
廈門大學



信息学院软件工程系

《JAVA 程序设计》实验报告

实验 5

姓名：黄勛

学号：22920212204392

学院：信息学院

专业：软件工程

完成时间：2023.3.28

一、实验目的及要求

- 熟悉继承
- 熟悉多态

二、实验题目及实现过程

实验环境：Windows 10 21H2、jdk17、utf-8 编码

（基本题目）题目 1：

（一）实验题目

- a) 根据今日课程所学，完善实验 4 内容，并在实验报告中写明修改点及思路

（二）实现过程

关于学生选课系统的修改思路：主要需要设计的是一个选课系统的 java 实现，学生（包括本科生和研究生）可以选择必修课程和选修课程。（要求用到继承和多态）

- a) 学生分本科生（有学号、姓名、班级属性）和研究生（有学号、姓名、班级、导师属性）两种；
- b) 课程（有编号、课程名、学分属性）分必修和选修两种；
- c) 学生类中自动选课的方法：为每个学生自动选修所有必修课；
- d) 学生类中秘书手动选课部分的方法：为每个同学选修 1-2 门选修课；
- e) 打印出每个学生的选课信息的方法
- f) 创建 4 个学生信息（2 个本科生，2 个研究生）
- g) 创建 4 门课程信息（2 门必修，2 门选修）

部分的修改具体信息如下：

1. 由原来只有一个课程类拓展为两个类，继承自课程类

```

97      // 定义必修课程类，继承自课程类
      3个用法
98      class RequiredCourse extends Course {
99          // 构造函数
      2个用法
100         public RequiredCourse(String id, String name, double credit) {
101             super(id, name, credit);
102         }
103     }
104
105     // 定义选修课程类，继承自课程类
      3个用法
106     class ElectiveCourse extends Course {
      2个用法
107         public ElectiveCourse(String id, String name, double credit) {
108             super(id, name, credit);
109         }
110     }

```

2. 把选课方法由原来写在 main 方法集成到类中

```

35      // 手动选课方法，从选修课程中选择2门课程添加到课程列表中
      4个用法
36      @ public void manualSelectCourse(Course[] courses) {
37          int count = 0;
38          for (Course course : courses) {
39              if (course instanceof ElectiveCourse && count < 2) {
40                  selectCourse(course);
41                  count++;
42              }
43          }
44      }

```

3. Override 改写方法

```

90      // toString方法，用于打印课程信息
91      @Override
92      public String toString() {
93          return id + " " + name + " " + credit;
94      }

```

4. 修改自动选课方法，可以利用 instanceof 判断必修课程类

```

26      // 自动选课方法，将所有必修课程添加到课程列表中
      4 个用法
27  @   public void autoSelectCourse(Course[] courses) {
28      for (Course course : courses) {
29          if (course instanceof RequiredCourse) {
30              selectCourse(course);
31          }
32      }
33  }

