图形面积计算器重构实验报告

一、实验目的

1. 掌握抽象类与继承的用法，实现代码复用与统一接口。
2. 理解多态性，通过统一方法处理不同类型的图形对象。
3. 体会面向对象设计中的“组合 vs 继承”原则，并能结合实际场景进行分析。
4. 熟悉使用 Trae AI 辅助编程工具进行代码编写与调试。

二、实验环境

开发工具：Trae，利用其智能补全、代码生成与调试功能完成实验

JDK 版本：11+

三、实验内容与步骤

1. 设计抽象类 Shape
   定义抽象方法 getArea()，作为所有图形的公共行为规范。
2. 实现三个具体图形类
   分别创建 `Circle`、`Rectangle`、`Triangle` 类继承 `Shape`，并各自实现面积计算。
3. 编写工具类 `ShapeUtil`
   提供静态方法 `printArea(Shape shape)`，利用多态打印任意图形的面积。
4. 编写测试类 `Demo`
   创建不同图形对象，调用 `printArea` 验证结果。
5. 绘制类图
   使用 Mermaid 描述类间关系，展示继承与依赖。
6. 记录 AI 使用情况
   说明在 Trae 中如何借助 AI 完成代码、类图和报告的辅助生成。

四、核心代码与类图

1. 代码实现

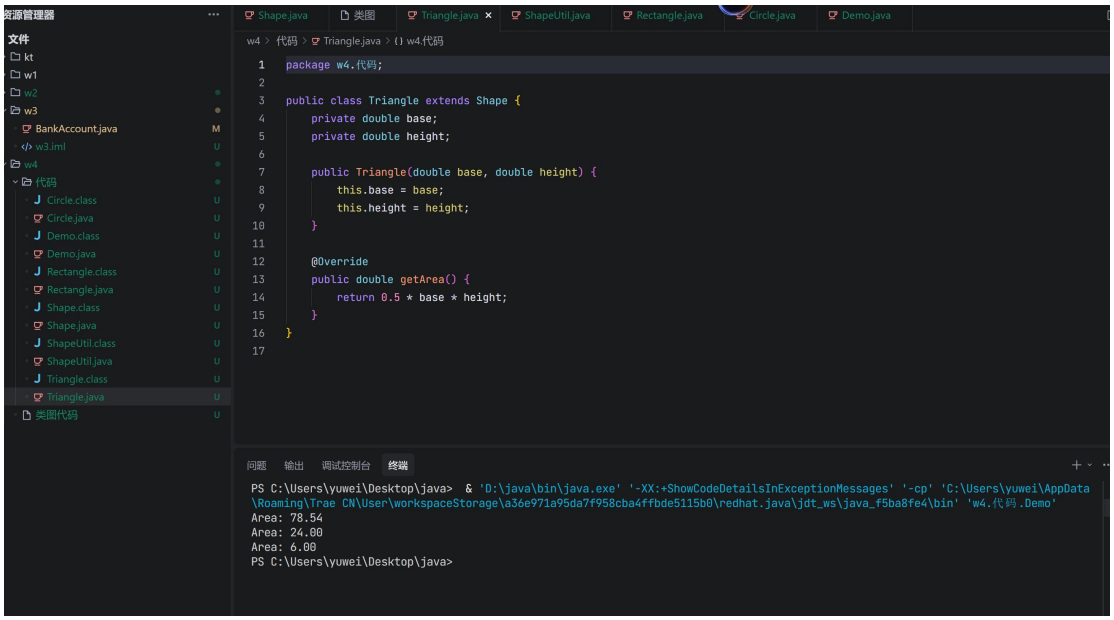
```java
// Shape.java
public abstract class Shape {
    public abstract double getArea();
}

// Circle.java
public class Circle extends Shape {
    private double radius;
    public Circle(double radius) { this.radius = radius; }
    @Override
    public double getArea() { return Math.PI * radius * radius; }
```

```
}

// Rectangle.java
public class Rectangle extends Shape {
	private double length, width;
	public Rectangle(double length, double width) {
		this.length = length;
		this.width = width;
	}
	@Override
	public double getArea() { return length * width; }
}

// Triangle.java
public class Triangle extends Shape {
	private double base, height;
	public Triangle(double base, double height) {
		this.base = base;
		this.height = height;
	}
	@Override
	public double getArea() { return 0.5 * base * height; }
}

// ShapeUtil.java
public class ShapeUtil {
	public static void printArea(Shape shape) {
		System.out.printf("图形面积：%.2f\n", shape.getArea());
	}
}

// Demo.java
public class Demo {
	public static void main(String[] args) {
		Shape circle = new Circle(5.0);
		Shape rectangle = new Rectangle(4.0, 6.0);
		Shape triangle = new Triangle(3.0, 4.0);
		ShapeUtil.printArea(circle);
		ShapeUtil.printArea(rectangle);
		ShapeUtil.printArea(triangle);
	}
}
```
```
```

2. 类图（Mermaid）

```mermaid
classDiagram
    class Shape {
        <<abstract>>
        +getArea() double*
    }
    class Circle {
        -radius : double
        +Circle(double)
        +getArea() double
    }
    class Rectangle {
        -length : double
        -width : double
        +Rectangle(double, double)
        +getArea() double
    }
    class Triangle {
        -base : double
        -height : double
        +Triangle(double, double)
        +getArea() double
    }
    class ShapeUtil {
        +printArea(Shape) void
    }
    Shape <|-- Circle
    Shape <|-- Rectangle
    Shape <|-- Triangle
    ShapeUtil --> Shape : uses
```

五、运行结果



六、AI 使用情况记录

本次实验在 Trae 环境中完成，AI 辅助主要体现在：

代码纠错调整

类图绘制：将代码粘贴后，AI 自动生成了 Mermaid 类图描述，并指导如何在 Trae 中预览。

报告整理：AI 根据实验内容生成报告初稿，并协助调整格式与语言。