else    {
            a = s1;
            b = s2;
        }
    }
    public double perimeter(){
        if (a == b)
        {
           double Perimeter = 2*3.14*a;
           return Perimeter;
        }
        else
        {
        double Perimeter = 3.14*(Math.sqrt(2*(a*a + b*b) - (a*a + b*b-2*a*b)/2));
        return Perimeter;
        }
     }
    public double area(){
        double Area = 3.14*a*b;
           return Area;
    }
    public double eccentricity(){
        double Eccentricity=0;
        if(a>b)
        {
           Eccentricity = (Math.sqrt(a*a-b*b))/a;
        }
        else if(b>a)
        {
           Eccentricity = (Math.sqrt(b*b-a*a))/b;
```

```
        }
            return Eccentricity;
    }
}
```

CLASS TEST SHAPE

```
public class TestShapes {
   public static Shape[] createShape() {
      final int SIZE = 5;
      final double DIMENSION = 100;
      final int NUMBEROFSHAPES = 5;

      Random generator = new Random();

      //create an array having b/w 1 and SIZE entries
      Shape[] randomShapes = new Shape[generator.nextInt(SIZE) + 1];

      for(int i = 0; i < randomShapes.length; i++)
      {
         //randomly generate values b/w 0 and NUMBEROFSHAPES - 1
         int assigner = generator.nextInt(NUMBEROFSHAPES);
         switch(assigner) {
            case 0: randomShapes[i] =
                  new
Rectangle(generator.nextDouble()*DIMENSION,generator.nextDouble()*DIMENSION);
break;
 case 1: randomShapes[i] = new Circle(generator.nextDouble()*DIMENSION);
          break;
case 2: randomShapes[i] = new Square(generator.nextDouble()*DIMENSION);
          break;
            case 3: randomShapes[i] = new EquilateralTriangle(generator.nextDouble()*DIMENSION);
          break;
            case 4: randomShapes[i] = new Ellipse
(generator.nextDouble()*DIMENSION,generator.nextDouble()*DIMENSION);
          break;
               }
          }
          return randomShapes;
    }
```

MAIN METHOD

```
    public static void main(String[] args){
          Shape[] randomShapes = TestShapes.createShape();

          for(int i = 0; i < randomShapes.length; i++){
```

```
                System.out.println(randomShapes[i]);

if(randomShapes[i] instanceof Circle)
{
        System.out.println("Radius= " + ((Circle) randomShapes[i]).getRadius());
}
else if(randomShapes[i] instanceof Square)
{
   System.out.println("Length= " +
((Square) randomShapes[i]).getLength());
}
else if(randomShapes[i] instanceof Rectangle)
{
   System.out.println("Length= " +
((Rectangle) randomShapes[i]).getLength()
                + "\nWidth= " +
((Rectangle) randomShapes[i]).getWidth());
}
else if(randomShapes[i] instanceof Ellipse)
{
  System.out.println("Eccentricity = "+
  ((Ellipse) randomShapes[i]).eccentricity());
}
else if(randomShapes[i] instanceof EquilateralTriangle)
{
                System.out.println("Each Side= " +
((EquilateralTriangle) randomShapes[i]).getSide());
}
        }
    }
}
```

Output:

```
javaapplication20.Ellipse
 Area=3655.3753150073962
Perimeter=350.54722994273766
Eccentricity = 0.9873717406051102

javaapplication20.Square
 Area=3086.7003563383983
Perimeter=222.23232370970334
Length= 55.558080927425834
```