厦門大學

信息学院软件工程系

《JAVA 程序设计》实验报告

实验四

姓名：任宇

学号：33920212204567

学院：信息学院

专业：软件工程

完成时间：2024-03-23

一、 实验目的及要求

- 开始熟悉图形界面
- 熟悉枚举
- 熟悉继承

二、 实验题目及实现过程

1. 题目一：

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。

（一）实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK 版本：JDK17

（二）实现过程

为了实现题目要求，我设计了 Student（学生），Undergraduate（本科生），Postgraduate（研究生），Course（课程），RequiredCourse（必修课），和 ElectiveCourse（选修课）这几个类。此外，我还设计了一个

CourseSelectionSystem 类来模拟选课系统，它包括自动选课和教秘手动选课的方法。具体设计如下：

1) Course（课程基类）

用途：表示一个通用的课程，提供课程的基本信息。

主要成员变量：

- courseId（String）：课程的唯一标识符。
- courseName（String）：课程名称。
- credits（int）：课程学分。
- courseType（CourseType，枚举）：课程类型（必修或选修）。

主要方法：

- 构造方法：初始化课程信息。
- toString()：返回课程的字符串表示，用于打印课程信息。

2) RequiredCourse 和 ElectiveCourse（Course 的子类）

用途：分别表示必修课和选修课，继承自 Course 类。

主要成员变量：继承自 Course 类，没有额外成员。

主要方法：使用基类的构造方法，并在构造时指定 courseType 为相应的枚举值。

3) Student（学生基类）

用途：表示一个通用的学生，提供学生的基本信息和选课功能。

主要成员变量：

- studentId（String）：学生的唯一标识符。
- name（String）：学生姓名。
- className（String）：学生所在班级。

- courses（List<Course>）：学生所选的课程列表。

主要方法：

- 构造方法：初始化学生信息。
- addCourse(Course course)：向学生的课程列表中添加一个课程。
- printCourses()：打印学生的选课信息。

4) Undergraduate 和 Postgraduate（Student 的子类）

用途：分别表示本科生和研究生，继承自 Student 类。

主要成员变量：Postgraduate 类包含额外的 supervisor（String）成员，表示研究生的导师。

主要方法：使用基类的方法，Postgraduate 类的构造方法还需初始化导师。

5) CourseSelectionSystem（模拟选课系统）

用途：管理学生、课程及选课流程的系统。

主要成员变量：

- students（List<Student>）：系统中的学生列表。
- courses（List<Course>）：系统中的课程列表。

主要方法：

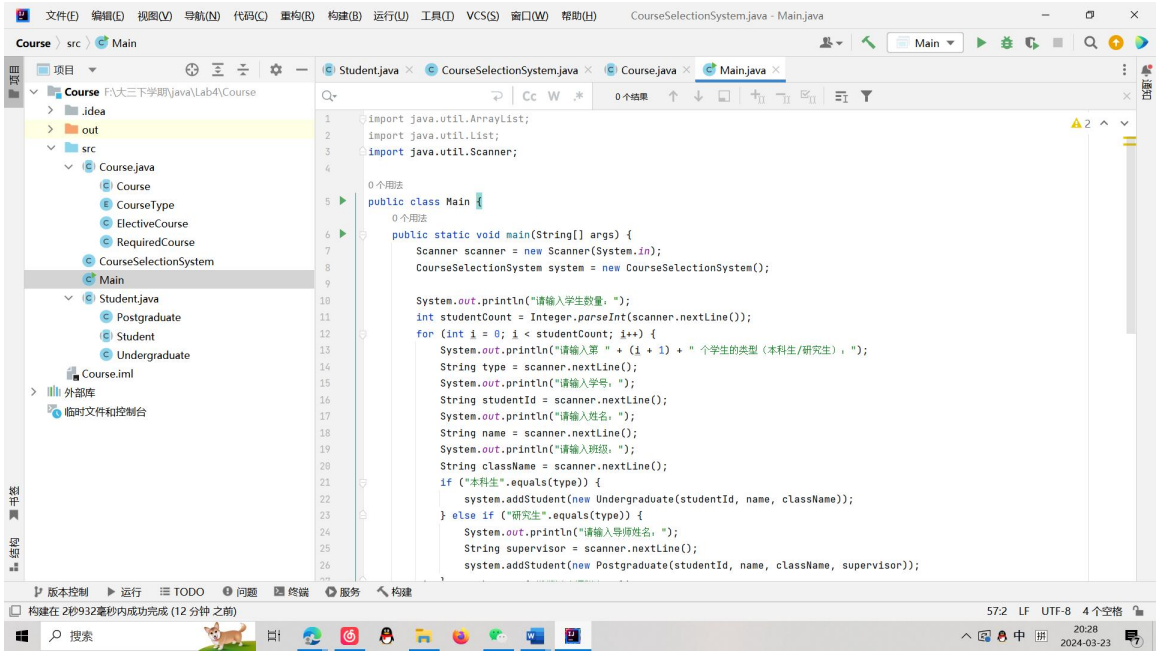
- addStudent(Student student)：向系统中添加一个学生。
- addCourse(Course course)：向系统中添加一个课程。
- autoSelectRequiredCourses()：为所有学生自动选修所有必修课。
- SelectElectiveCourses()：教秘为学生手动选修 1-2 门选修课。
- printAllStudentCourses()：打印所有学生的选课信息。

6) Main（主类）

调用关系：CourseSelectionSystem 是整个程序的核心，负责创建和管理 Student 对象和 Course 对象的集合。当程序运行时，首先通过 CourseSelectionSystem 创建并添加课程（RequiredCourse 和 ElectiveCourse）和学生（Undergraduate 和 Postgraduate）。接着自动选课，autoSelectRequiredCourses 方法确保所有学生自动选修所有必修课。该过程遍历所有学生和课程，自动为学生添加必修课程到他们的课程列表中。自动选择必修课后，程序通过 SelectElectiveCourses 方法，根据用户从控制台输入的信息，为学生手动添加选修课程。这一步骤允许秘书为每位学生选择 1 至 2 门选修课。最后，系统可以通过 printAllStudentCourses 方法在控制台上打印每个学生的所有选课信息，展示每位学生所选的课程详情。