```

关于 Rational 有理数类的修改思路：

由于之前根据实验的图片材料，已经写的比较完善，主要修改在于改变 private 和 public 属性，使得更人性化和安全。

```

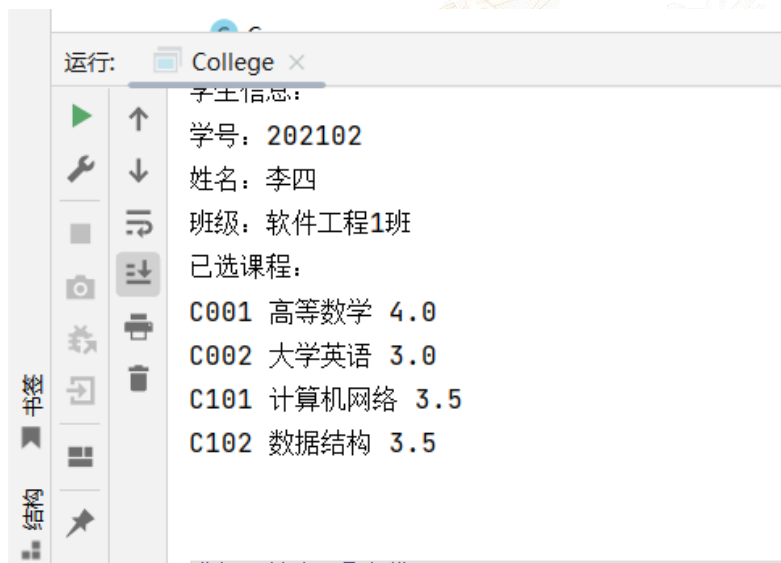
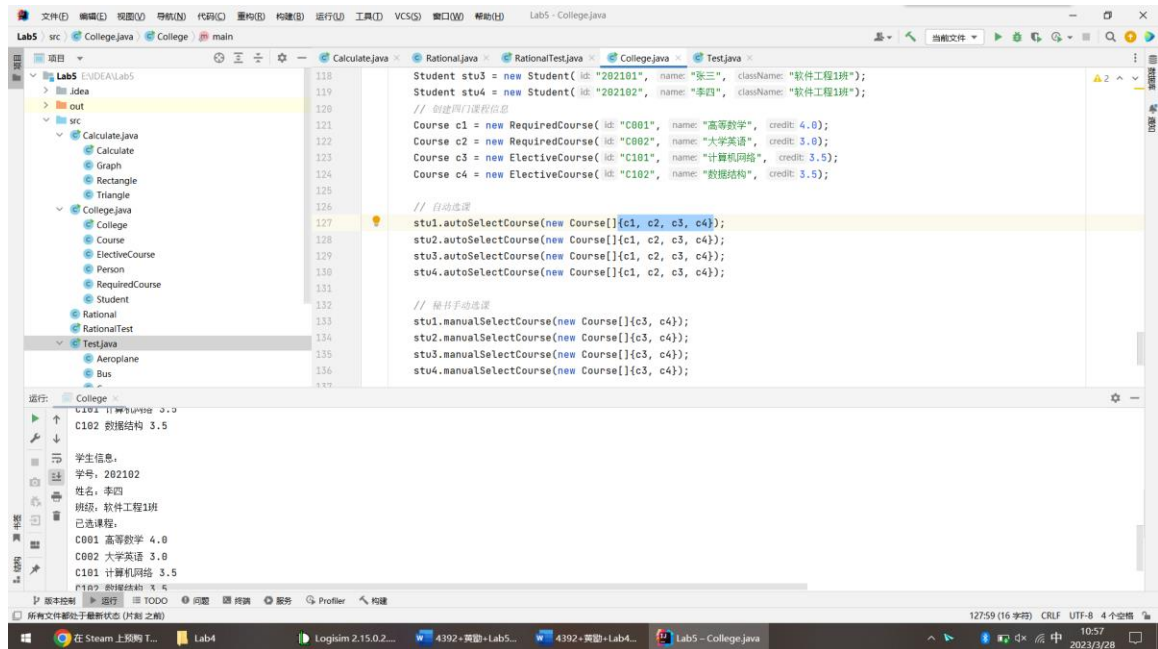
7      private long numerator; // 分子
      8 个用法
8      private long denominator; // 分母
9
      11 个用法
10     public long getNumerator() { return numerator; }
14
      0 个用法
15     public void setNumerator(long numerator) { this.numerator = numerator; }

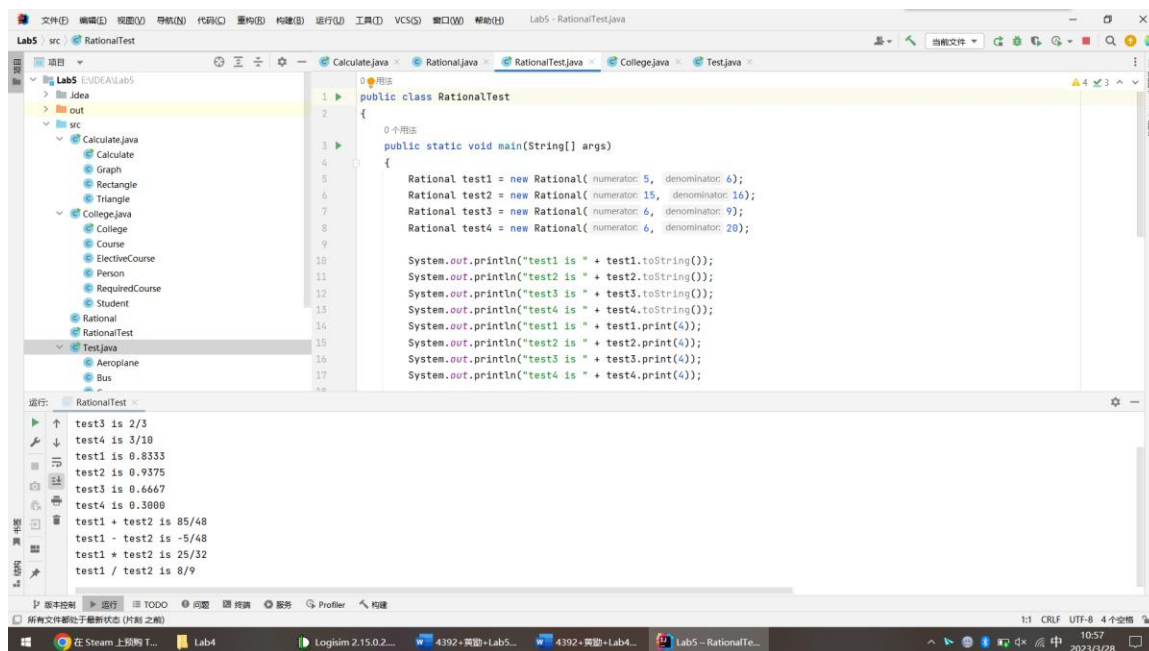
```

