

# 廈門大學



## 信息学院软件工程系

### 《JAVA 程序设计》实验报告

#### 实验 4

姓名：黄勛

学号：22920212204392

学院：信息学院

专业：软件工程

完成时间：2023.3.21

## 一、实验目的及要求

- 熟悉枚举
- 熟悉继承

## 二、实验题目及实现过程

实验环境：Windows 10 21H2、jdk17、javafx+scene builder

### （基本题目）题目 1：

#### （一）实验题目

设计程序，具备以下功能（要求用到继承）：

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

#### （二）实现过程（college.java）

思路：主要需要设计的是一个选课系统的实现，学生（包括本科生和研究生）可以选择必修课程和选修课程。

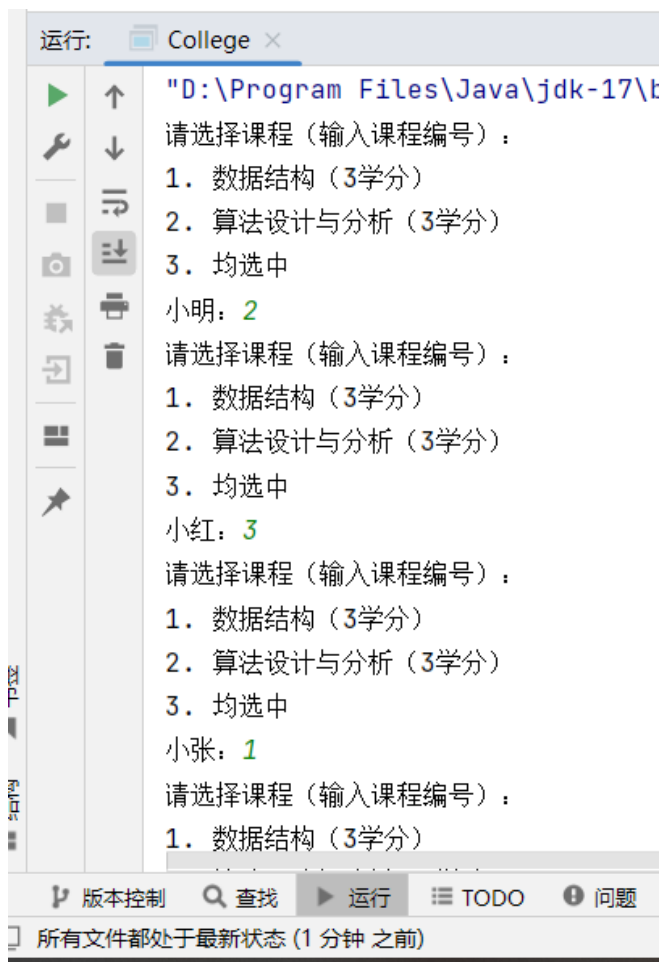
具体的实现过程如下：

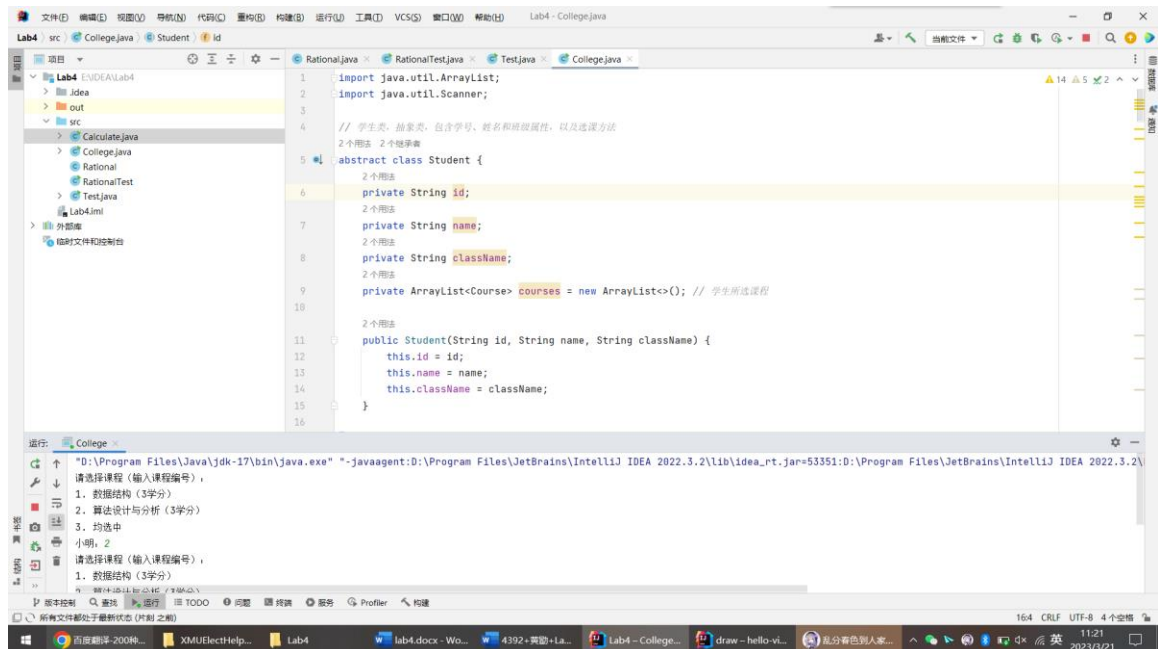
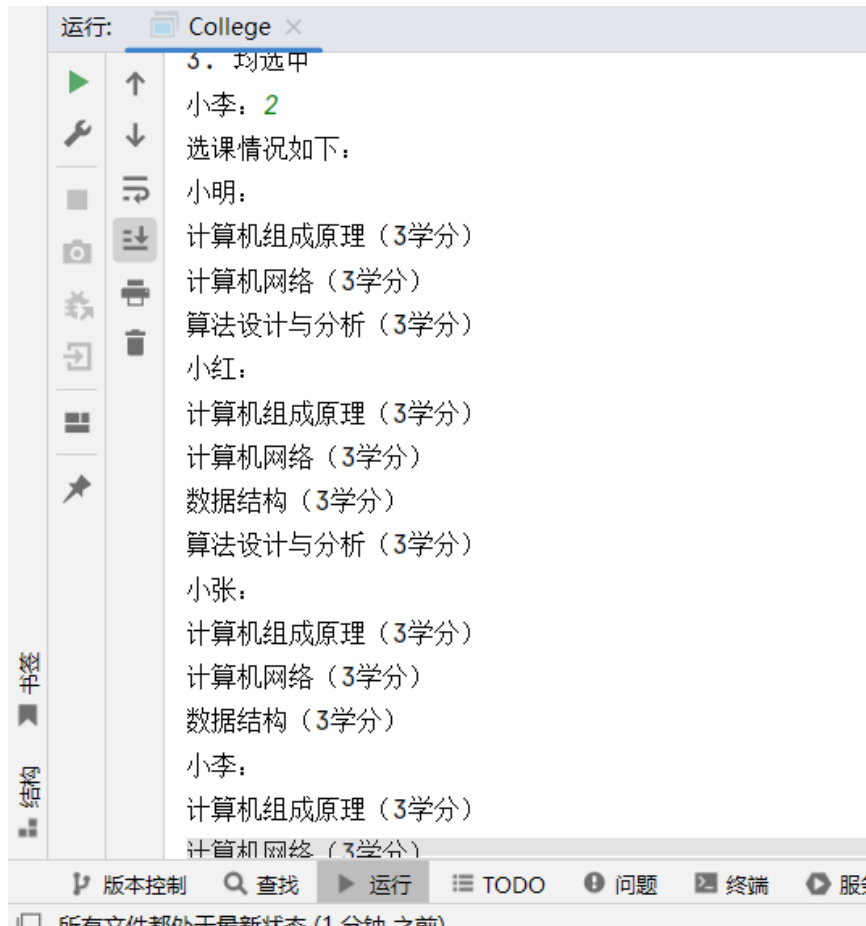
- 定义学生类（抽象类），包含学号、姓名和班级属性，以及选课方法。该类中包含一个 ArrayList 用于保存学生所选课程。
- 定义本科生类和研究生类，均继承自学生类。本科生类和研究生类分别增加一个构造函数。

- 定义课程类，包含编号、名称和学分属性。
- 在主函数中，创建必修课程和选修课程。创建学生信息，包括本科生和研究生。为每个学生自动选修所有必修课。
- 实现秘书手动选课部分：为每个学生手动选修 1-2 门选修课。使用 Scanner 类获取用户的输入，根据输入选择课程并调用学生的 selectCourse 方法。

### (三) 过程截图

#### 最终结果





具体代码

## 学生类：

```
4 // 学生类，抽象类，包含学号、姓名和班级属性，以及选课方法
   2 个用法 2 个继承者
5 abstract class Student {
   2 个用法
6     private String id;
   2 个用法
7     private String name;
   2 个用法
8     private String className;
   2 个用法
9     private ArrayList<Course> courses = new ArrayList<>(); // 学生所选课程
10
   2 个用法
11     public Student(String id, String name, String className) {
12         this.id = id;
13         this.name = name;
14         this.className = className;
15     }
16
   24 个用法
17     public void selectCourse(Course course) { courses.add(course); }
20
   0 个用法
21     public String getId() { return id; }
```