（三）过程截图

全屏截图：



Student 类主要代码：

```java
abstract class Student {
    2 个用法
    protected String studentId;
    3 个用法
    protected String name;
    2 个用法
    protected String className;
    3 个用法
    protected List<Course> courses = new ArrayList<>();

    2 个用法
    public Student(String studentId, String name, String className) {
        this.studentId = studentId;
        this.name = name;
        this.className = className;
    }

    2 个用法
    public void addCourse(Course course) { courses.add(course); }

    1 个用法
    public void printCourses() {
        System.out.println(name + " 的选课信息:");
        for (Course course : courses) {
            System.out.println(course);
        }
    }
```

Undergraduate 和 Postgraduate 类实现代码：

```java
class Undergraduate extends Student {
    1 个用法
    public Undergraduate(String studentId, String name, String className) { super(studentId, name, className); }
}

1 个用法
class Postgraduate extends Student {
    2 个用法
    private String supervisor;

    1 个用法
    public Postgraduate(String studentId, String name, String className, String supervisor) {
        super(studentId, name, className);
        this.supervisor = supervisor;
    }
```

Course 类主要代码：

```java
abstract class Course {
    3 个用法
    protected String courseId;
    3 个用法
    protected String courseName;
    3 个用法
    protected int credits;
    3 个用法
    protected CourseType courseType;

    2 个用法
    public Course(String courseId, String courseName, int credits, CourseType courseType) {
        this.courseId = courseId;
        this.courseName = courseName;
        this.credits = credits;
        this.courseType = courseType;
    }

    @Override
    public String toString() {
        return String.format("%s (%s), 学分: %d, 类型: %s", courseName, courseId, credits, courseType == CourseType.REQUIRED ?
    }
```

RequiredCourse 和 ElectiveCourse 主要代码：

```java
class RequiredCourse extends Course {
    1 个用法
    public RequiredCourse(String courseId, String courseName, int credits) {
        super(courseId, courseName, credits, CourseType.REQUIRED);
    }
}

4 个用法
class ElectiveCourse extends Course {
    1 个用法
    public ElectiveCourse(String courseId, String courseName, int credits) {
        super(courseId, courseName, credits, CourseType.ELECTIVE);
    }
}
```

CourseSelectionSystem 主要代码：

```java
public class CourseSelectionSystem {
    4 个用法
    private List<Student> students = new ArrayList<>();
    3 个用法
    private List<Course> courses = new ArrayList<>();
    1 个用法
    private Scanner scanner = new Scanner(System.in);

    2 个用法
    public void addStudent(Student student) {
        students.add(student);
    }

    2 个用法
    public void addCourse(Course course) {
        courses.add(course);
    }
```

```java
    //自动选课，为每个学生自动选修所有必修课
    1 个用法
    public void autoSelectRequiredCourses() {
        for (Student student : students) {
            for (Course course : courses) {
                if (course.getCourseType() == CourseType.REQUIRED) {
                    student.addCourse(course);
                }
            }
        }
    }
```

```java
//手动选课，为每个同学选修1-2门选修课
1 个用法
public void SelectElectiveCourses() {
    System.out.println("现在开始手动选课操作。");
    for (Student student : this.getStudents()) {
        System.out.println("为学生 " + student.getName() + " (" + student.getStudentId() + ") 选课。");
        List<String> selectedElectiveCourseIds = new ArrayList<>();
        for (Course course : student.getCourses()) {
            if (course instanceof ElectiveCourse) {
                selectedElectiveCourseIds.add(course.getCourseId());
            }
        }
        int electiveCount = 0;
        while (electiveCount < 2) {
            System.out.println("当前可选的选修课程有：");
            for (Course course : this.getCourses()) {
                if (course instanceof ElectiveCourse && !selectedElectiveCourseIds.contains(course.getCourseId())) {
                    System.out.println(course.getCourseId() + ": " + course.getCourseName());
                }
            }
            System.out.println("请输入选修课程编号来为学生 " + student.getName() + " 选修课程（输入'完成'结束选课）：");
            String input = scanner.nextLine();
            if ("完成".equals(input)) {
                break;
            }
            boolean validCourseSelected = false;
            for (Course course : this.getCourses()) {
                if (course.getCourseId().equals(input) && course instanceof ElectiveCourse && !selectedElectiveCourseIds.cor
                    student.addCourse(course);
                    selectedElectiveCourseIds.add(course.getCourseId());
                    electiveCount++;
                    System.out.println("已为学生 " + student.getName() + " 选修了课程：" + course.getCourseName());
                    validCourseSelected = true;
                    break;
                }
            }
            if (!validCourseSelected) {
                System.out.println("课程编号无效或课程已选，请重新输入。");
            }
            if (electiveCount == 2) {
                System.out.println("已为学生 " + student.getName() + " 选修了最多允许的选修课程数目（2门）。");
                break;
            }
        }
    }
}

1 个用法
```

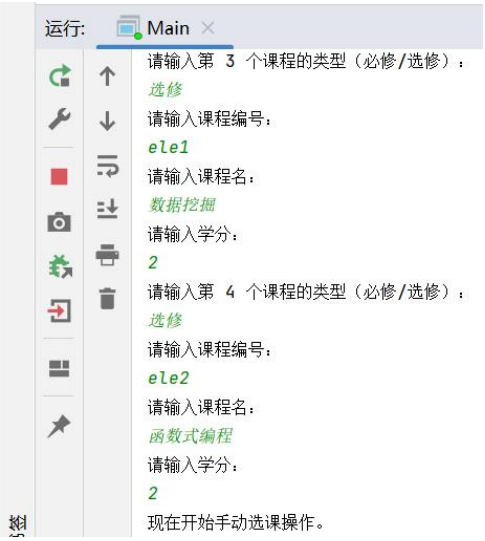
Main 类主要代码：

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    CourseSelectionSystem system = new CourseSelectionSystem();

    System.out.println("请输入学生数量：");
    int studentCount = Integer.parseInt(scanner.nextLine());
    for (int i = 0; i < studentCount; i++) {
        System.out.println("请输入第 " + (i + 1) + " 个学生的类型（本科生/研究生）：");
        String type = scanner.nextLine();
        System.out.println("请输入学号：");
        String studentId = scanner.nextLine();
        System.out.println("请输入姓名：");
        String name = scanner.nextLine();
        System.out.println("请输入班级：");
        String className = scanner.nextLine();
        if ("本科生".equals(type)) {
            system.addStudent(new Undergraduate(studentId, name, className));
        } else if ("研究生".equals(type)) {
            System.out.println("请输入导师姓名：");
            String supervisor = scanner.nextLine();
            system.addStudent(new Postgraduate(studentId, name, className, supervisor));
        }
    }

