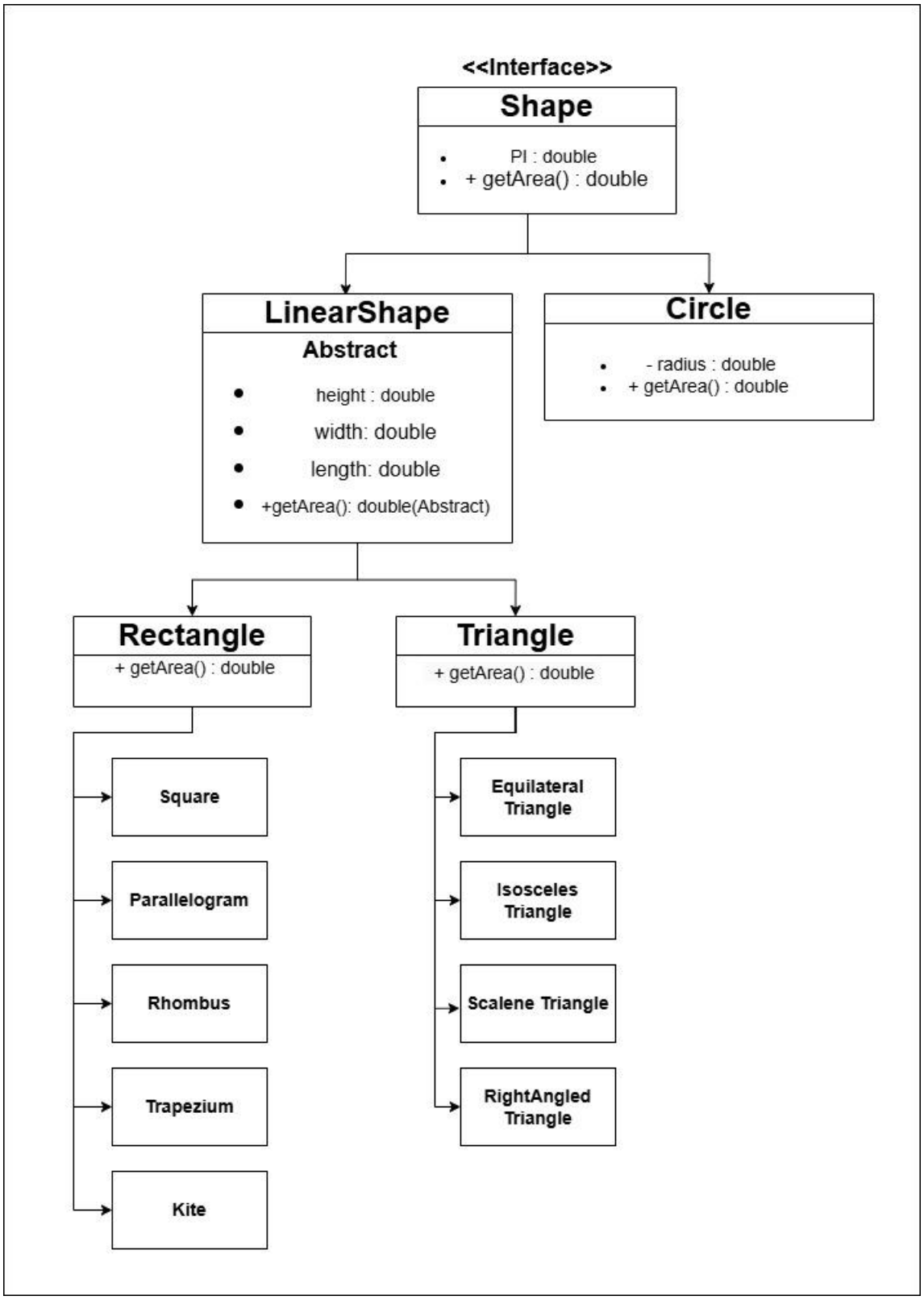
❖ **The problem Flow Chart:**



❖ **The Code Input in Java :**

Shape.aava

```
package bd.edu.bubt.cse;

public interface Shape {
    public static final double PI = 3.1416;
    double getArea();
}
```

Circle.java

```
package bd.edu.bubt.cse;

public class Circle implements Shape {
    private double radius;

    public Circle(double radius){
        this.radius = radius;
    }


    public double getArea() {
        return PI * radius * radius;
    }

    public double getRadius() {
        return radius;
    }
}
```

LinearShape.java

```
package bd.edu.bubt.cse;

public abstract class LinearShape implements Shape{
    protected double length;
    protected double width;
    protected double height;

    public LinearShape(double length, double width, double height){
        this.length = length;
        this.width = width;
        this.height = height;
    }
}
```