(三) 过程截图

最终结果







```
1 public class RationalTest
2 {
3     0个用法
4     public static void main(String[] args)
5     {
6         Rational test1 = new Rational( numerator: 5, denominator: 6);
7         Rational test2 = new Rational( numerator: 15, denominator: 16);
8         Rational test3 = new Rational( numerator: 6, denominator: 9);
9         Rational test4 = new Rational( numerator: 6, denominator: 20);
10
11         System.out.println("test1 is " + test1.toString());
12         System.out.println("test2 is " + test2.toString());
13         System.out.println("test3 is " + test3.toString());
14         System.out.println("test4 is " + test4.toString());
15         System.out.println("test1 is " + test1.print());
16         System.out.println("test2 is " + test2.print());
17         System.out.println("test3 is " + test3.print());
18         System.out.println("test4 is " + test4.print());
19     }
20 }
```

运行: RationalTest

```
test1 is 5/6
test2 is 15/16
test3 is 2/3
test4 is 3/10
test1 is 0.8333
test2 is 0.9375
test3 is 0.6667
test4 is 0.3000
test1 + test2 is 85/48
test1 - test2 is -5/48
test1 * test2 is 25/32
test1 / test2 is 8/9
```

题目 2:

(一) 实验题目

- (1) 编写一个交通工具类 Vehicle 类, 创建一个 run 方法, 从控制台中输出“这是交通工具 run 方法”。
- (2) 创建 Vehicle 类的三个子类, Motor 类表示汽车, Ship 类表示船, Aeroplane 类表示飞机类, 分别写出他们的 run 方法;
- (3)、创建 Motor 的二个子类, Bus 和 Car, 分别表示公共汽车和轿车, 分别写出各自的 run 方法。
- (4)、创建一个测试类 Test, 分别创建上面的各种类, 调用相应的 run 方法。

(二) 实现过程 (Test.java)