    System.out.println("请输入课程数量：");
    int courseCount = Integer.parseInt(scanner.nextLine());
    for (int i = 0; i < courseCount; i++) {
        System.out.println("请输入第 " + (i + 1) + " 个课程的类型（必修/选修）：");
        String type = scanner.nextLine();
        System.out.println("请输入课程编号：");
        String courseId = scanner.nextLine();
        System.out.println("请输入课程名：");
```

Main 类运行情况：

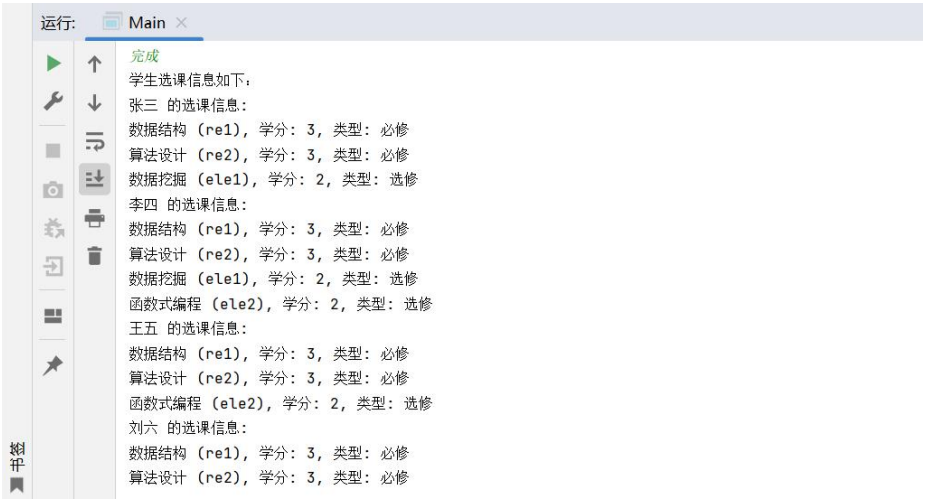
- 创建 4 个学生信息（2 个本科生，2 个研究生）

- 创建 4 门课程信息（2 门必修，2 门选修）



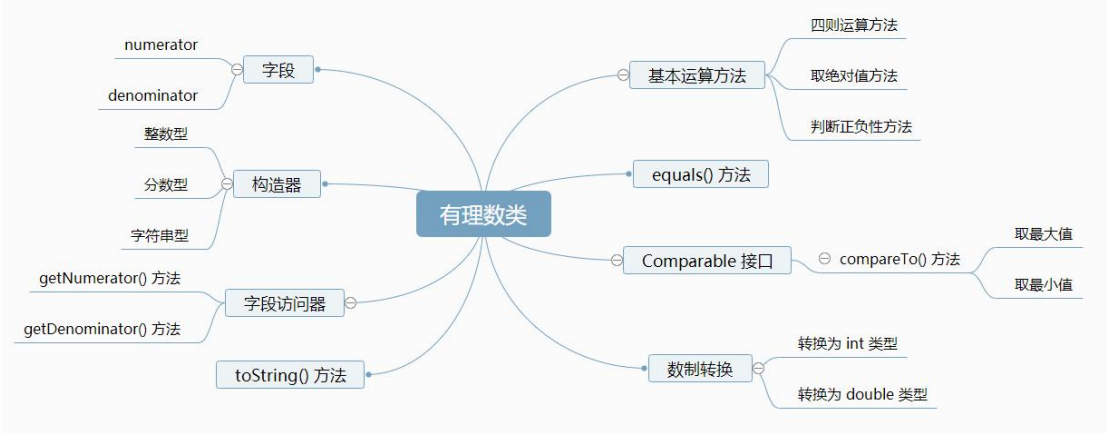
- 秘书手动选课：为每个同学选修 1-2 门选修课

- 打印出每个学生的选课信息



2. 题目二：

写一个有理数类（Rational）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。

（一）实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK 版本：JDK17

（二）实现过程

实验中实现了 Rational 类和 Test 类，其设计用于表示和操作有理数，有理数是由两个整数的比（分子和分母）所构成的数，重点分析 Rational 类：

- 主要成员变量
  - numerator（分子）：表示有理数的分子部分，为整数类型。
  - denominator（分母）：表示有理数的分母部分，为整数类型。分母不能为零。
- 构造方法
  - 接受两个整数参数：分别代表分子和分母，用于创建有理数实例。
  - 接受一个字符串参数：字符串格式为"分子/分母"，用于从字符串表示创建有理数实例。

- 接受一个浮点数参数：将浮点数转换为其有理数表示形式。

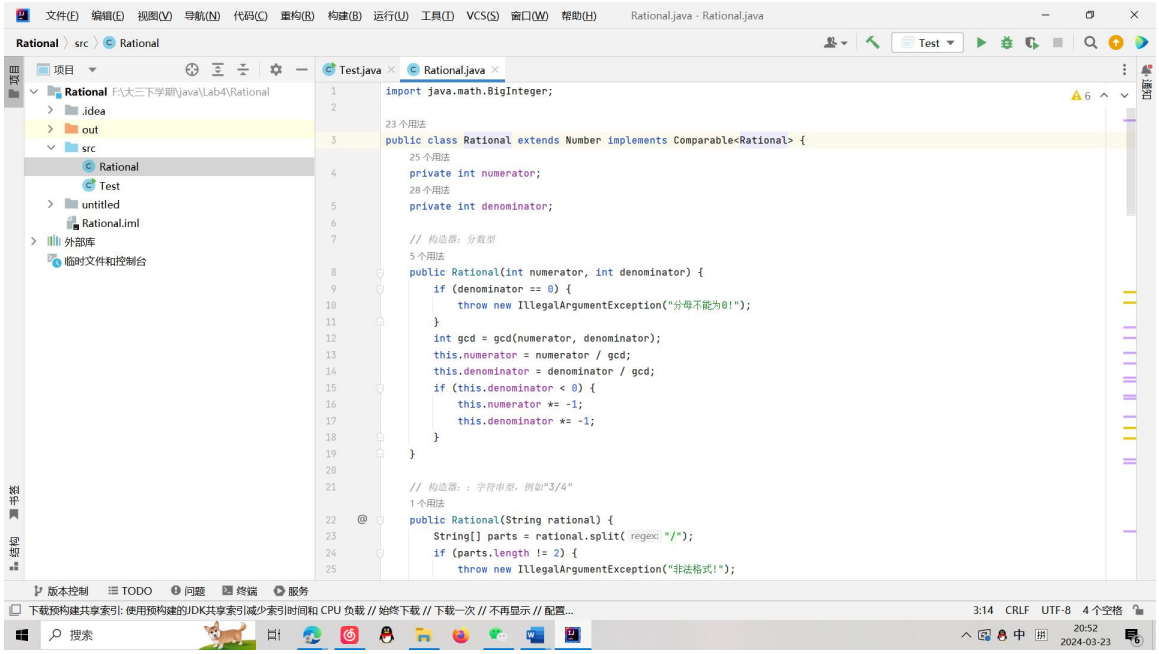
- 主要方法
  - getNumerator()和 getDenominator()：获取有理数的分子和分母。
  - toString()：返回有理数的字符串表示，形式为"分子/分母"。
  - 基本运算方法：包括加（add）、减（subtract）、乘（multiply）、除（divide）等操作。
  - abs()：取绝对值。
  - isPositive()和 isNegative()：判断有理数的正负性。
  - equals(Object obj)：判断两个有理数是否相等。重写自 Object 类。
  - 实现 Comparable<Rational>接口：允许有理数之间进行比较，主要用于排序。
  - 数制转换方法：包括转换为整型（toInt）和双精度浮点型（toDouble）。

- 调用关系

Rational 类的对象可以独立存在，其方法主要涉及到与其他 Rational 对象的交互。运算方法创建并返回新的 Rational 实例，这些实例是对原有理数进行运算后的结果。equals 方法和 compareTo 方法用于比较两个 Rational 对象，支持有理数的等值判断和排序。

（三）过程截图

全屏截图：



Rational 类主要代码：