## 继承的本科生类以及研究生类：

```
38 // 本科生类，继承自学生类，增加一个构造函数
    4个用法
39 class Undergraduate extends Student {
40
    2个用法
41     public Undergraduate(String id, String name, String className) {
42         super(id, name, className);
43     }
44 }
45
46 }
47
48 // 研究生类，继承自学生类，增加一个构造函数，以及导师属性
    4个用法
49 class Graduate extends Student {
    2个用法
50     private String tutor;
51
    2个用法
52     public Graduate(String id, String name, String className, String tutor) {
53         super(id, name, className);
54         this.tutor = tutor;
55     }
}
```

课程类：

```
62 // 课程类，包含编号、名称和学分属性
    16个用法
63 class Course {
    2个用法
64     private String id;
    2个用法
65     private String name;
    2个用法
66     private int credit;
    2个用法
67     private boolean iscompulsory;
68
    4个用法
69     public Course(String id, String name, int credit, boolean comp) {
70         this.id = id;
71         this.name = name;
72         this.credit = credit;
73         this.iscompulsory = comp;
74     }
}
```

## Main:

```
94 public static void main(String[] args) {
95     // 创建必修课程
96     Course c1 = new Course( id: "0001", name: "计算机组成原理", credit: 3, comp: true);
97     Course c2 = new Course( id: "0002", name: "计算机网络", credit: 3, comp: true);
98
99     // 创建选修课程
100    Course c3 = new Course( id: "0003", name: "数据结构", credit: 3, comp: false);
101    Course c4 = new Course( id: "0004", name: "算法设计与分析", credit: 3, comp: false);
102
103    // 创建学生信息
104    Undergraduate u1 = new Undergraduate( id: "001", name: "小明", className: "计算机科学与技术");
105    Undergraduate u2 = new Undergraduate( id: "002", name: "小红", className: "软件工程");
106    Graduate g1 = new Graduate( id: "003", name: "小张", className: "计算机应用", tutor: "赵老师");
107    Graduate g2 = new Graduate( id: "004", name: "小李", className: "计算机科学与技术", tutor: "刘老师");
108
109    // 自动选课部分: 为每个学生自动选修所有必修课
110    u1.selectCourse(c1);
111    u1.selectCourse(c2);
112    u2.selectCourse(c1);
113    u2.selectCourse(c2);
114    g1.selectCourse(c1);
115    g1.selectCourse(c2);
116    g2.selectCourse(c1);
117    g2.selectCourse(c2);
```

## 教秘选课:

```
119 // 秘书手动选课部分: 为每个同学选修1-2门选修课
120 Scanner scanner = new Scanner(System.in);
121
122 System.out.println("请选择课程(输入课程编号): ");
123 System.out.println("1. " + c3.getName() + " (" + c3.getCredit() + "学分)");
124 System.out.println("2. " + c4.getName() + " (" + c4.getCredit() + "学分)");
125 System.out.println("3. 均选中");
126 System.out.print(u1.getName() + ": ");
127 int choice = scanner.nextInt();
128 if (choice == 1) {
129     u1.selectCourse(c3);
130 } else if (choice == 2) {
131     u1.selectCourse(c4);
132 } else if (choice == 3) {
133     u1.selectCourse(c3);
134     u1.selectCourse(c4);
135 }
```

## 遍历输出课程:

```

System.out.println(g2.getName() + ": ");
courses = g2.getCourses();
for (Course course : courses) {
    System.out.println(course.getName() + " (" + course.getCredit() + "学分)");
}

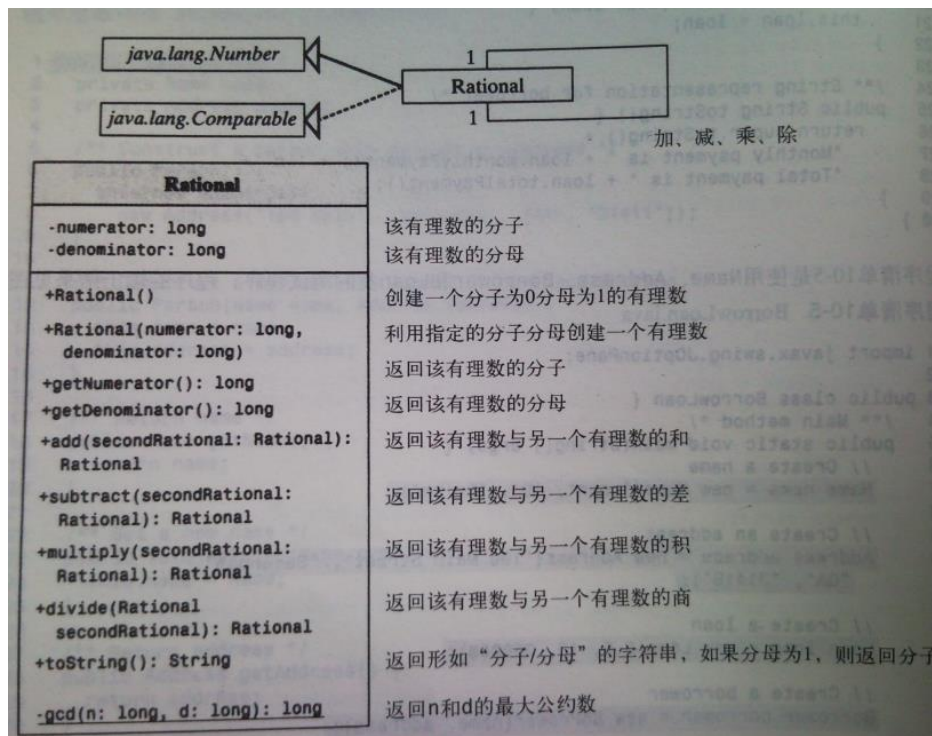
```

