|  |
| --- |
| **《Java面向对象程序设计SSD3》**  **实验报告**  项目名称 实验3《对象和类》  专业班级 软件工程2005班  学 号 8209200504  姓 名 李均浩  **实验成绩：**  **批阅教师：邝砾**  2021年 5 月 10 日 |

**中南大学计算机学院实验报告**

**课程名称** Java面向对象程序设计SSD3

**实验项目名称** 实验3《对象和类》

**学生姓名** 李均浩 **专业班级** 软件工程2005班 **学号** 8209200504

**实验成绩**  **日期** 2021年5月10日

实验学时： 2

每组人数： 1

实验类型： 1 （1：基础性 2：综合性 3：设计性 4：研究性）

实验要求： 1 （1：必修 2：选修 3：其它）

实验类别： 2 （1：基础 2：专业基础 3：专业 4：其它）

1. **实验目的**
2. 设计类，并画出UML类图
3. 实现UML中的类
4. 使用类开发应用程序

**二、实验内容**

1、（P305, 9.1）【矩形类Rectangle】遵照9.2节中Circle类的例子，设计一个名为Rectangle的类表示矩形。这个类包括：

* 两个名为width和height的double型数据域，它们分别表示矩形的宽和高。width和height的默认值都为1。
* 创建默认矩形的无参构造方法。
* 创建width和height为指定值的矩形的构造方法。
* 一个名为getArea()的方法返回这个矩形的面积。
* 一个名为getPerimeter()的方法返回矩形周长。

画出该类的UML图并实现这个类。编写一个测试程序，创建两个Rectangle对象：一个矩形的宽为4高为40，另一个矩形的宽为3.5高为35.9。依次显示每个矩形的宽、高、面积和周长。

2. （P307, 9.8）【风扇类Fan】设计一个名为Fan的类表示一个风扇。这个类包括：

* 三个名为SLOW、MEDIUM和FAST而值为1、2、3的常量表示风扇的速度。
* 一个名为speed的int类型私有数据域表示风扇的速度（默认值为SLOW）。
* 一个名为on的boolean类型私有数据域表示风扇是否打开（默认值为false）。
* 一个名为radius的double类型私有数据域表示风扇的半径（默认值为5）。
* 一个名为color的String类型私有数据域表示风扇的颜色（默认值为blue）。
* 这四个数据域的访问器和修改器。
* 一个创建默认风扇的无参构造方法。
* 一个名为toString()的方法返回描述风扇的字符串。如果风扇是打开的，那么该方法在一个组合的字符串中返回风扇的速度、颜色和半径。如果风扇没有打开，该方法返回一个由“fan is off”和风扇颜色、半径组成的字符串。

画出该类的UML图。实现这个类。编写一个测试程序，创建两个Fan对象。将第一个对象设置为最大速度、半径为10、颜色为yellow、状态为打开。将第二个对象设置为中等速度、半径为5、颜色为blue、状态为关闭。通过调用它们的toString方法显示这些对象。

3.（P308，9.10\*）【二次方程式】为二次方程式ax2+bx+c=0设计一个名为QuadraticEquation的类。这个类包括：

* 代表三个系数的私有数据域a、b、c。
* 一个参数为a、b、c的构造方法。
* a、b、c的三个get方法。
* 一个名为getDiscriminant()的方法返回判别式，b2-4ac。
* 一个名为getRoot1()和getRoot2()的方法返回等式的两个根。

这些方法只有在判别式为非负数时才有用。如果判别式为负，方法返回0。

画出该类的UML图。实现这个类。编写一个测试程序，提示用户输入a、b、c的值，然后显示判别式的结果。如果判别式为正数，显示两个根；如果判别式为0，显示一个根；否则，显示“The equation has no roots”。

4.（P308, 9.13\*\*）【位置类】设计一个名为Location的类，定位二维数组中的最大值及其位置。这个类包括公共的数据域row、column和maxValue，二维数组中的最大值及其下标用double型的maxValue以及int型的row和column存储。