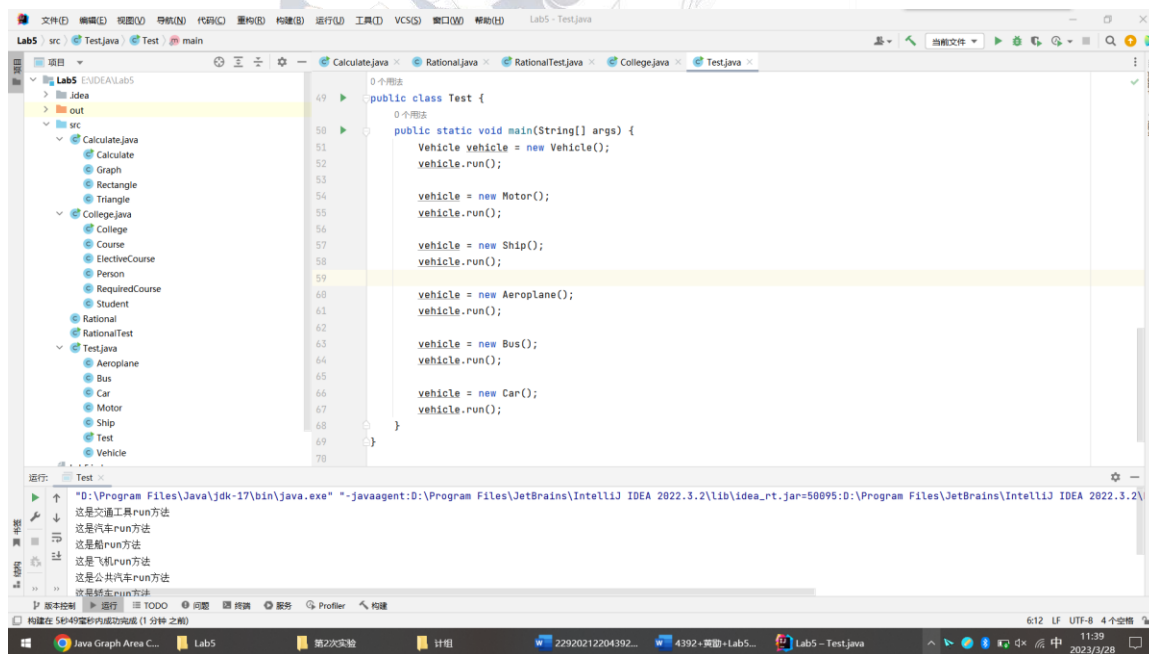
具体方法: 这个题目要求设计包含了一个交通工具类 Vehicle 和几个子类, 每个子类都继承了交通工具类 Vehicle 的 run 方法并需要对其进行了重写。具体来说, Motor 类、Ship 类和 Aeroplane 类继承了 Vehicle 类, 而 Bus 类和 Car 类则继承了 Motor 类。每个子类中的 run 方法输出了该交通工具类型的名称。

在测试类 Test 中，需要创建一个 Vehicle 对象、一个 Motor 对象、一个 Ship 对象、一个 Aeroplane 对象、一个 Bus 对象和一个 Car 对象，然后对每个对象调用它们各自的 run 方法。这样可以测试每个子类的 run 方法是否正常输出了预期的信息。

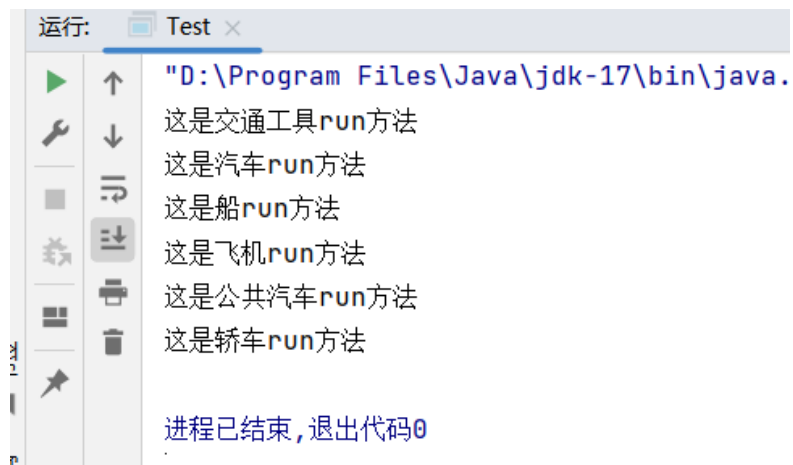
具体设计如下：

- 类 Vehicle：表示交通工具，包含一个 run 方法。
- 类 Motor、类 Ship 和类 Aeroplane：分别表示汽车、船和飞机，都继承了 Vehicle 类并重写了其 run 方法。
- 类 Bus 和类 Car：分别表示公共汽车和轿车，都继承了 Motor 类并重写了其 run 方法。
- 测试类 Test：包含一个 main 方法，在其中创建了几个交通工具对象并调用了它们各自的 run 方法。

