//Lu Wang

//19230509

//Assignment7

```
Choose Shape1: [1.Circle 2.Rectangle 3.Random num exit input]1
Input Radius: 5
Choose Shape1: [1.Circle 2.Rectangle 3.Random num exit input]2
Input Width: 3
Input Length: 4
Choose Shape1: [1.Circle 2.Rectangle 3.Random num exit input]1
Input Radius: 2
Choose Shape1: [1.Circle 2.Rectangle 3.Random num exit input]9


Circle [radius=5.0,area=78.5]
Rectangle [length=4.0, width=3.0,area=12.0]
Circle [radius=2.0,area=12.56]

-------------------------------------

 Largest shape in the collection is:
Circle [radius=5.0,area=78.5]
-------------------------------------
```

Parent Class:Shapes

```
package assignment7;

//Create a parent Class:Shapes with an Interface:ShapesRelate

abstract class Shapes implements ShapesRelate {

    //Protected type can be inherited by subclass

    protected double area;


     //Constructor of Shapes

    public Shapes() {
```

```
    }
    public double getArea() {
        return area;
    }
     //Create an abstract method: calculateArea
     //Note that subclass must override all the abstract methods in father,
then the subclass can be Instantiated.
    abstract void calculateArea();
       //Implementation the interface
    public int compareShapes(ShapesRelate ss) {
        //Force cast new object:ss to Shapes Class and put into temp
          Shapes temp=(Shapes)ss;
          //Compare area: if current area<new area, return value is 1
        if(this.area<temp.getArea()) {
            return 1;
        }
        //Compare area: if current area<=new area, return value is 0
        else {
            return 0;
        }
    }
```

Interface:ShapesRelate

```
package assignment7;


//Create an interface class called ShapesRelate
public interface ShapesRelate {
    // Declare compareShapes method name:compareShapes
    public int compareShapes(ShapesRelate ss);
}
```

SubClass:Circle

```
package assignment7;



//Create a subclass:Circle inherited father class:Shapes
//Plz be aware circle class must implement father class's abstract method
//Or subclass will be an abstract class as father:shapes.
public class Circle extends Shapes {
    private double radius;
    private double PI=3.14;
      //Constructor:
    public Circle() {
        super();
```

```
}

public Circle(double r) {
    this.setRadius(r);
}

public void setRadius(double radius) {
    this.radius = radius;
}

public double getRadius() {
    return radius;
}
//Override to implement Parent calculateArea Method
public void calculateArea() {
    super.area = PI * radius*radius;
}

//The inherited interface must be override
public int compareShapes(ShapesRelate ss) {
    return 0;
```

```
    }
     @Override
        public String toString() {
            return "\nCircle [radius=" + radius + ",area="+area+"]";
        }
```

Subclass:Rectangle

```
package assignment7;
//Create a subclass:Rectangle inherited father class:Shapes
//Plz be aware Rectangle class must implement father class's abstract method
//Or subclass will be an abstract class as father:shapes.
public class Rectangle extends Shapes {

    private double length;
    private double width;

    // Constructor
    public Rectangle() {
        super();
    }

    public Rectangle(double length, double width) {
        this.length = length;
```

```
        this.width = width;
    }

    // Method
    public void setWidth(double width) {
        this.width = width;

    }

    public double getWidth() {
        return width;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public double getLength() {
        return length;
    }

    // Override Parent Class:Shapes calculateArea Method
```

```
    public void calculateArea() {
        super.area = length * width;
    }

    // The inherited interface must be override
    public int compareShapes(ShapesRelate ss) {
        return 0;
    }

    // Override toString
    public String toString() {
        return "\nRectangle [length=" + length + ", width=" + width +
",area="+area+"]";
    }
```

Driver Class

```
package assignment7;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Driver {
```

```
    public static void main(String[] args) {// Start main
        // Screen input declaration:
        Scanner scan = new Scanner(System.in);
        // Create ArrayList
        ArrayList<Shapes> shapes = new ArrayList<Shapes>();
        // Initialize num=0
        int num = 0;
        // Infinite loop choose shape and related variables
        while (true) {
            System.out.print("Choose Shape" + (num + 1) + ": [1.Circle
2.Rectangle 3.Random num exit input]");
            int t = scan.nextInt();
            scan.nextLine();
            if (t == 1) {
                // Create instant:circle
                Circle circle = new Circle();
                System.out.print("Input Radius: ");
                int r = scan.nextInt();
                scan.nextLine();
                // setRadius:r of Circle class
                circle.setRadius(r);
                // Call calculateArea of to calculate Circle area
```

```
        circle.calculateArea();
        shapes.add(circle);
    }
    if (t == 2) {
        // Create instant:rectangle
        Rectangle rectangle = new Rectangle();
        System.out.print("Input Width: ");
        int w = scan.nextInt();
        scan.nextLine();
        rectangle.setWidth(w);
        System.out.print("Input Length: ");
        int l = scan.nextInt();
        scan.nextLine();
        rectangle.setLength(l);
        // Call calculateArea of to calculate Rectangle area
        rectangle.calculateArea();
        shapes.add(rectangle);
    }
        //input other numbers to end input
    if (t != 1 && t != 2) {
        break;
    }
```

```
        }
        System.out.print("\n");
        for (int i = 0; i < shapes.size(); i++) {
            System.out.print(shapes.get(i));
        }
        System.out.print("\n");
        // Call below largestShape method
        largestShape(shapes);
    }// end main
    // largestShape method to find out the largest area
    public static Shapes largestShape(ArrayList<Shapes> list) {
        // Create a new object:max and randomly choose object:0 as the current
largest box
        Shapes max = list.get(0);
        // for loop to choose each compare's larger area
        for (int i = 0; i < list.size() - 1; i++) {
            // Call parent's interface compareShapes method,if return value is 1
            if (max.compareShapes(list.get(i)) == 1) {
                // Store the current object into largest box
                max = list.get(i);
            }
        }
```

```
            System.out.print("\n-----------------------------------\n");
            System.out.print("\n Largest shape in the collection is: " + max);
            System.out.print("\n-----------------------------------\n");
            return max;
        }// end method
}// end class
```