编写下面的方法，返回一个二维数组中最大值的位置。

public static Location locateLargetst(double[][] a)

返回值是一个Location的实例。编写一个测试程序，提示用户输入一个二维数组，然后显示这个数组中的最大元素及下标。运行实例如下：

输入二维数组的行数和列数: 3 4

输入数组:

23.5 35 2 10

4.5 3 45 3.5

35 44 5.5 9.6

最大元素及其下标是: 45 在(1,2)

三、实验要求：

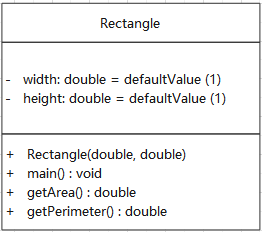
要求每个学生独立完成实验任务。

四、实验报告

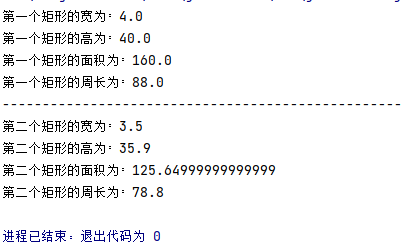
**1. 实验结果与分析**

**第一题：**

UML类图：



( 1 )测试结果：

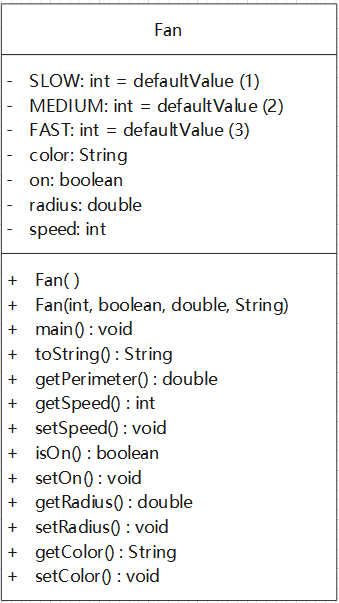


( 2 )结果分析

在main函数中构建了两个Rectangle类的实例，向构造方法传递矩形的长、宽，经过getArea()和getPerimeter()的计算得到了两个矩形的面积和周长，结果符合预期。

