|  |  |  |  |
| --- | --- | --- | --- |
| ROLL NO: | 22BCP317 | Batch | G8 |
| NAME: | Patel shiv vijaykumar | | |
| Practical | 2/2 | | |
| Aim: | package creation. | | |

**Creating package**

package pdeu.drawing;

public abstract class Shape {

private String color;

private String pattern;

public Shape(String color, String pattern) {

this.color = color;

this.pattern = pattern;

}

public abstract double calculateArea();

public abstract double calculatePerimeter();

public static int countShapes() {

return 0;

}

public String getColor() {

return color;

}

public String getPattern() {

return pattern;

}

}

class Square extends Shape {

private double side;

public Square(String color, String pattern, double side) {

super(color, pattern);

this.side = side;

}

@Override

public double calculateArea() {

return side \* side;

}

@Override

public double calculatePerimeter() {

return 4 \* side;

}

}

class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(String color, String pattern, double length, double width) {

super(color, pattern);

this.length = length;

this.width = width;

}

@Override

public double calculateArea() {

return length \* width;

}

@Override

public double calculatePerimeter() {

return 2 \* (length + width);

}

}

class Circle extends Shape implements Resizable {

private double radius;

public Circle(String color, String pattern, double radius) {

super(color, pattern);

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

@Override

public double calculatePerimeter() {

return 2 \* Math.PI \* radius;

}

@Override

public void resize(int factor) {

radius \*= factor;

}

}

class Triangle extends Shape {

private double side1;

private double side2;

private double side3;

public Triangle(String color, String pattern, double side1, double side2, double side3) {

super(color, pattern);

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

@Override

public double calculateArea() {

double s = (side1 + side2 + side3) / 2;

return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

@Override

public double calculatePerimeter() {

return side1 + side2 + side3;

}

}

interface Resizable {

void resize(int factor);

}

**Source code:**

package pdeu.drawingTest;

import pdeu.drawing.\*;

public class TestDrawing {

public static void highestArea(Shape[] sp) {

double maxArea = 0;

Shape shapeWithMaxArea = null;

for (Shape shape : sp) {

double area = shape.calculateArea();

if (area > maxArea) {

maxArea = area;

shapeWithMaxArea = shape;

}

}

if (shapeWithMaxArea != null) {

System.out.println("Shape with the highest area: " + shapeWithMaxArea.getClass().getSimpleName());

System.out.println("Color: " + shapeWithMaxArea.getColor());

System.out.println("Pattern: " + shapeWithMaxArea.getPattern());

System.out.println("Area: " + maxArea);

} else {

System.out.println("No shapes in the array.");

}

}

public static void resizableShapes(Shape[] sp) {

for (Shape shape : sp) {

if (shape instanceof Resizable) {

System.out.println("Resizable shape: " + shape.getClass().getSimpleName());

System.out.println("Color: " + shape.getColor());

System.out.println("Pattern: " + shape.getPattern());

}

}

}

public static double totalDecorativeMaterialForCircle(Shape[] sp) {

double totalMaterial = 0;

for (Shape shape : sp) {

if (shape instanceof Circle) {

double area = shape.calculateArea();

totalMaterial += area;

}

}

return totalMaterial;

}

public static void main(String[] args) {

Shape[] shapes = new Shape[] {

new Square("Red", "Striped", 5.0),

new Rectangle("Blue", "Dotted", 4.0, 6.0),

new Circle("Green", "Solid", 3.0),

new Triangle("Yellow", "Checkered", 7.0, 8.0, 9.0)

};

highestArea(shapes);

System.out.println();

resizableShapes(shapes);

double totalMaterial = totalDecorativeMaterialForCircle(shapes);

System.out.println("Total decorative material for circles: " + totalMaterial);

}

}

**Output:**

Shape with the highest area: Square

Color: Red

Pattern: Striped

Area: 25.0

Resizable shape: Square

Color: Red

Pattern: Striped

Resizable shape: Circle

Color: Green

Pattern: Solid

Total decorative material for circles: 28.2743338823081

|  |  |  |  |
| --- | --- | --- | --- |
| ROLL NO: | 22BCP317 | Batch | G8 |
| NAME: | Patel shiv vijaykumar | | |
| Practical | 2/3 | | |
| Aim: | Write a program to show the use of static functions and to pass variable length arguments in a function. | | |

**Source code:**

public class StaticFunctionWithVarargs {

static void displayNumbers(String message, int... numbers) {

System.out.print(message + ": ");

for (int num : numbers) {

System.out.print(num + " ");

}

System.out.println();

}

public static void main(String[] args) {

displayNumbers("Even Numbers", 2, 4, 6, 8, 10);

displayNumbers("Odd Numbers", 1, 3, 5, 7);

displayNumbers("Prime Numbers", 2, 3, 5, 7, 11, 13);

displayNumbers("Empty List");

int[] customNumbers = { 1, 4, 9, 16, 25 };

displayNumbers("Custom Numbers", customNumbers);

}

}

**Output:**

Even Numbers: 2 4 6 8 10

Odd Numbers: 1 3 5 7

Prime Numbers: 2 3 5 7 11 13

Empty List:

Custom Numbers: 1 4 9 16 25

|  |  |  |  |
| --- | --- | --- | --- |
| ROLL NO: | 22BCP317 | Batch | G8 |
| NAME: | Patel shiv vijaykumar | | |
| Practical | 2/4 | | |
| Aim: | Write a program to show the use of static functions and to pass variable length arguments in a function. | | |

**Source code:**

class timepass {

 int value;

 public timepass(int value) {

 this.value = value;

 }

}

public class practical2\_4 {

 public static void modifyValue(int x) {

 x = 42;

 }

 public static void modifyObjectValue(timepass obj) {

 obj.value = 42;

 }

 public static int add(int a, int b) {

 return a + b;

 }

 public static timepass createObject(int value) {

 return new timepass(value);

 }

 public static void main(String[] args) {

 int num = 10;

 System.out.println("Before Value: " + num);

 modifyValue(num);

 System.out.println("After Value: " + num);

 timepass myObj = new timepass(10);

 System.out.println("Before ObjectValue: " + myObj.value);

 modifyObjectValue(myObj);

 System.out.println("After ObjectValue: " + myObj.value);

 int sum = add(5, 7);

 System.out.println("Result of add method: " + sum);

 timepass newObj = createObject(100);

 System.out.println("Value of the created object: " + newObj.value);

 }

}

**Output:**

Before Value: 10

After Value: 10

Before ObjectValue: 10

After ObjectValue: 42

Result of add method: 12

Value of the created object: 100