```
3   public class Rational extends Number implements Comparable<Rational> {
        25 个用法
4       private int numerator;
        28 个用法
5       private int denominator;
6
7       // 构造器：分数型
        5 个用法
8       public Rational(int numerator, int denominator) {
9           if (denominator == 0) {
10              throw new IllegalArgumentException("分母不能为0!");
11          }
12          int gcd = gcd(numerator, denominator);
13          this.numerator = numerator / gcd;
14          this.denominator = denominator / gcd;
15          if (this.denominator < 0) {
16              this.numerator *= -1;
17              this.denominator *= -1;
18          }
19      }
20
21      // 构造器：：字符串型，例如"3/4"
        1 个用法
22  @   public Rational(String rational) {
23          String[] parts = rational.split( regex: "/");
24          if (parts.length != 2) {
25              throw new IllegalArgumentException("非法格式!");
26          }
27          int num = Integer.parseInt(parts[0]);
28          int times = Integer.parseInt(parts[1]);
29          int gcd = gcd(num, times);
30          this.numerator = num / gcd;
31          this.denominator = times / gcd;
32      }
33
```

四则运算：

```
58      // 基本运算方法
        1 个用法
59  @   public Rational add(Rational other) {
60          int newNumerator = this.numerator * other.denominator + this.denominator * other.numerator;
61          int newDenominator = this.denominator * other.denominator;
62          return new Rational(newNumerator, newDenominator);
63      }
64
        1 个用法
65  @   public Rational subtract(Rational other) {
66          int newNumerator = this.numerator * other.denominator - this.denominator * other.numerator;
67          int newDenominator = this.denominator * other.denominator;
68          return new Rational(newNumerator, newDenominator);
69      }
70
        1 个用法
71  @   public Rational multiply(Rational other) {
72          return new Rational( numerator: this.numerator * other.numerator,  denominator: this.denominator * other.denominator);
73      }
74
        1 个用法
75  @   public Rational divide(Rational other) {
76          return new Rational( numerator: this.numerator * other.denominator,  denominator: this.denominator * other.numerator);
77      }
```

判断正负性和相等：

```
79      // 取绝对值和判断正负性方法
        0 个用法
80      public Rational abs() { return new Rational(Math.abs(numerator), denominator); }
83
        0 个用法
84      public boolean isPositive() { return numerator > 0; }
87
        0 个用法
88      public boolean isNegative() { return numerator < 0; }
91
92      // equals方法
93      @Override
94      public boolean equals(Object obj) {
95          if (this == obj) return true;
96          if (!(obj instanceof Rational)) return false;
97          Rational other = (Rational) obj;
98          return this.numerator == other.numerator && this.denominator == other.denominator;
99      }
```

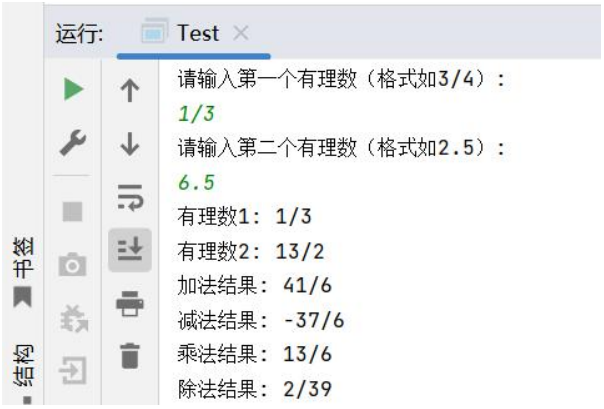
实现 Number 类的 Comparable 接口

```
122     // Comparable接口
123     @Override
124 @   public int compareTo(Rational other) {
125         return Integer.compare(this.numerator * other.denominator, this.denominator * other.numerator);
126     }
```

Test 类主要代码：