## 题目 2:

### (一) 实验题目

写一个有理数类，相关属性和方法要求如下图。

写一个测试类，创建两个有理数对象，输出两个有理数的加、减、乘、除结果。



### (二) 实现过程 (Ration/RationTest.java)

思路： 定义一个名为 **Rational** 的类，它表示有理数，并实现了分数加减乘除以及分数的大小比较。

类的声明：声明 **Rational** 类，实现 **Comparable** 接口表示有理数，可以进行分数的运算，并能够比较分数的大小。

类的属性：



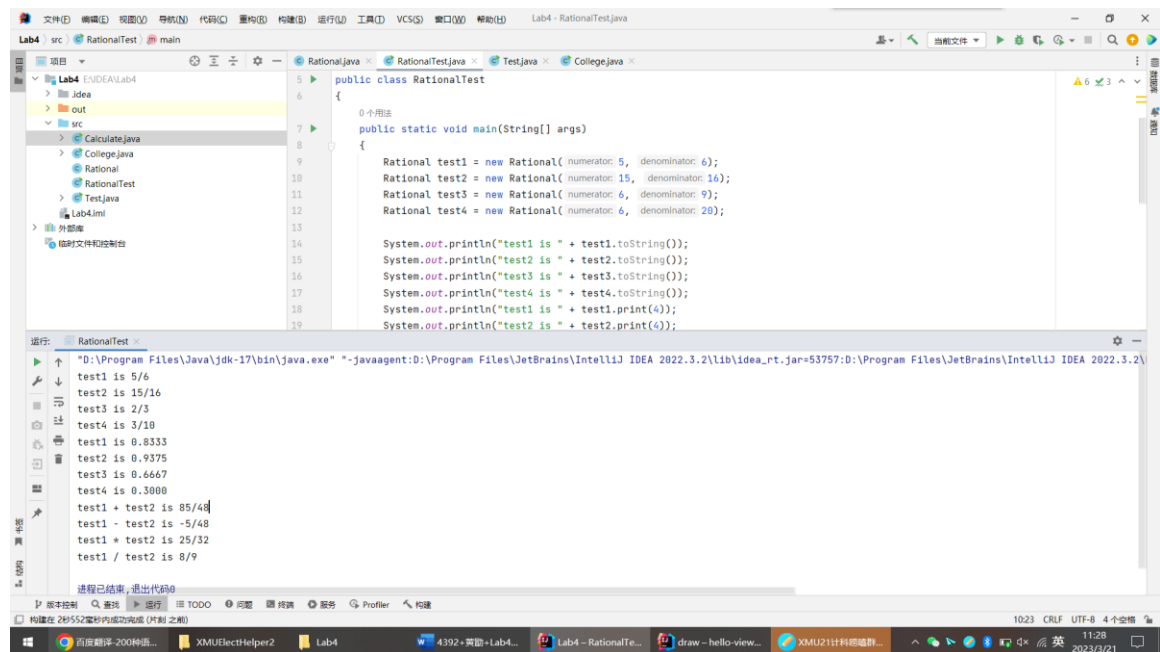
```
private long numerator; // 分子
```

```
private long denominator; // 分母
```

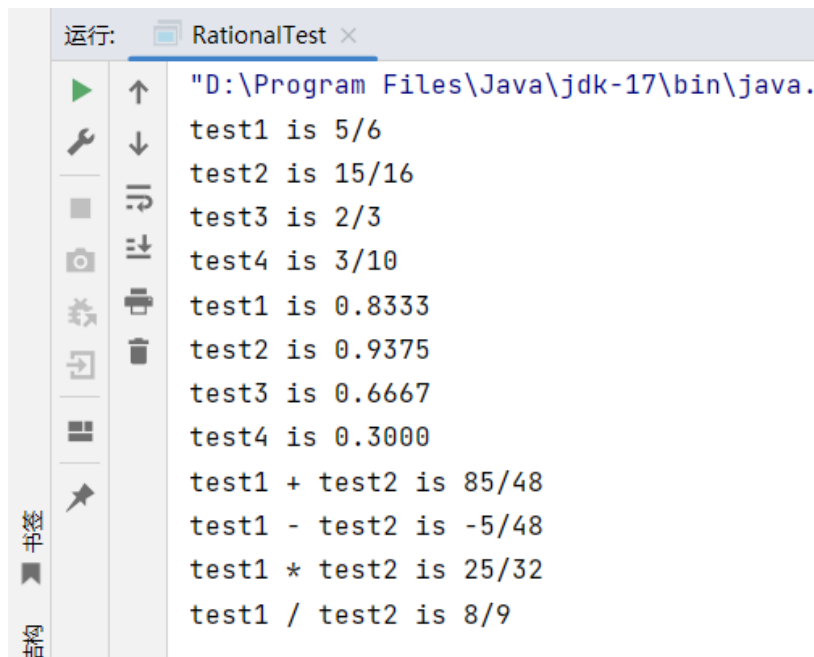
该类具有两个私有属性 `numerator` 和 `denominator`，分别表示有理数的分子和分母。