**第二题：**

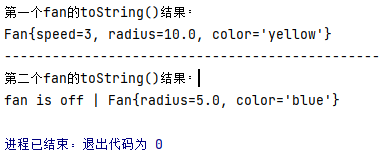
UML类图：



测试点( a )：



( 1 )测试结果：



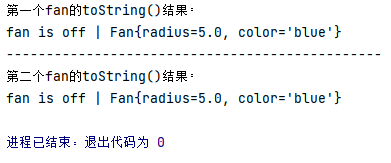
( 2 )结果分析

通过给构造函数传入不同值为Fan类的实例设定属性，再通过toString()的字段拼接得到结果，符合预期。

测试点( b )：



( 1 )测试结果：



( 2 )结果分析

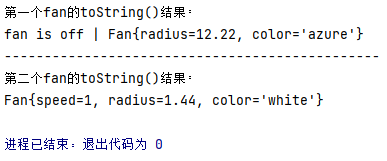
调用无参构造方法，按照默认值设定两个Fan类实例的属性，符合预期。

测试点( c )：



( 1 )测试结果：

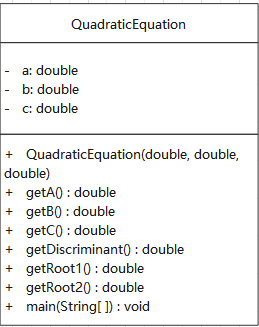
( 2 )结果分析



通过给构造函数传入不同值为Fan类的实例设定属性，再通过toString()的字段拼接得到结果，符合预期。

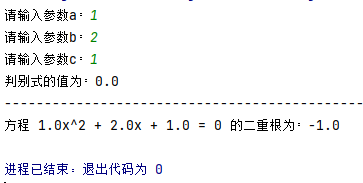
**第三题：**

UML类图：



测试点( a )：1 2 1

( 1 )测试结果：

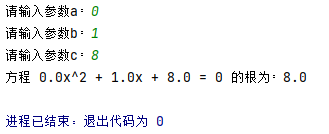


( 2 )结果分析

判别式为0，有两个相同的根，执行getRoot1(),输出结果，符合预期。

测试点( b )：0 1 8

( 1 )测试结果：

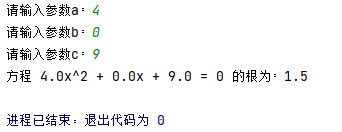


( 2 )结果分析

方程的二次项系数为0，则输出的根的结果为c÷b，符合预期。

测试点( c )：4 0 9

( 1 )测试结果：



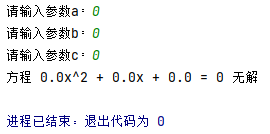
( 2 )结果分析

方程的一次项为0，则输出的根的结果为sqrt(c÷a)，符合预期。

测试点( d )：0 0 0

( 1 )测试结果：

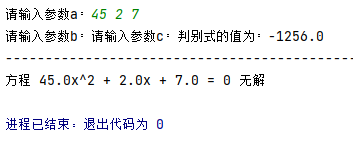
( 2 )结果分析



三个系数都为0，方程无意义，符合预期。

测试点( e )：45 2 7

( 1 )测试结果：



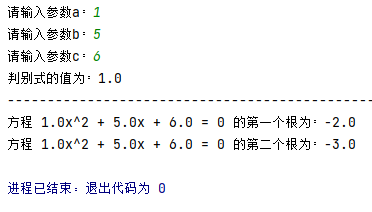
( 2 )结果分析

判别式小于0，方程无解，符合预期。

测试点( f )：1 5 6

( 1 )测试结果：

( 2 )结果分析

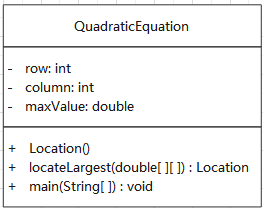


判别式大于0，方程有两个不相等的实数根，执行getRoot1()和getRoot2()，输出两个根的结果，符合预期。

**第四题：**

UML类图：

测试( a )：



( 1 )测试点以及测试结果：

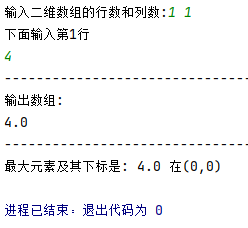


( 2 )结果分析

从a[0][0]开始遍历，遇到比loc.maxValue大的数值就对索引位置和maxValue重新赋值，最后循环结束，返回Location类的实例loc，输出结果，符合预期。

测试( b )：

( 1 )测试点以及测试结果：



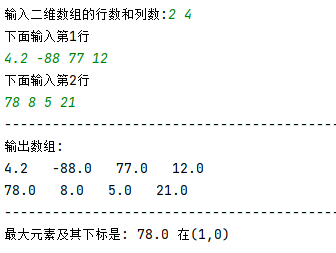
( 2 )结果分析

从a[0][0]开始遍历，遇到比loc.maxValue大的数值就对索引位置和maxValue重新赋值，最后循环结束，返回Location类的实例loc，输出结果，符合预期。

测试( c )：

( 1 )测试点以及测试结果：

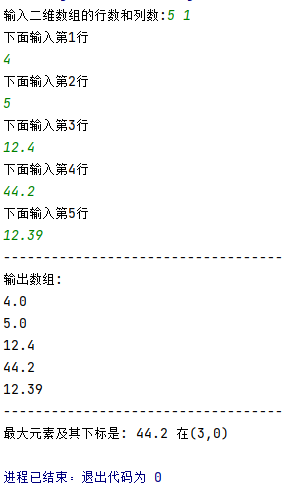
( 2 )结果分析



从a[0][0]开始遍历，遇到比loc.maxValue大的数值就对索引位置和maxValue重新赋值，最后循环结束，返回Location类的实例loc，输出结果，符合预期。