```
4   public static void main(String[] args) {
5       Scanner scanner = new Scanner(System.in);
6       System.out.println("请输入第一个有理数（格式如3/4）:");
7       String input1 = scanner.nextLine();
8       Rational r1 = new Rational(input1);
9
10      System.out.println("请输入第二个有理数（格式如2.5）:");
11      double input2 = scanner.nextDouble();
12      Rational r2 = new Rational(input2);
13
14      System.out.println("有理数1: " + r1);
15      System.out.println("有理数2: " + r2);
16      System.out.println("加法结果: " + r1.add(r2));
17      System.out.println("减法结果: " + r1.subtract(r2));
18      System.out.println("乘法结果: " + r1.multiply(r2));
19      System.out.println("除法结果: " + r1.divide(r2));
20  }
```

Test 类运行情况：



3. 题目三：

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。

（一）实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK 版本：JDK17

（二）实现过程

本题具体设计如下：

- Graph（抽象基类）
  - 用途：提供一个抽象方法 getArea()，用于计算图形的面积。
  - getArea()：抽象方法，要求继承 Graph 类的子类必须实现这个方法以返回图形的面积。
- Rectangle（Graph 的子类）
  - 用途：表示矩形图形，继承自 Graph 类。
  - 主要成员：length（double）表示矩形的长度以及 width（double）表示矩形的宽度。
  - 构造方法：接受长度和宽度作为参数并初始化。
  - getArea()：重写 Graph 类的方法，计算并返回矩形的面积。
- Triangle（Graph 的子类）
  - 用途：表示三角形图形，继承自 Graph 类。
  - 主要成员：a、b、c（double）表示三角形的三边长。
  - 构造方法：接受三边长作为参数并初始化。
  - getArea()：重写 Graph 类的方法，使用海伦公式计算并返回三角形的面积。
- Main 类

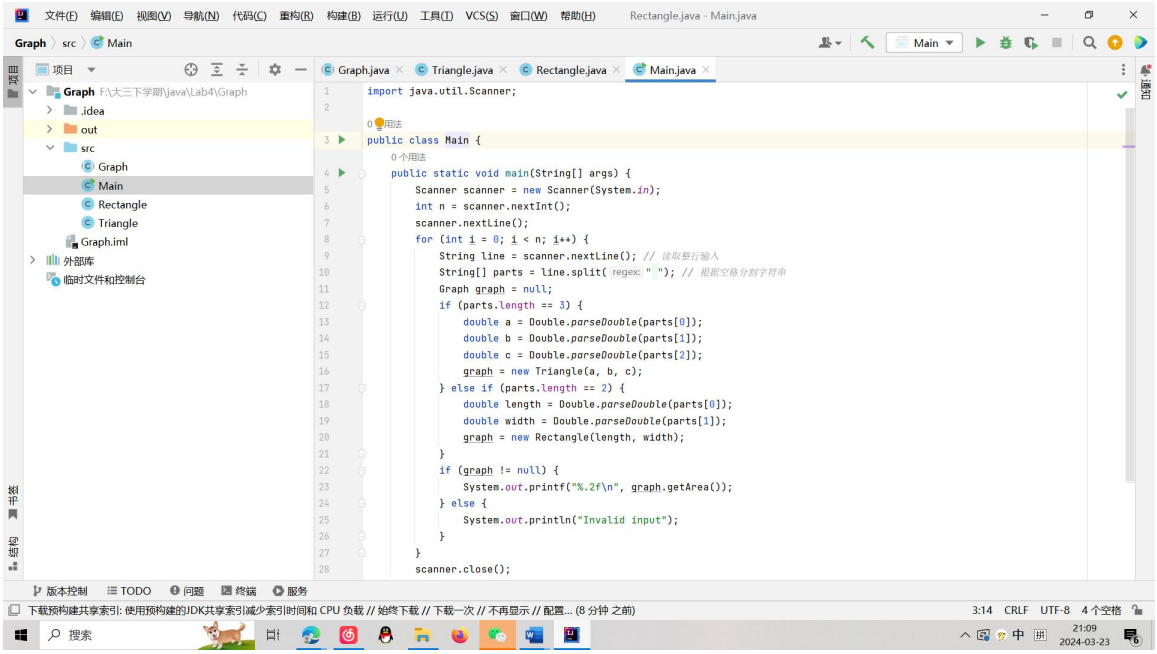
- main(String[] args)：主方法，负责处理输入，创建对应的图形对象，并计算打印这些图形的面积。

- 运行逻辑和调用关系

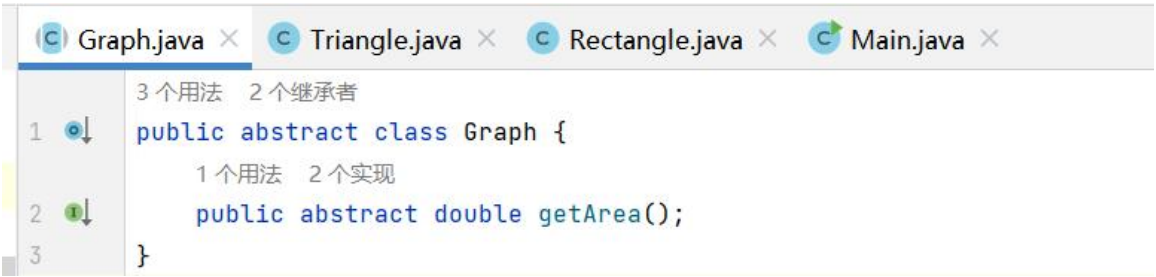
程序启动后，Main 类的 main 方法首先读取输入，根据输入的参数数量决定创建 Triangle 或 Rectangle 对象。对于每个输入行，根据参数数量判断图形类型。如果数量为 2，认为是矩形；如果数量为 3，认为是三角形。对每个图形对象调用 getArea()方法计算其面积，并打印出来。

（三）过程截图

全屏截图：



Graph 类主要代码：



Rectangle 主要代码：

```java
public class Rectangle extends Graph {
    2 个用法
    private double length, width;

    1 个用法
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    1 个用法
    @Override
    public double getArea() {
        return length * width;
    }
}
```

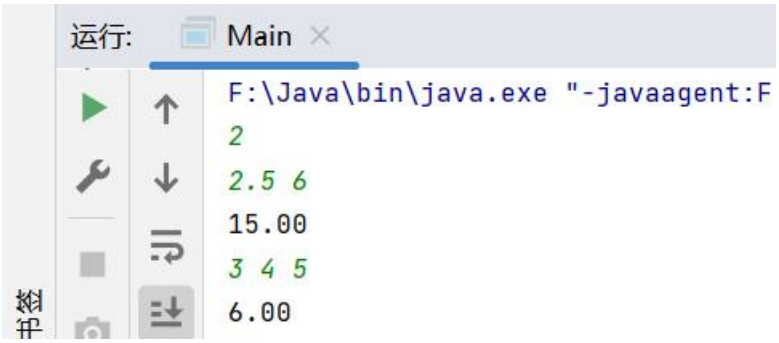
Triangle 主要代码：