具体方法：实现四则运算、求 gcd 以及 lcm

### (三) 过程截图



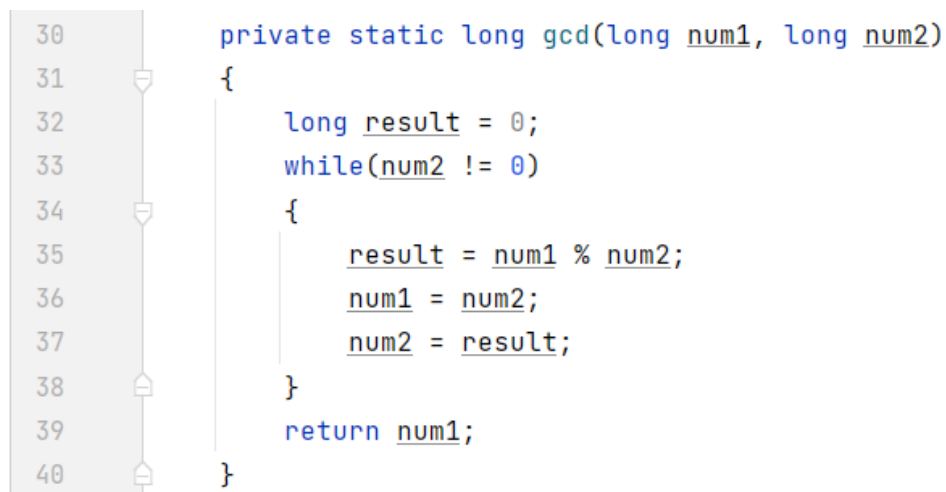
具体运算：



```
运行: RationalTest x
"D:\Program Files\Java\jdk-17\bin\java.
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
```

代码展示:

求 gcd:



```
30 private static long gcd(long num1, long num2)
31 {
32     long result = 0;
33     while(num2 != 0)
34     {
35         result = num1 % num2;
36         num1 = num2;
37         num2 = result;
38     }
39     return num1;
40 }
```

四则运算（利用 this 实现）：

```

54 @ 1个用法
55 public Rational add(Rational secondRational) {
56     long lcd = (Math.abs(this.denominator * secondRational.getDenominator())) / gcd(this.denominator, secondRational.getDenominator());
57
58     long num1 = (lcd / this.denominator) * this.numerator;
59     long num2 = (lcd / secondRational.getDenominator()) * secondRational.getNumerator();
60
61     return new Rational( numerator: num1 + num2, lcd);
62 }
63
64 @ 1个用法
65 public Rational sub(Rational secondRational)
66 {
67     long lcd = (Math.abs(this.getDenominator() * secondRational.getDenominator())) / gcd(this.getDenominator(), secondRational.getDenominator());
68
69     long num1 = (lcd / this.getDenominator()) * this.getNumerator();
70     long num2 = (lcd / secondRational.getDenominator()) * secondRational.getNumerator();
71
72     return new Rational( numerator: num1 - num2, lcd);
73 }
74
75 @
76 public Rational multiply(Rational secondRational)
77 {
78     return new Rational( numerator: this.getNumerator() * secondRational.getNumerator(), denominator: this.getDenominator() * secondRational.getDenominator());
79 }
80
81 @ 1个用法
82 public Rational divide(Rational secondRational)
83 {
84     return new Rational( numerator: this.getNumerator() * secondRational.getDenominator(), denominator: this.getDenominator() * secondRational.getNumerator());
85 }
86
87 public String toString() { return String.format("%d/%d", numerator, denominator); }

```

重写 compare 比较:

```

94 94 public int compareTo(Object o)
95 {
96     Rational num = (Rational)o;
97     if((double)this.getNumerator()/this.getDenominator() > (double)num.getNumerator() / num.getDenominator())
98         return 1;
99     else if((double)this.getNumerator()/this.getDenominator() == (double)num.getNumerator() / num.getDenominator())
100         return 0;
101     else
102         return -1;
103 }