Rectangle.java

```
package bd.edu.bubt.cse;

public class Rectangle extends LinearShape {
    public Rectangle(double length, double width){
        super(length, width, 0);
    }


    public double getArea() {
        return length * width;
    }
}
```

Square.java

```
package bd.edu.bubt.cse;

public class Square extends Rectangle {
    private double side;

    public Square(double side){
        super(side, side);
        this.side = side;
    }
}
```

Parallelogram.java

```
package bd.edu.bubt.cse;

public class Parallelogram extends LinearShape {
    private double base;
    private double height;

    public Parallelogram(double base, double height) {
        super(base, 0, height);
        this.base = base;
        this.height = height;
    }


    public double getArea() {
        return base * height;
    }
}
```

Rhombus.java

```
package bd.edu.bubt.cse;

public class Rhombus extends LinearShape {
    private double base;
    private double height;

    public Rhombus(double base, double height){
        super(base, 0, height);
        this.base = base;
        this.height = height;
    }


    public double getArea() {
        return base * height;
    }
}
```

Trapezium.java

```
package bd.edu.bubt.cse;

public class Trapezium extends LinearShape {
    private double a, b, height;

    public Trapezium(double a, double b, double height){
        super(a, b, height);
        this.a = a;
        this.b = b;
        this.height = height;
    }


    public double getArea() {
        return ((a + b) / 2) * height;
    }
}
```

Kite.java

```
package bd.edu.bubt.cse;

public class Kite extends LinearShape {
    private double d1, d2;

    public Kite(double d1, double d2){
        super(d1, d2, 0);
        this.d1 = d1;
        this.d2 = d2;
    }
```

```
    public double getArea() {
        return (d1 * d2) / 2;
    }
}
```

Triangle.java

```
package bd.edu.bubt.cse;

public class Trangle extends LinearShape{

    public Trangle(double length, double width, double height){
        super(length, width, height);
    }


    public double getArea() {
        return 0.5 * length * height;
    }
}
```

EquilateralTriangle.java

```
package bd.edu.bubt.cse;

public class EquilateralTriangle extends Trangle {
    public EquilateralTriangle(double side){
        super(side, 0, (Math.sqrt(3)/2) * side);
    }


    public double getArea() {
        return (Math.sqrt(3)/4) * length * length;
    }
}
```

IsoscelesTriagle.java

```
package bd.edu.bubt.cse;

public class IsoscelesTriangle extends Trangle {
    public IsoscelesTriangle(double base, double height){
        super(base, 0, height);
    }
}
```

ScaleneTriangle.java

```
package bd.edu.bubt.cse;

public class ScaleneTriangle extends Trangle {
    private double sideB, sideC;

    public ScaleneTriangle(double a, double b, double c){
        super(a, 0, 0);
        this.sideB = b;
        this.sideC = c;
    }


    public double getArea() {
        double s = (length + sideB + sideC) / 2;
        return Math.sqrt(s * (s - length) * (s - sideB) * (s - sideC));
    }
}
```

RightAngledTriangle.java

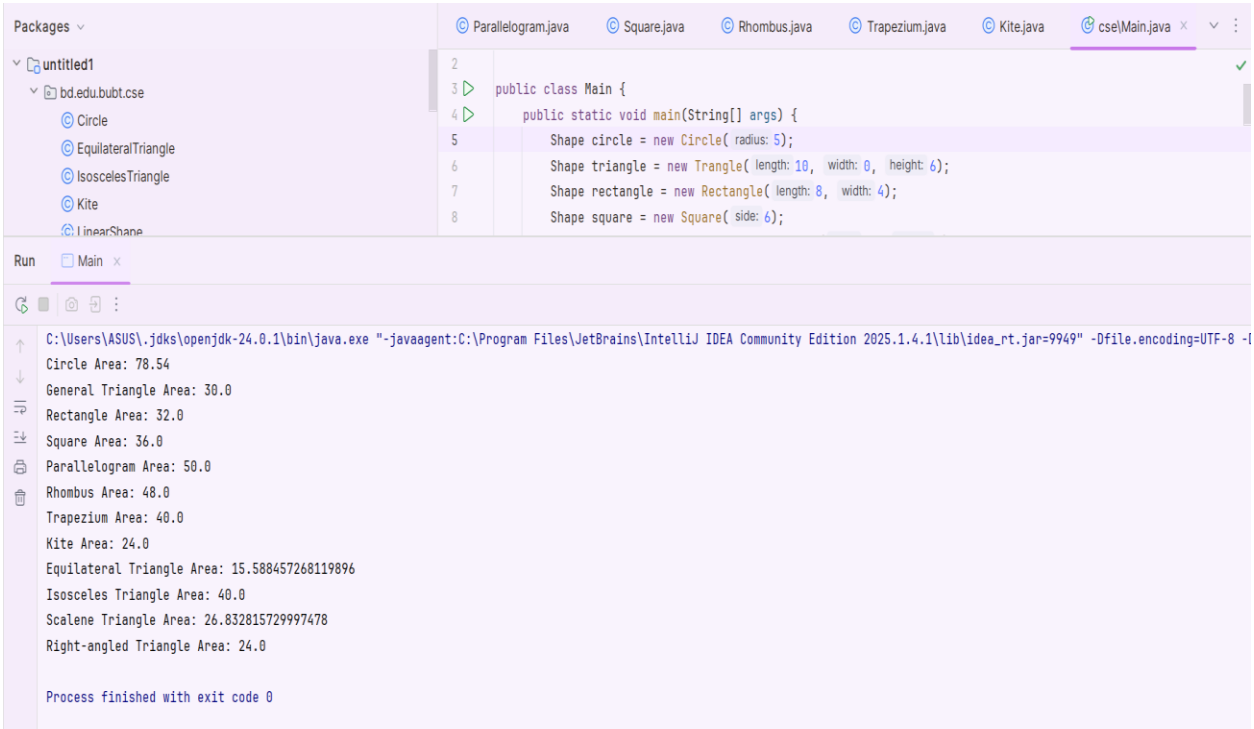
```
package bd.edu.bubt.cse;

public class RightAngledTriangle extends Trangle {
    public RightAngledTriangle(double base, double height){
        super(base, 0, height);
    }


    public double getArea() {
        return 0.5 * length * height;
    }
}
```