```java
public class Triangle extends Graph {
    3 个用法
    private double a, b, c;

    1 个用法
    public Triangle(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }
    1 个用法
    @Override
    public double getArea() {
        double s = (a + b + c) / 2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c)); // 海伦公式计算面积
    }
```

Main 主要代码：

```java
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    scanner.nextLine();
    for (int i = 0; i < n; i++) {
        String line = scanner.nextLine(); // 读取整行输入
        String[] parts = line.split( regex: " "); // 根据空格分割字符串
        Graph graph = null;
        if (parts.length == 3) {
            double a = Double.parseDouble(parts[0]);
            double b = Double.parseDouble(parts[1]);
            double c = Double.parseDouble(parts[2]);
            graph = new Triangle(a, b, c);
        } else if (parts.length == 2) {
            double length = Double.parseDouble(parts[0]);
            double width = Double.parseDouble(parts[1]);
            graph = new Rectangle(length, width);
        }
        if (graph != null) {
            System.out.printf("%.2f\n", graph.getArea());
        } else {
            System.out.println("Invalid input");
        }
    }
```

Main 执行情况：



4. 题目四：

创建一个简单的 JavaFX 绘图程序，要求如下：

- 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。
- 绘制 20 个图形后，不再创建新的图形。

（一）实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK 版本：JDK17，JavaFx22

（二）实现过程

本题具体设计如下：

- DrawGraphApplication 类

继承：继承自 Application 类，是 JavaFX 程序的入口点。

- 主要成员：
  - MAX_SHAPES：定义最大绘制图形数量。
  - shapesDrawn：追踪已绘制图形的数量。

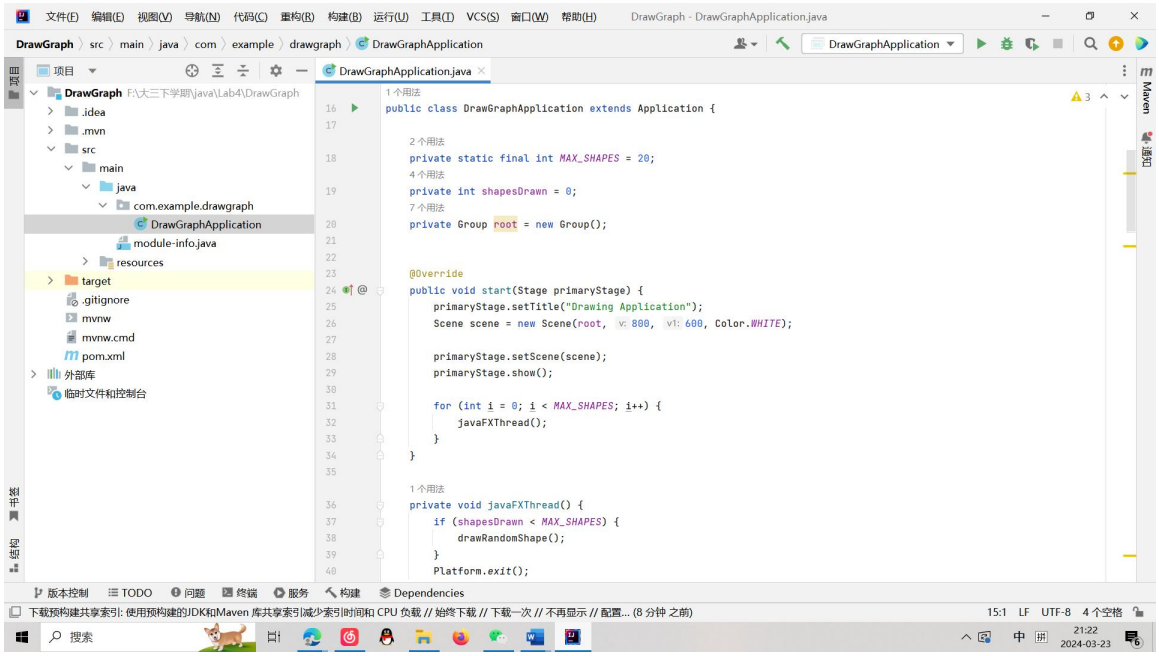
- root（Group 对象）：用作绘制图形的容器。
- 主要方法：
- start(Stage primaryStage)：覆盖自 Application，设置舞台（Stage）和场景（Scene），并开始绘制图形流程。
- javaFXThread()：控制图形绘制的流程，确保不超过最大图形数量。
- drawRandomShape()：随机选择图形类型并调用相应的绘制方法。
- drawLine(), drawRectangle(), drawEllipse()：具体绘制线条、矩形和椭圆的方法，使用 TextInputDialog 接收用户输入，并将绘制的图形添加到 root 容器。
- showAlert(String title, String message)：显示错误提示信息的辅助方法。

- JavaFX 组件和控件
  - Scene 和 Group：Scene 表示程序的主画面，Group 作为根节点（root）容纳其他图形节点。所有绘制的图形都会添加到这个 Group 中。
  - Line、Rectangle、Ellipse：JavaFX 提供的图形类，分别用于绘制线条、矩形和椭圆。
  - TextInputDialog：用于从用户那里接收图形参数的输入对话框。
  - Alert：用于显示错误信息的对话框。
- 调用关系
  - start 方法是程序的起点，它初始化舞台和场景，并开始绘制图形。
  - javaFXThread 方法控制图形绘制的主循环，直至达到最大图形数量。
  - drawRandomShape 方法根据随机数选择图形类型，并调用对应的绘制方法（drawLine, drawRectangle, drawEllipse）。

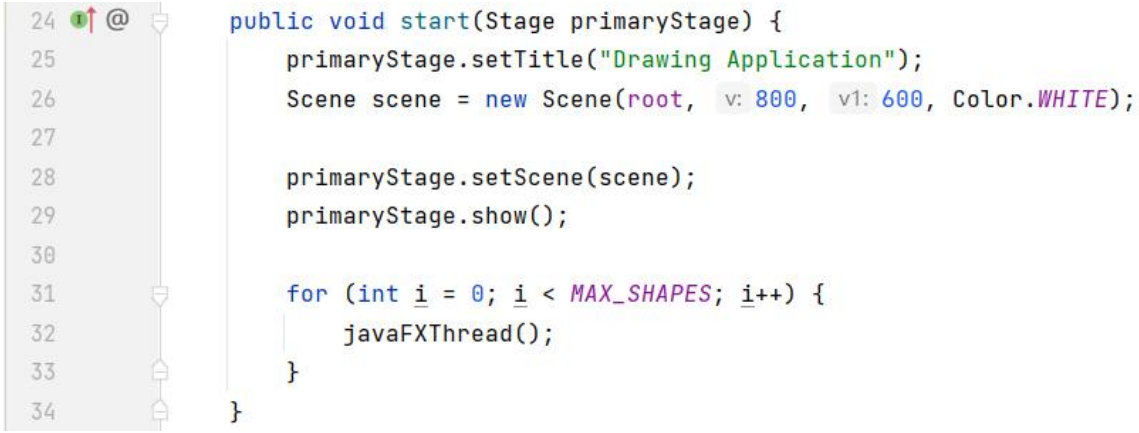
- 绘制方法通过显示 TextInputDialog 来接收用户输入，根据输入创建并添加图形到场景中。

（三）过程截图

全屏截图：



主要方法实现：



随机绘制图形：