(三) 过程截图



具体信息：

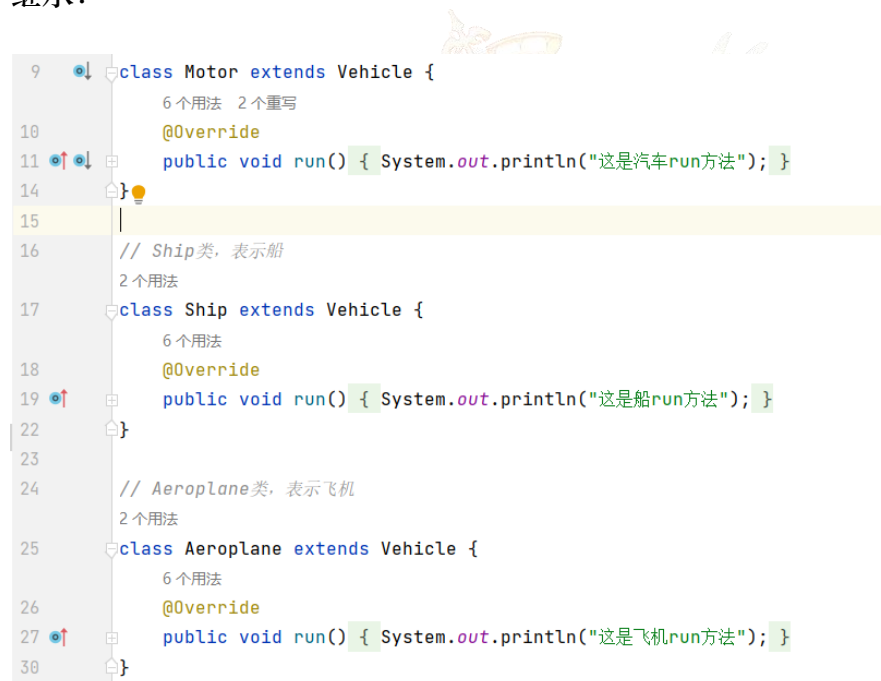


```
运行: Test x
"D:\Program Files\Java\jdk-17\bin\java.
这是交通工具run方法
这是汽车run方法
这是船run方法
这是飞机run方法
这是公共汽车run方法
这是轿车run方法

进程已结束,退出代码0
```

代码展示:

继承:



```
9 class Motor extends Vehicle {
    6 个用法 2 个重写
10     @Override
11     public void run() { System.out.println("这是汽车run方法"); }
14
15
16 // Ship类, 表示船
    2 个用法
17 class Ship extends Vehicle {
    6 个用法
18     @Override
19     public void run() { System.out.println("这是船run方法"); }
22
23
24 // Aeroplane类, 表示飞机
    2 个用法
25 class Aeroplane extends Vehicle {
    6 个用法
26     @Override
27     public void run() { System.out.println("这是飞机run方法"); }
30 }
```

生成实例、多态:


```
49  ▶ ▶ public class Test {  
    0 个用法  
50  ▶ ▶ public static void main(String[] args) {  
51      Vehicle vehicle = new Vehicle();  
52      vehicle.run();  
53  
54      vehicle = new Motor();  
55      vehicle.run();  
56  
57      vehicle = new Ship();  
58      vehicle.run();  
59  
60      vehicle = new Aeroplane();  
61      vehicle.run();  
62  
63      vehicle = new Bus();  
64      vehicle.run();  
65  
66      vehicle = new Car();  
67      vehicle.run();  
68  }  
69  }
```

题目 3:

(一) 实验题目

请你实现一个基础图形类 Graph，然后实现三角形类 Triangle 和矩形类 Rectangle，继承自 Graph。根据输入的边数实现不同的对象，并计算面积。