Main.java

```
package bd.edu.bubt.cse;

public class Main {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape triangle = new Trangle(10, 0, 6);
        Shape rectangle = new Rectangle(8, 4);
        Shape square = new Square(6);
        Shape parallelogram = new Parallelogram(10, 5);
        Shape rhombus = new Rhombus(8, 6);
        Shape trapezium = new Trapezium(10, 6, 5);
        Shape kite = new Kite(8, 6);

        Shape equilateral = new EquilateralTriangle(6);
        Shape isosceles = new IsoscelesTriangle(10, 8);
        Shape scalene = new ScaleneTriangle(7, 8, 9);
```

```
        Shape rightAngled = new RightAngledTriangle(6, 8);

        System.out.println("Circle Area: " + circle.getArea());
        System.out.println("General Triangle Area: " +
triangle.getArea());
        System.out.println("Rectangle Area: " + rectangle.getArea());
        System.out.println("Square Area: " + square.getArea());
        System.out.println("Parallelogram Area: " +
parallelogram.getArea());
        System.out.println("Rhombus Area: " + rhombus.getArea());
        System.out.println("Trapezium Area: " + trapezium.getArea());
        System.out.println("Kite Area: " + kite.getArea());

        System.out.println("Equilateral Triangle Area: " +
equilateral.getArea());
        System.out.println("Isosceles Triangle Area: " +
isosceles.getArea());
        System.out.println("Scalene Triangle Area: " +
scalene.getArea());
        System.out.println("Right-angled Triangle Area: " +
rightAngled.getArea());
    }
}
```

❖ The Output:



❖ Conclusion:

This project is designed using **Object-Oriented Programming (OOP)** principles to model different **geometric shapes** and calculate their areas.

1. **Shape Interface** – defines the contract (getArea()) that every shape must implement. This ensures **abstraction** and **polymorphism**.
2. **Circle Class** – directly implements Shape and calculates area using the formula:

$$Area = \pi r^2$$

3. **LinearShape (Abstract Class)** – acts as a **base class** for all polygon-based shapes that have length, width, or height. It enforces code reuse and provides a consistent structure.
4. **Rectangle Class** – extends LinearShape and calculates area as:

$$Area = length * width$$

5. **Square Class** – a special type of rectangle with all sides equal, passing (side, side) to the rectangle's constructor.
6. **Parallelogram Class** – uses **base × height** to calculate area.
7. **Rhombus Class** – uses the **base × height** formula (since it's a special parallelogram).
8. **Trapezium Class** – calculates area using:

$$Area = \frac{(a+b)}{2} * height$$

   where a and b are the lengths of parallel sides.

9. **Kite Class** – uses diagonals for calculation:

$$Area = \frac{d1*d2}{2}$$

10. **Triangle Hierarchy** – The Trangle base class calculates a general triangle's area using 0.5 × base × height. Its specializations override the formula:
    - **Equilateral Triangle** → $\sqrt{\frac{3}{4}} * a^2$
    - **Isosceles Triangle** → $\frac{1}{2} base * height$
    - **Scalene Triangle** → **Heron's Formula**
    - **Right-angled Triangle** → $\frac{1}{2} base * height$
11. **Main Class** – creates objects of all shapes, stores them in Shape references, and prints their areas. This demonstrates **polymorphism**, because the correct getArea() implementation is chosen at runtime based on the actual object type.