```

### 题目 3:

#### (一) 实验题目

创建一个简单的绘图应用程序，要求如下：

1) 随机产生一个随机数 (0,1,2)，三个随机数分别对应直线、矩形和椭圆三种图形。

根据随机数对应图形，提示用户输入图形所需初始化参数，提示信息应包括参数的范围，用户输入后进行范围检查，若合法，则根据用户输入的信息在界面上绘制出相应的图形。

2) 绘制 20 个图形后，不再创建新的图形。

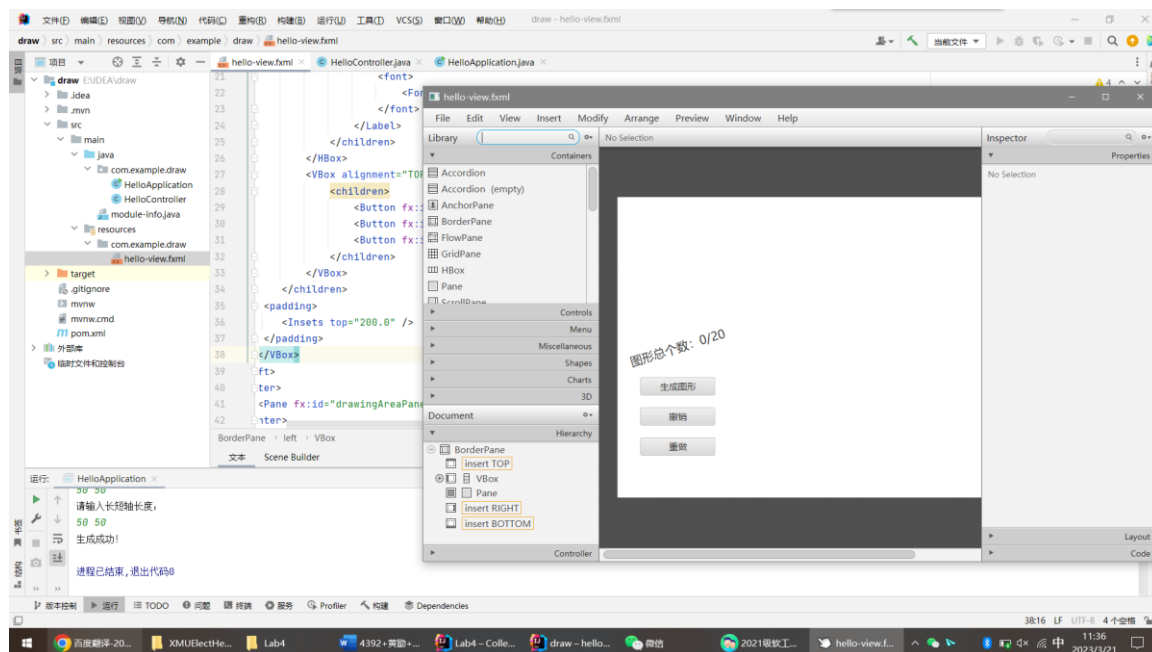
## (二) 实现过程 (draw 工程项目)

思路:

- 创建 javafx 项目，配置 fxml、application 和 controller 文件
- 在 application 中初始化窗口并载入 fxml

```
1 个用法
9 ▶ public class HelloApplication extends Application {
    0 个用法
10 ▶ public static void main(String [] args) { launch(args); }
13
14
15 @Override
16 public void start(Stage stage) throws Exception {
17     Parent root = FXMLLoader.load(getClass().getResource("hello-view.fxml"));
18     Scene scene = new Scene(root);
19     stage.setTitle("绘图小程序");
20     stage.setScene(scene);
21     stage.show();
22 }
```

- 利用 scene builder 绘制窗口并保存





```
14      <children>
15          <Label text="图形总个数: ">
16              <font>
17                  <Font size="20.0" />
18              </font>
19          </Label>
20          <Label fx:id="countLabel" text="0/20">
21              <font>
22                  <Font size="20.0" />
23              </font>
24          </Label>
25      </children>
26  </HBox>
27  <VBox alignment="TOP_CENTER" prefHeight="200.0" prefWidth="150.0" spacing="20.0">
28      <children>
29          <Button fx:id="drawButton" mnemonicParsing="false" onAction="#drawClicked" prefHeight="30.0" prefWidth="125.0" text="生成图形" />
30          <Button fx:id="undoButton" mnemonicParsing="false" onAction="#undoClicked" prefHeight="30.0" prefWidth="125.0" text="撤销" />
31          <Button fx:id="clearButton" mnemonicParsing="false" onAction="#clearClicked" prefHeight="30.0" prefWidth="125.0" text="重做" />
32      </children>
33  </VBox>
34  </children>
```