输入格式：

一行，一个整数 n，表示图形个数。

n 行，每行是用空格隔开的整数。

输出格式：

n 行，每行是一个图形的面积。

输入样例：

2

5 5

6 6 6

输出样例：

25

15

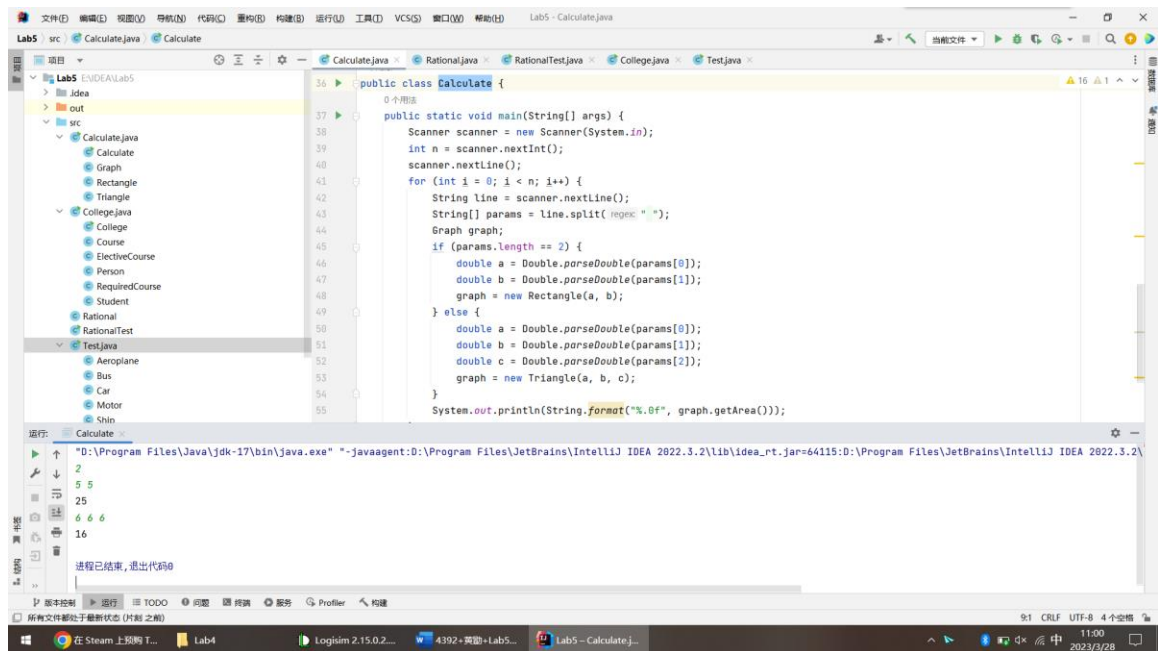
（二）实现过程 (Calculate.java)

思路：需要设计一个简单的图形面积计算程序，包含三个类：Graph、Triangle 和 Rectangle。Graph 是所有图形的基类，其中只有一个保护属性 area（面积）和一个公共方法 getArea（获取面积）。Triangle 和 Rectangle 是派生类，分别表示三角形和矩形。它们都继承了 Graph 类，拥有各自的构造方法和计算面积的方式。在主函数 Calculate 中，通过输入参数和用户输入的信息创建相应的图形对象，并调用该对象的 getArea 方法计算并输出该图形的面积。

具体而言，主函数从用户输入中读取整数 n，表示接下来将输入 n 个图形的信息。对于每个图形的信息，用户需要输入一行字符串。这行字符串可以是两个参数（表示一个矩形的宽和高）或三个参数（表示一个三角形的三边长）。使用 Scanner 类解析用户的输入，然后根据参数的数量创建相应的图形对象。最后，输出该图形的面积，格式化为整数。

（三）过程截图

最终结果



Graph 类及其继承、方法重写:

```
4 class Graph {  
    3 个用法  
5     protected double area;  
6  
    1 个用法  
7     public double getArea() { return area; }  
10 }  
11  
12 // 三角形类  
    1 个用法  
13 class Triangle extends Graph {  
    1 个用法  
14     private double a, b, c;  
15  
    1 个用法  
16     public Triangle(double a, double b, double c) {  
17         this.a = a;  
18         this.b = b;  
19         this.c = c;  
20         double p = (a + b + c) / 2;  
21         area = Math.sqrt(p * (p - a) * (p - b) * (p - c));  
22     }  
23 }
```

三、实验总结与心得记录

Java 中的继承和多态是面向对象编程的重要概念,对于 Java 程序员来说是必须掌握的基础知识。继承是一种创建新类的方式,让已有类的属性和方法在新的类中得以重复使用。通过继承,可以实现代码的复用性和可维护性。Java 中的继承是单继承的,即每个类只能直接继承自一个父类。

在继承的基础上,Java 还提供了多态机制。多态是指同一种行为(方法)在不同的对象上有不同的表现形式。在 Java 中,多态可以通过方法重载和方法重写来实现。方法重载是指在同一个类中定义多个方法,方法名相同但参数不同,编译器会根据不同的参数类型和个数选择合适的方法。方法重写是指子类重写父类中的方法,使得子类可以使用自己的实现方式。

继承和多态是 Java 面向对象编程的内核概念,理解和掌握这些概念对于 Java 程序员来说非常重要。在实际编程中,应该尽可能地使用继承和多态,以提高代码

的复用性和可维护性。总之，这个实验对我的 Java 编程技能和面向对象编程能力的提升非常有帮助。通过这个实验，我深入了解了 Java 的更多知识。这些技能和知识将在我的未来的 Java 开发中起到重要的作用。