测试( d )：

( 1 )测试点以及测试结果：



( 2 )结果分析

从a[0][0]开始遍历，遇到比loc.maxValue大的数值就对索引位置和maxValue重新赋值，最后循环结束，返回Location类的实例loc，输出结果，符合预期。

**2. 心得体会**

本次实验练习了类的创建，体会了如何去编写一个类，用不同类型的数据来描述一个具象事物的属性，以方法来模拟事物的行为或实现功能。其次也体会到UML图的便利，在创建一个较大的类的时候就应该先绘制类图，将思维捋清楚，这样便能更高效率的编写程序代码，对方法的签名、数据域、类与类之间的联系都会了解得更加清楚，可见UML图是非常好用的一个工具。

【**附源程序**】

public class Rectangle {  
 private double width = 1;  
 private double height = 1;  
  
 Rectangle(double w, double h) {  
 this.width = w;  
 this.height = h;  
 }  
  
 public double getArea() {  
 return this.width \* this.height;  
 }  
  
 public double getPerimeter() {  
 return 2 \* this.width + 2 \* this.height;  
 }  
  
 public static void main(String[] args) {  
 Rectangle test\_rectangle\_1 = new Rectangle(4, 40);  
 Rectangle test\_rectangle\_2 = new Rectangle(3.5, 35.9);  
 System.*out*.println("第一个矩形的宽为：" + test\_rectangle\_1.width);  
 System.*out*.println("第一个矩形的高为：" + test\_rectangle\_1.height);  
 System.*out*.println("第一个矩形的面积为：" + test\_rectangle\_1.getArea());  
 System.*out*.println("第一个矩形的周长为：" + test\_rectangle\_1.getPerimeter());  
 System.*out*.println("--------------------------------------------------");  
 System.*out*.println("第二个矩形的宽为：" + test\_rectangle\_2.width);  
 System.*out*.println("第二个矩形的高为：" + test\_rectangle\_2.height);  
 System.*out*.println("第二个矩形的面积为：" + test\_rectangle\_2.getArea());  
 System.*out*.println("第二个矩形的周长为：" + test\_rectangle\_2.getPerimeter());  
 }  
}

源代码3.1 Rectangle.java

public class Fan {  
 static final int *SLOW* = 1;  
 static final int *MEDIUM* = 2;  
 static final int *FAST* = 3;  
 private int speed;  
 private boolean on;  
 private double radius;  
 private String color;  
  
 public int getSpeed() {  
 return speed;  
 }  
  
 public void setSpeed(int speed) {  
 this.speed = speed;  
 }  
  
 public boolean isOn() {  
 return on;  
 }  
  
 public void setOn(boolean on) {  
 this.on = on;  
 }  
  
 public double getRadius() {  
 return radius;  
 }  
  
 public void setRadius(double radius) {  
 this.radius = radius;  
 }  
  
 public String getColor() {  
 return color;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 Fan() {  
 this.speed = *SLOW*;  
 this.on = false;

源代码3.2 Fan.java

this.radius = 5.0;  
 this.color = "blue";  
 }  
  
 Fan(int speed, boolean on, double radius, String color) {  
 this.speed = speed;  
 this.on = on;  
 this.radius = radius;  
 this.color = color;  
 }  
  
 public String toString() {  
 if (this.on)  
 return "Fan{" +  
 "speed=" + speed +  
 ", radius=" + radius +  
 ", color='" + color + '\'' +  
 '}';  
 else  
 return "fan is off | " + "Fan{" +  
 "radius=" + radius +  
 ", color='" + color + '\'' +  
 '}';  
 }  
  
 public static void main(String[] args) {  
 Fan fan\_1 = new Fan(*FAST*, true, 10, "yellow");  
 Fan fan\_2 = new Fan(*MEDIUM*, false, 5, "blue");  
  