- 编写 controller，包括三个按钮点击事件，生成随机数并让用户输入参数  
生成图形
- drawClicked 事件

```
26 private void drawClicked(ActionEvent event) {
27     if (count >= 20) {
28         drawButton.setDisable(true);
29         return;
30     }
31     Integer choice = random.nextInt( bound: 3);
32     switch (choice) {
33         case 0 -> {
34             System.out.println("绘画直线: ");
35             System.out.println("请输入开始点: ");
36             Double startX = 0.0;
37             Double startY = 0.0;
38             startX = scanner.nextDouble();
39             startY = scanner.nextDouble();
40             System.out.println("请输入结束点: ");
41             Double endX = 0.0;
42             Double endY = 0.0;
43             endX = scanner.nextDouble();
44             endY = scanner.nextDouble();
45             System.out.println("生成成功! ");
46             Line line = new Line();
47             line.setStartX(startX);
48             line.setStartY(startY);
49             line.setEndX(endX);
```

● undoClicked 和 clearClicked 事件

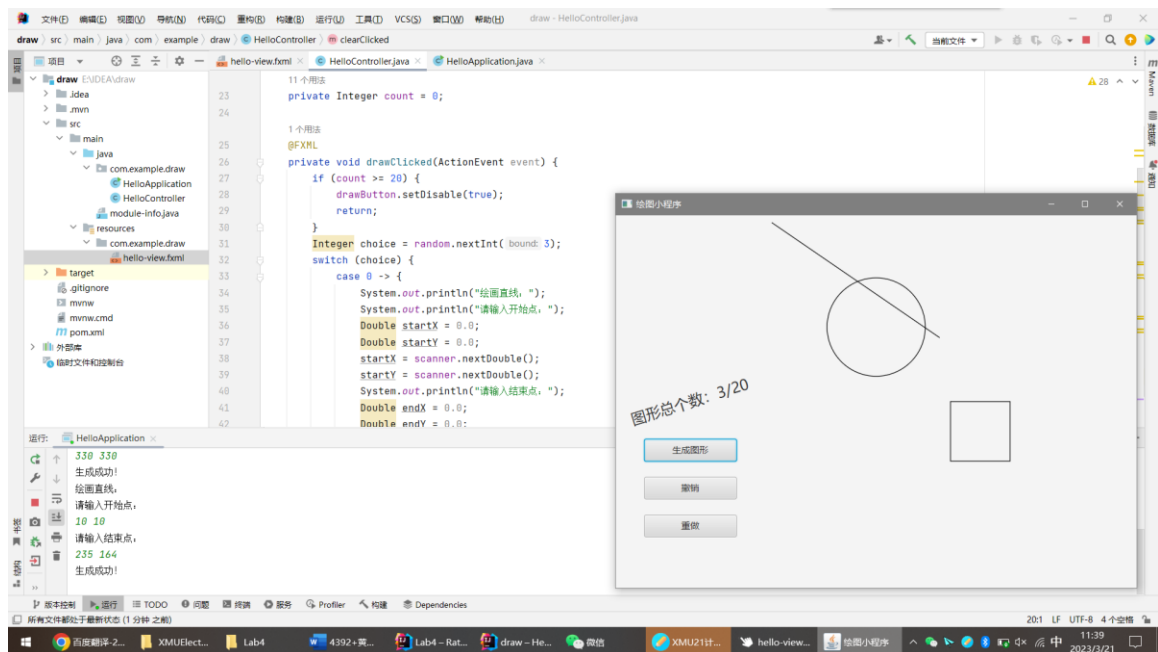
```
99 private void undoClicked() {
100     if (count > 0) {
101         drawingAreaPane.getChildren().remove( index: count - 1);
102         count--;
103         countLabel.setText(count + "/" + 20);
104     }
105 }
106
107 1 个用法
108 @FXML
109 private void clearClicked() {
110     drawingAreaPane.getChildren().clear();
111     count = 0;
112     countLabel.setText(count + "/" + 20);
113 }
```

超过 20 个退出：

```
if (count >= 20) {
    drawButton.setDisable(true);
    return;
}
```

### (三) 过程截图

#### 最终结果



### 题目 4:

#### (一) 实验题目

写一个交通信号灯枚举类 TrafficLight，并在 Test 类中输出每种灯颜色的 RGB 值。

#### (二) 实现过程 (Test.java)

- 创建一个枚举类 TrafficLight，用于表示交通信号灯。
- 在 TrafficLight 中定义三个枚举常量 RED、YELLOW 和 GREEN，它们分别对应红灯、黄灯和绿灯，并用构造方法为每个常量设置 RGB 颜色值。

```
2 enum TrafficLight {  
    3     3 个用法  
    3     RED( r: 255, g: 0, b: 0), // 红灯  
    4     3 个用法  
    4     YELLOW( r: 255, g: 255, b: 0), // 黄灯  
    5     3 个用法  
    5     GREEN( r: 0, g: 255, b: 0); // 绿灯  
    6
```

