《Java语言程序设计》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机科学与技术 | 年级 | 2017 | 班级 | 计工本1704 |
| 学生姓名 | 付泽坤 | 指导老师 | 李焱 | 时间 | 2019.05.09 |

|  |  |
| --- | --- |
| 实验名称 | 面向对象 |
| 实  验  目  的  及  要  求 | 目的：  了解熟悉顺序Java程序设计的形式，编写完整Java程序。  要求：   * 掌握类的设计。 * 掌握类之间的关系。 * 掌握用类解决问题。 * 会用Java编写完整的程序。 |
| 实  验  环  境 | Microsoft Windows 7 with SP1专业版（简体中文）32位  JDK 1.8  Eclipse 2017  或者Jcreator |
| 实  验  内  容 | 请按照要求编写出完整程序   * 第9章编程练习题: 9.10, 9.12, 9.13 * 第10章编程练习题:10.4-5， |
| 实  验  步  骤  或  实  验  方  案 | **package** cn.work.twelve;  **import** java.util.Scanner;  **public class** \_12 {   **private double**[] \_9\_12() {   Scanner cin = **new** Scanner(System.***in***);  **double** x1 = cin.nextDouble();  **double** x2 = cin.nextDouble();  **double** x3 = cin.nextDouble();  **double** x4 = cin.nextDouble();  **double** y1 = cin.nextDouble();  **double** y2 = cin.nextDouble();  **double** y3 = cin.nextDouble();  **double** y4 = cin.nextDouble();  **double** a = (y2 - y1) / (x2 - x1);  **double** b = -1;  **double** c = (y4 - y2) / (x4 - x3);  **double** e = -(a \* x1 - y1);  **double** d = -1;  **double** f = -(c \* x3 - y3);  **double**[] res = **new double**[2];  LinEq linEq = **new** LinEq(a, b, c, d, e, f);  res[0] = linEq.getX();  res[1] = linEq.getY();  **return** res;  }   **public static** Location locationLargest(**double**[][] a)  {  **int** x = 0, y = 0;  **double** MAX = a[0][0];  **for**(**int** i = 0; i < a.**length**; i++)  {  **for**(**int** j = 0; j < a[i].**length**; j++)  {  **if**(MAX < a[i][j])  {  MAX = a[i][j];  x = i + 1;  y = j + 1;  }  }  }  Location location = **new** Location();  location.**row** = x;  location.**column** = y;  location.**maxValue** = MAX;  **return** location;  }    **public void** \_10\_5(**int** val)  {  StackInteger stack = **new** StackInteger();  **for**(**int** i = 2; i \* i <= val; i++)  {  **while** (val % i == 0)  {  stack.push(i);  val /= i;  }  }  **if**(val > 1) stack.push(val);  }   **public void** main(String[] args) { *// QEquation equation = new QEquation(); // double[] res = new double[2]; // res = this.\_9\_12();* \_10\_5(100);  } }  **class** StackInteger {  **private int**[] **element**;  **private int size**;  **public static final int *DEEP*** = 100;    StackInteger(**int** capacity)  {  **size** = 0;  **element** = **new int**[capacity];  }    StackInteger()  {  **this**(***DEEP***);  }    **public void** push(**int** val)  {  **if**(**size** >= **element**.**length**)  {  **int**[] tmp = **new int**[**element**.**length** \* 2];  System.*arraycopy*(**element**, 0, tmp, 0, **element**.**length**);  **element** = tmp;  }  **element**[**size**++] = val;  }    **public int** getSize()  {  **return size**;  }    **public boolean** empty()  {  **return size** == 0;  }    **public int** pop()  {  **return element**[--**size**];  }     }  **class** MyPoint {  **private double x**;  **private double y**;   MyPoint(**double** x, **double** y) {  **this**.**x** = x;  **this**.**y** = y;  }   MyPoint()  {  **this**.**x** = 0;  **this**.**y** = 0;  }    **public double** distance(**double** x, **double** y)  {  **return** Math.*sqrt*((y - **this**.**y**)\*(y - **this**.**y**) + (x - **this**.**x**) \* ( x- **this**.**x**));  }  **public double** distance(MyPoint myPoint)  {  **double** y = myPoint.**y**;  **double** x = myPoint.**x**;  **return** Math.*sqrt*((y - **this**.**y**)\*(y - **this**.**y**) + (x - **this**.**x**) \* ( x- **this**.**x**));  }  }   **class** Location {  **public int row**;  **public int column**;  **public double maxValue**; }  **class** QEquation {  **private double a**;  **private double b**;  **private double c**;   QEquation()  {   }   QEquation(**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** getDiscri()  {  **return b** \* **b** - 4 \* **a** \* **c**;  }   **public double** getRoot1()  {  **if**(**this**.getDiscri() < 0)  {  **return** 0;  }  **else** {  **return** (-**b** + Math.*sqrt*(**this**.getDiscri())) / (2 \* **a**);  }  }  **public double** getRoot2()  {  **if**(**this**.getDiscri() < 0)  {  **return** 0;  }  **else** {  **return** (-**b** - Math.*sqrt*(**this**.getDiscri())) / (2 \* **a**);  }  } }  **class** LinEq {  **private double a**;  **private double b**;  **private double c**;  **private double d**;  **private double e**;  **private double f**;   **public** LinEq(**double** a, **double** b, **double** c, **double** d, **double** e, **double** f) {  **this**.**a** = a;  **this**.**b** = b;  **this**.**c** = c;  **this**.**d** = d;  **this**.**e** = e;  **this**.**f** = f;  }   **public boolean** isSlove()  {  **return** (**a** \* **d** - **b** \* **c** != 0);  }   **public double** getX()  {  **return** (**e** \* **d** - **b** \* **f**) / (**a** \* **d** - **b** \* **c**);  }  **public double** getY()  {  **return** (**a** \* **f** - **e** \* **c**) / (**a** \* **d** - **b** \* **c**);  }  **public double** getA() {  **return a**;  }   **public double** getB() {  **return b**;  }   **public double** getC() {  **return c**;  }   **public double** getD() {  **return d**;  }   **public double** getE() {  **return e**;  }   **public double** getF() {  **return f**;  }  } |
| 调  试  过  程  及  实  验  结  果 |  |
| 总  结 |  |
| 附  录 |  |