```java
    private void drawRandomShape() {
        int shapeType = (int) (Math.random() * 3);
        switch (shapeType) {
            case 0:
                drawLine();
                break;
            case 1:
                drawRectangle();
                break;
            case 2:
                drawEllipse();
                break;
        }
    }
```

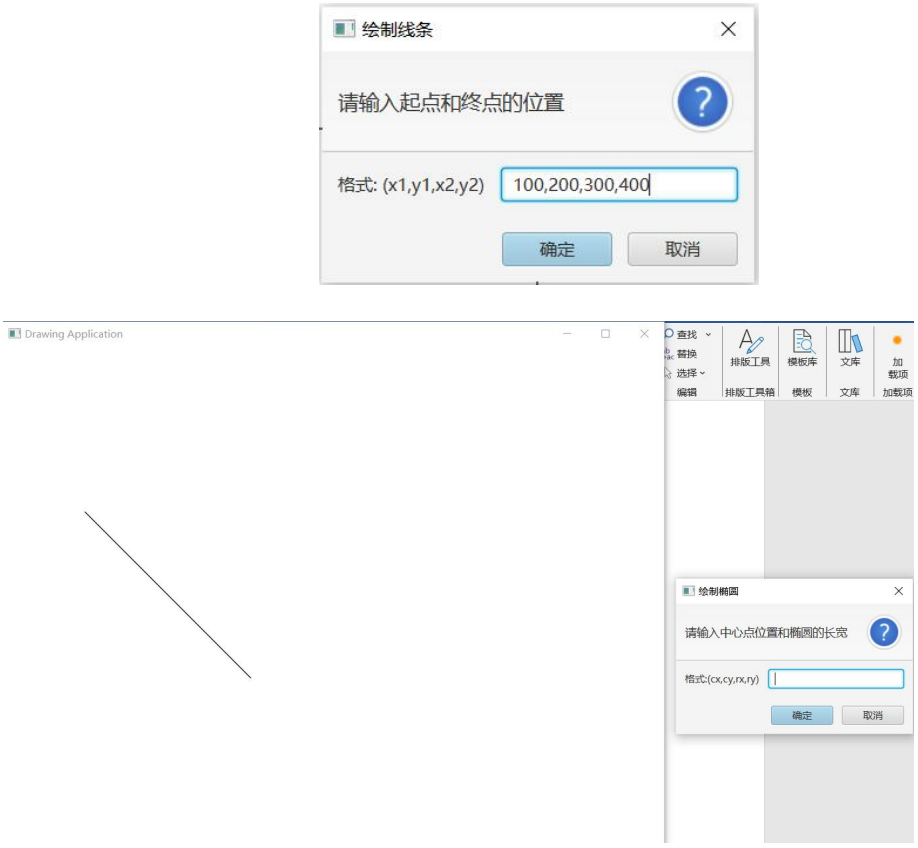
Drawline 方法，其他方法类似：

```java
private void drawLine() {
    TextInputDialog dialog = new TextInputDialog();
    dialog.setTitle("绘制线条");
    dialog.setHeaderText("请输入起点和终点的位置");
    dialog.setContentText("格式: (x1,y1,x2,y2)");