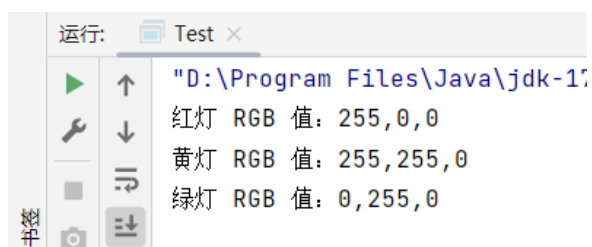
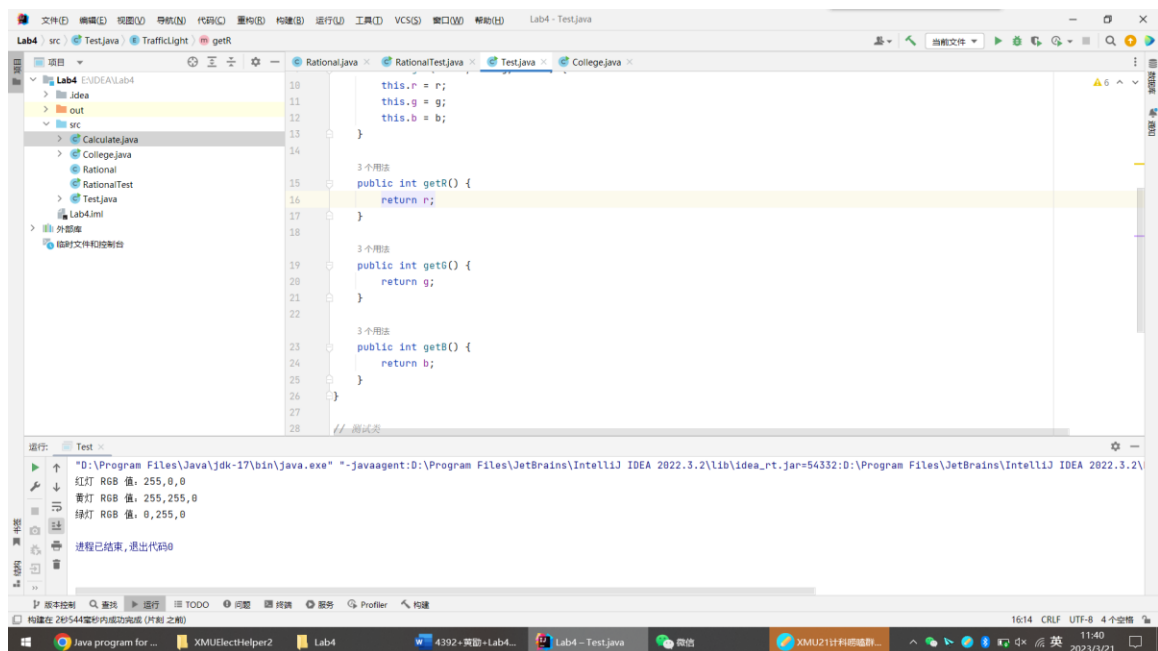
- 在 TrafficLight 中定义三个方法 getR()、getG() 和 getB(), 用于获取每个枚举常量的 RGB 颜色值。
- 创建一个测试类 Test, 在其中打印输出每个枚举常量的 RGB 值。

```
19 3 个用法  
20 public int getG() {  
21     return g;  
22 }  
23 3 个用法  
24 public int getB() {  
25     return b;  
26 }  
27  
28 // 测试类  
29 0 个用法  
30 public class Test {  
31     0 个用法  
32     public static void main(String[] args) {  
33         System.out.println("红灯 RGB 值: " + TrafficLight.RED.getR() + "," + TrafficLight.RED.getG() + "," + TrafficLight.RED.getB());  
34         System.out.println("黄灯 RGB 值: " + TrafficLight.YELLOW.getR() + "," + TrafficLight.YELLOW.getG() + "," + TrafficLight.YELLOW.getB());  
35         System.out.println("绿灯 RGB 值: " + TrafficLight.GREEN.getR() + "," + TrafficLight.GREEN.getG() + "," + TrafficLight.GREEN.getB());  
36     }  
37 }
```

- 运行测试类, 查看输出结果。

### (三) 过程截图





### 三、实验总结与心得记录

在这个实验中，我学习了如何在 Java 中使用枚举类型来定义常量集合，并且如何在程序中使用这些常量。通过定义交通信号灯枚举类型，我了解到如何使用枚举类型来组织和管理相关的常量，并且可以轻松地访问和使用它们。

在继续实验的过程中，我还学习了如何使用 scene builder 来创建窗口，并且使用 JavaFX 库来在图形用户界面中绘制图形。通过这个实验，我掌握了如何使用 JavaFX 中提供的各种工具，以及如何使用面向对象的编程方式来组织和管理自定义图形对象。

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。