 System.*out*.println("第一个fan的toString()结果：");  
 System.*out*.println(fan\_1.toString());  
 System.*out*.println("---------------------------------------------");  
 System.*out*.println("第二个fan的toString()结果：");  
 System.*out*.println(fan\_2.toString());  
 }  
}

源代码3.3 QuadraticEquation.java

import java.util.Scanner;  
  
public class QuadraticEquation {  
 private double a;  
 private double b;  
 private double c;  
  
 QuadraticEquation(double a, double b, double c) {  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
  
 public double getA() {  
 return a;  
 }  
  
 public double getB() {  
 return b;  
 }  
  
 public double getC() {  
 return c;  
 }  
  
 public double getDiscriminant() {  
 return getB() \* getB() - 4 \* getA() \* getC();  
 }  
  
 public double getRoot1() {  
 if (getDiscriminant() >= 0)  
 return (-getB() + Math.sqrt(getB() \* getB() - 4 \* getA() \* getC())) / (2 \* getA());  
 else  
 return 0;  
 }  
  
 public double getRoot2() {  
 if (getDiscriminant() >= 0)  
 return (-getB() - Math.sqrt(getB() \* getB() - 4 \* getA() \* getC())) / (2 \* getA());  
 else  
 return 0;  
 }

public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("请输入参数a：");  
 double a = input.nextDouble();  
 System.out.print("请输入参数b：");  
 double b = input.nextDouble();  
 System.out.print("请输入参数c：");  
 double c = input.nextDouble();  
  
 QuadraticEquation quad = new QuadraticEquation(a, b, c);  
 if (quad.a != 0 && quad.b != 0) {  
 System.out.println("判别式的值为：" + quad.getDiscriminant());  
 System.out.println("-------------------------------------------------------------------------------------------");  
 }  
 if (quad.a != 0 && quad.b != 0) {  
 if (quad.getDiscriminant() > 0) {  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 的第一个根为：" + quad.getRoot1());  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 的第二个根为：" + quad.getRoot2());  
 } else if (quad.getDiscriminant() == 0) {  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 的二重根为：" + quad.getRoot1());  
 } else {  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 无解");  
 }  
 }  
 if (quad.a == 0 && quad.b != 0)  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 的根为：" + quad.c / quad.b);  
 if (quad.a != 0 && quad.b == 0)  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 的根为：" + Math.sqrt(quad.c / quad.a));  
 if (quad.a == 0 && quad.b == 0)  
 System.out.println("方程 " + quad.a + "x^2 + " + quad.b + "x + " + quad.c + " = 0 无解");  
 }  
}

import java.util.Arrays;  
import java.util.Scanner;  
  
public class Location {  
 public int row;  
 public int column;  
 public double maxValue;  
  
 public static Location locateLargest(double[][] a) {  
 Location loc = new Location();  
 *//初始化对比标示值* loc.maxValue = a[0][0];  
 loc.row = 0;  
 loc.column = 0;  
  
 for (int i = 0; i < a.length; i++) {  
 for (int j = 0; j < a[i].length; j++) {  
 if (a[i][j] > loc.maxValue) {  
 loc.maxValue = a[i][j];  
 loc.row = i;  
 loc.column = j;  
 }  
 }  
 }  
 return loc;  
 }  
  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("输入二维数组的行数和列数:");  
 int row = input.nextInt();  
 int col = input.nextInt();  
 double[][] array = new double[row][col];  
 *//输入数组内容* for (int i = 0; i < row; i++) {  
 System.out.println("下面输入第" + (i + 1) + "行");  
 for (int j = 0; j < col; j++) {  
 array[i][j] = input.nextDouble();  
 }  
 }  
 *//输出数组内容* System.out.println("----------------------------------------------------------");

源代码3.4 Location.java

System.out.println("输出数组:");  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 System.out.print(array[i][j] + " ");  
 }  
 System.out.println();  
 }  
 System.out.println("----------------------------------------------------------");  
 Location loc = locateLargest(array);  
 System.out.println("最大元素及其下标是: " + loc.maxValue + " 在(" + loc.row + "," + loc.column + ")");  
 }  
}