湖南大學

HUNAN UNIVERSITY

高级程序设计实验二

实验名称： 图形面积计算器重构

学号： 202401070210

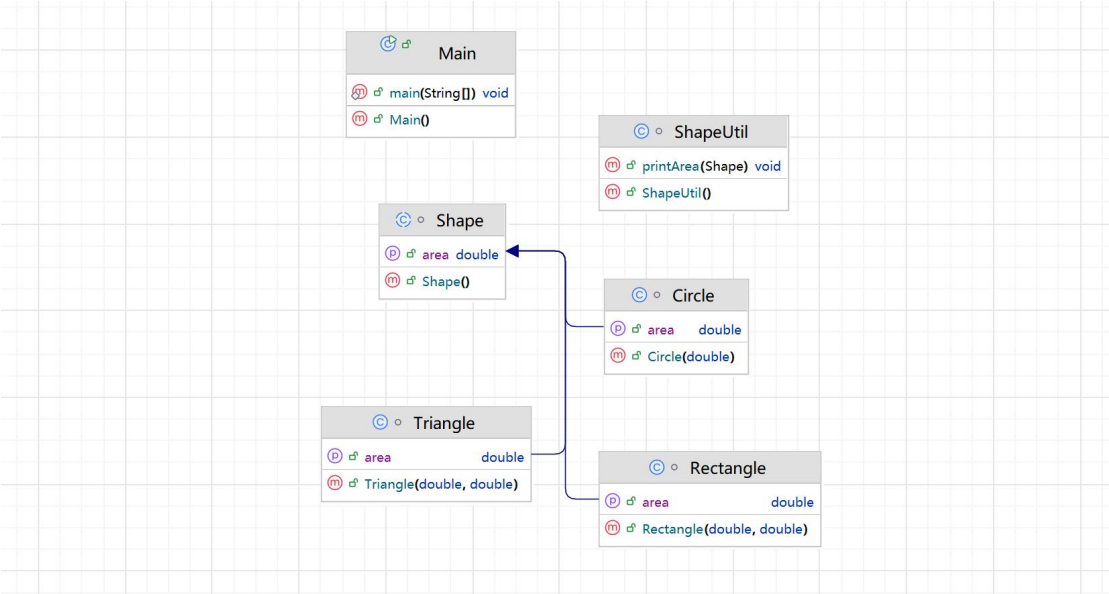
姓名： 郑诗艺

一、实验目的

1. 掌握抽象类与抽象方法的设计思想：通过定义抽象类 Shape 及抽象方法 getArea()，理解抽象类在统一接口规范中的作用，学会使用抽象类约束子类行为。
2. 理解继承与多态的核心应用：让 Circle、Rectangle、Triangle 类继承 Shape 并实现面积计算方法，体会 “is-a” 继承关系，掌握方法重写与多态调用的实现方式。
3. 实现统一处理的工具类设计：通过编写 ShapeUtil 工具类及 printArea(Shape shape) 方法，实践面向接口编程，实现对不同图形对象的统一处理，提升代码的可扩展性与复用性。
4. 理解类图的可视化表达：通过绘制类图，清晰呈现类之间的继承、依赖关系，掌握 UML 类图在面向对象设计中的作用。
5. 对比组合与继承的设计思想：在实验反思中分析 “组合 vs 继承” 的适用场景，理解两种设计模式的耦合度、复用方式与扩展灵活性差异，建立合理的面向对象设计思维。

二、类图

本类图展示了图形面积计算器的面向对象架构：抽象类 Shape 定义了面积计算的统一接口，Circle、Rectangle、Triangle 继承自 Shape 并实现各自的面积计算逻辑，体现了继承与多态特性；工具类 ShapeUtil 依赖 Shape 类型，通过 printArea() 方法实现对不同图形的统一处理，主类 Main 作为程序入口完成功能测试，整体设计解决了原有图形类无法统一处理的问题，符合面向对象的封装、继承与多态思想。



三、核心代码

1. 主类 Main

首先创建工具类 ShapeUtil 对象，用于统一处理面积输出；再分别创建圆形、矩形、三角形实例并传入参数；最后调用 util.printArea 方法完成面积计算与打印。

```
public class Main {
    public static void main(String[] args) {
        ShapeUtil util = new ShapeUtil();

        // 测试圆形
        Shape circle = new Circle( radius: 5);
        System.out.print("圆形（半径5）：");
        util.printArea(circle);

        // 测试矩形
        Shape rectangle = new Rectangle( length: 4,  width: 6);
        System.out.print("矩形（长4，宽6）：");
        util.printArea(rectangle);

        // 测试三角形
        Shape triangle = new Triangle( base: 3,  height: 4);
        System.out.print("三角形（底3，高4）：");
        util.printArea(triangle);
    }
}
```