    Optional<String> result = dialog.showAndWait();
    result.ifPresent(input -> {
        String[] parts = input.split( regex: ",");
        if (parts.length == 4) {
            try {
                double x1 = Double.parseDouble(parts[0]);
                double y1 = Double.parseDouble(parts[1]);
                double x2 = Double.parseDouble(parts[2]);
                double y2 = Double.parseDouble(parts[3]);
                root.getChildren().clear();
                Line line = new Line(x1, y1, x2, y2);
                root.getChildren().add(line);
                shapesDrawn++;
            } catch (NumberFormatException e) {
                showAlert( title: "非法输入", message: "请输入合法数字。");
                drawLine();
            }
        } else {
            showAlert( title: "非法输入", message: "请按照提示的格式输入。");
            drawLine();
        }
    });
}
```

运行截图：

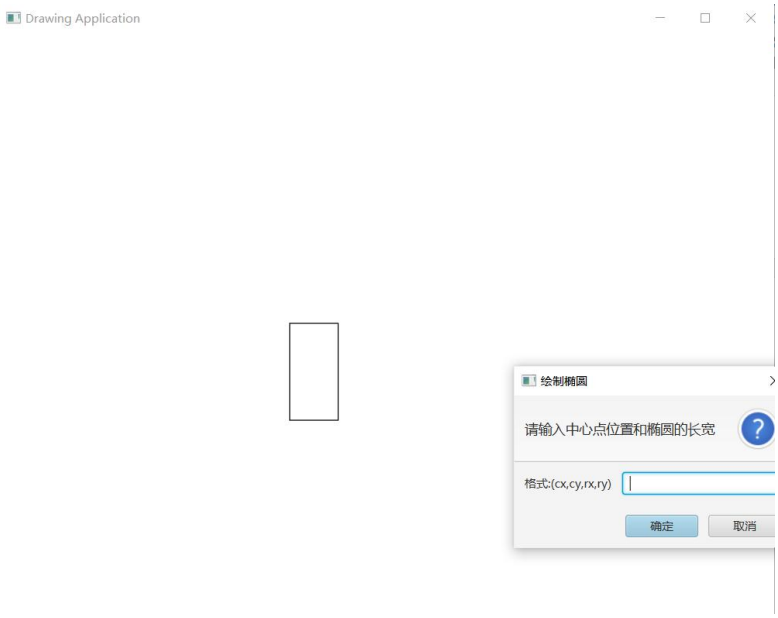
绘制椭圆：

绘制线条：



绘制矩形：


绘制矩形
请输入左上顶点的位置以及矩形的宽度和高度
格式: (x,y,width,height)
300,300,50,100
确定
取消


Drawing Application
绘制椭圆
请输入中心点位置和椭圆的长宽
格式:(cx,cy,rx,ry)
确定
取消

错误情况：


绘制椭圆
请输入中心点位置和椭圆的长宽
格式:(cx,cy,rx,ry)
200,100,100
确定
取消


非法输入
请按照提示的格式输入。
确定

绘制 20 个图形后程序自动退出。

5. 题目五：

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。

（一）实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK 版本：JDK17

（二）实现过程

程序包含两个主要部分：TrafficLight 枚举和 Test 类，具体如下：

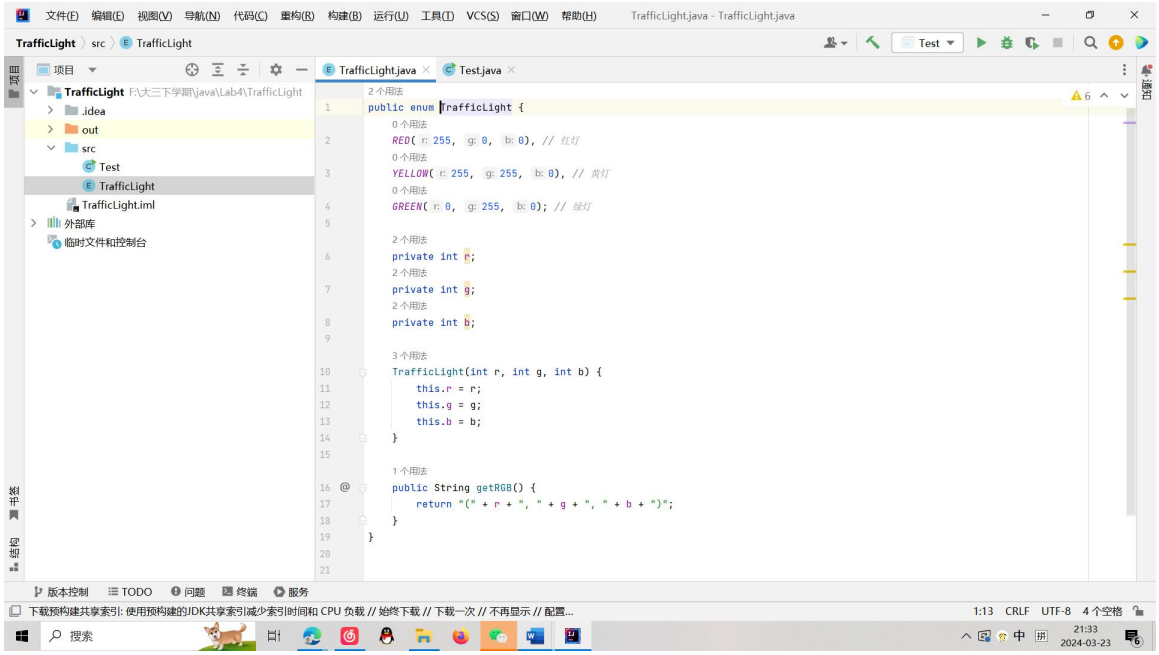
- TrafficLight 枚举
  - 用途：定义了交通信号灯的三种颜色（红、黄、绿）以及它们对应的 RGB 颜色值。
  - r、g、b：代表颜色的红（R）、绿（G）、蓝（B）分量。

- TrafficLight(int r, int g, int b)：用于初始化每个枚举实例的 RGB 值。
- getRGB()：返回表示颜色 RGB 值的字符串，格式为"(r, g, b)"。

- Test 类
  - 用途：包含了 main 方法，用于遍历 TrafficLight 枚举的实例，并打印出每个实例的名称和对应的 RGB 值。
  - 在 main 方法中，使用 for 循环遍历 TrafficLight 枚举的所有实例。对于每个实例，将实例的名称（通过 toString()方法隐式获取）和通过调用 getRGB()方法获取的 RGB 值打印到控制台。

（三）过程截图

全屏截图：



TrafficLight 类主要代码：

```java
public enum TrafficLight {
    0 个用法
    RED( r: 255,  g: 0,  b: 0), // 红灯
    0 个用法
    YELLOW( r: 255,  g: 255,  b: 0), // 黄灯
    0 个用法
    GREEN( r: 0,  g: 255,  b: 0); // 绿灯

    2 个用法
    private int r;
    2 个用法
    private int g;
    2 个用法
    private int b;

    3 个用法
    TrafficLight(int r, int g, int b) {
        this.r = r;
        this.g = g;
        this.b = b;
    }

    1 个用法
    public String getRGB() {
        return "(" + r + ", " + g + ", " + b + ")";
    }
}
```

Test 类主要代码：

```java
public class Test {
    0 个用法
    public static void main(String[] args) {
        // 遍历TrafficLight枚举的所有值
        for (TrafficLight light : TrafficLight.values()) {
            System.out.println(light + " Light RGB: " + light.getRGB());
        }
    }
}
```

Test 类运行情况：

三、实验总结与心得记录

通过完成本次实验，我深入理解和掌握了 Java 编程中的几个重要概念：图形界面的基本操作、枚举的定义与应用、以及继承机制的实际运用。在实验过程中，我遇到了一些挑战，特别是在实现图形界面和处理用户输入时。通过不断地查阅资料和反复试验，我逐步解决了这些问题。这个过程让我更加熟悉了 JavaFX 库的使用。同时，通过枚举和继承的实践，我对 Java 语言的面向对象编程有了更深入的理解。

此外，我也认识到了代码设计的重要性。在实验中，良好的代码结构和清晰的逻辑关系大大提高了代码的可读性和可维护性。例如，通过继承和多态性，我能够用更简洁的代码实现功能更为丰富的程序。