2. 抽象类 Shape

Shape 为抽象类，使用 abstract 修饰，作为所有图形的父类。类中只定义抽象方法 getArea ()，没有具体实现，作用是为所有图形制定统一的计算接口，

强制子类必须实现面积计算方法，为后续多态调用奠定基础。

```
// 抽象类 Shape（不加 public）
abstract class Shape {  7 个用法  3 个继承者
    public abstract double getArea();  1 个用法  3 个实现
}

```

3. 圆形类 Circle

Circle 类继承自 Shape 抽象类，使用 private 封装半径 radius 属性，通过构造方法初始化数据。重写 getArea 方法，按照圆的面积公式 πr^2 实现计算，体现继承、封装和方法重写特性

```
// 圆形类
class Circle extends Shape {  1 个用法
    private double radius;  3 个用法

    public Circle(double radius) {  1 个用法
        this.radius = radius;
    }

    @Override  1 个用法
    public double getArea() {
        return Math.PI * radius * radius;
    }
}
```

4. 矩形类 Rectangle

Rectangle 继承 Shape 类，封装长和宽两个属性，构造方法用于初始化长宽数值。重写的 getArea 方法按照 “长 × 宽” 实现矩形面积计算，遵循父类接口规范，实现自身业务逻辑。

```
// 矩形类
class Rectangle extends Shape {  1 个用法
    private double length;  2 个用法
    private double width;  2 个用法

    public Rectangle(double length, double width) {  1 个用法
        this.length = length;
        this.width = width;
    }

    @Override  1 个用法
    public double getArea() {
        return length * width;
    }
}
```

5. 三角形类 Triangle

Triangle 继承 Shape 类，封装底和高属性，通过构造方法赋值。重写 getArea 方法使用 “底 × 高 ÷2” 计算三角形面积，与其他图形保持统一接口，实现多态基础。

```java
// 三角形类
class Triangle extends Shape {  1 个用法
    private double base;  2 个用法
    private double height;  2 个用法

    public Triangle(double base, double height) {  1 个用法
        this.base = base;
        this.height = height;
    }

    @Override  1 个用法
    public double getArea() {
        return (base * height) / 2;
    }
}
```

6. 工具类 ShapeUtil

ShapeUtil 是通用工具类，printArea 方法接收 Shape 类型参数，利用多态特性自动调用对应图形的 getArea 方法，无需区分具体图形类型，实现统一处理与格式化输出。该设计降低代码耦合度，方便扩展新图形，体现面向对象的多态优势。

```java
// 工具类
class ShapeUtil {  2 个用法
    public void printArea(Shape shape) {  3 个用法
        double area = shape.getArea();
        System.out.printf("该图形的面积为：%.2f%n", area);
    }
}
```

四、运行结果截图

程序运行后，控制台依次输出了三种图形的面积计算结果：圆形（半径 5）面积为 78.54，矩形（长 4、宽 6）面积为 24.00，三角形（底 3、高 4）面积为 6.00。该结果与数学公式计算完全一致，验证了代码逻辑的正确性。

从输出可以看出，工具类 ShapeUtil 通过多态特性实现了对不同图形的统一处理，无需区分具体图形类型即可完成面积计算与格式化输出，充分体现了面向对象设计中封装、继承与多态的优势，达成了 “统一处理不同图形” 的实验目标。

```
圆形（半径5）：该图形的面积为：78.54
矩形（长4，宽6）：该图形的面积为：24.00
三角形（底3，高4）：该图形的面积为：6.00
```

五、实验总结

1. 本次实验通过抽象类、继承与多态完成了图形面积计算器的重构，成功解决了原有 Circle、Rectangle、Triangle 类独立存在、无法统一处理的问题。通过定义抽象类 Shape 规范统一接口，让三个图形类继承并实现面积计算方法，再由工具类 ShapeUtil 基于多态实现统一的面积打印功能，最终达成了代码复用、统一处理与可扩展的设计目标，深刻理解了面向对象封装、继承、多态的核心思想。
2. 遇到的问题：初期未正确使用 abstract 修饰 Shape 类与 getArea()方法，导致无法约束子类必须实现面积计算逻辑。

   解决办法：查阅资料后明确抽象类与抽象方法的语法规范，将 Shape 类声明为 abstract，并在其中定义抽象方法 getArea()，强制子类重写该方法，保证了接口的统一性。