**北京科技大学java实验报告**

学院：计通学院 专业：计算机与科学技术系 班级：计1604

姓名：王鑫雨 学号：41601435

**第一次上机实验**

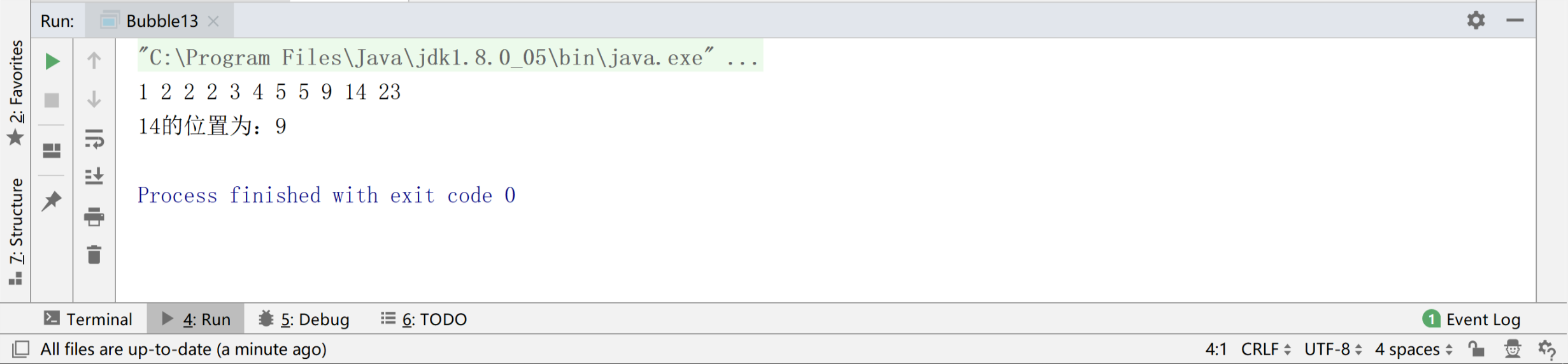
**题目：**

1.3 使用冒泡排序编写一个对整形数组进行排序的方法，使用二分查找法找到数组（已排好序）中的特定元素。

**源代码：**

|  |
| --- |
| **public class** Bubble13 {  *//冒泡排序* **void** bubble(**int** a[],**int** s,**int** e){  **for**(**int** i=s;i<e;i++){  **for**(**int** j=s;j<e-i;j++){  **if**(a[j]>a[j+1]){  **int** t=a[j];  a[j]=a[j+1];  a[j+1]=t;  }  }  }  }  *//二分查找,返回查找元素在数组中的位置。 -1代表无法查找* **int** find(**int** a[],**int** t){  **if**(a[0]>t||a[a.**length**-1]<t){  **return** -1;  }  **int** left=0,right=a.**length**-1;  **int** mid=(left+right)/2;  **while** (left<right){  **if**(a[mid]==t){  **return** mid;  }  **else if**(a[mid]>t){  right=mid-1;  mid=(left+right)/2;  }  **else**{  left=mid+1;  mid=(left+right)/2;  }  }  **return** -1;  }   **public static void** main(String[] args) {  **int** [] a={1,5,2,9,14,23,4,3,2,5,2};  Bubble13 b=**new** Bubble13();  *//sort* b.bubble(a,0,a.**length**-1);  *//out* **for**(**int** i=0;i<a.**length**;i++){  System.***out***.print(a[i]+**" "**);  }   *//find&out* System.***out***.println(**"\n14的位置为："**+b.find(a,14));  } } |

**结果：**



**过程中的问题与解决：**

问题：对java语言不熟悉，在过程中创建数组等一些基本操作不知如何使用。

解决:百度一下，问题解决。

**题目：**

1.4 多维数组的定义与使用：随机定义一定个数的空间点（x,y,z），编写程序找出距离最近的点对。

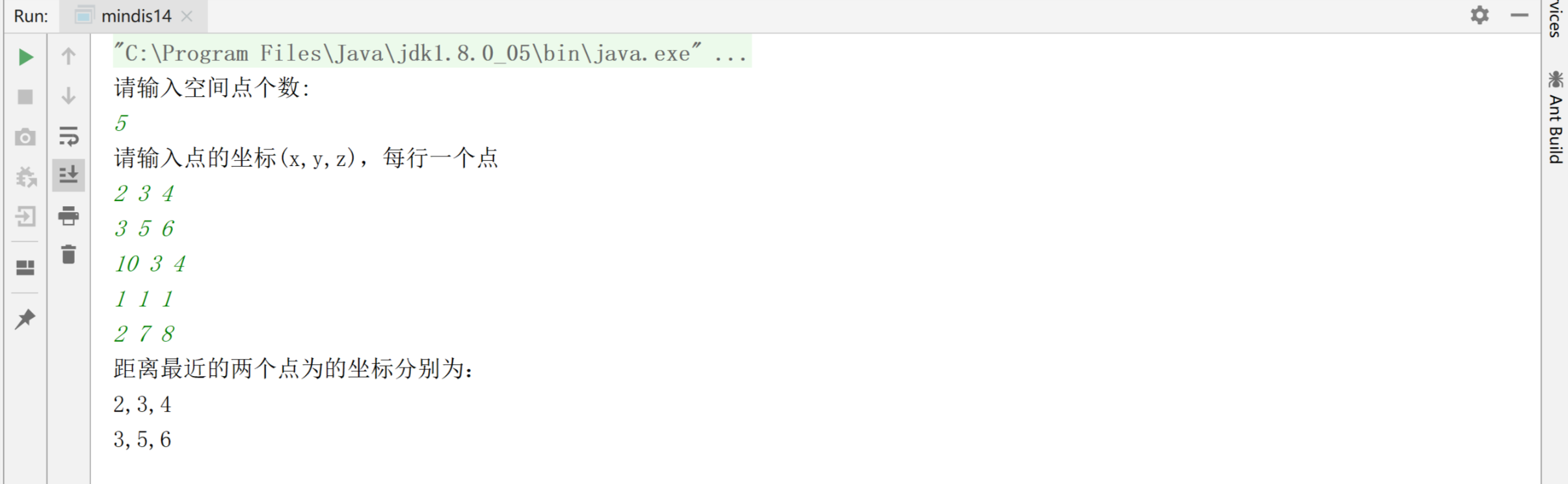
**源代码：**

|  |
| --- |
| **import** java.util.Scanner;  **public class** mindis14 {  **public static void** main(String[] args) {  Point p[]=**new** Point[100];  System.***out***.println(**"请输入空间点个数:"**);  **int** num=0;  *//输入声明* Scanner input=**new** Scanner(System.***in***);  num=input.nextInt();   System.***out***.println(**"请输入点的坐标(x,y,z)，每行一个点"**);  **for**(**int** i=0;i<num;i++){  **int** x,y,z;  x=input.nextInt();  y=input.nextInt();  z=input.nextInt();  Point t=**new** Point(x,y,z);  p[i]=t;  }  Point ans[]=**new** Point[2];  **int** sh=1000000000;  **for**(**int** i=0;i<num;i++){  **for**(**int** j=i+1;j<num;j++){  **int** a,b,c;  a= (**int**) (Math.*pow*((p[i].**x**-p[j].**x**),2));  b=(**int**) (Math.*pow*(p[i].**y**-p[j].**y**,2));  c=(**int**) (Math.*pow*(p[i].**z**-p[j].**z**,2));  **if**(sh>a+b+c){  ans[0]=p[i];  ans[1]=p[j];  sh=a+b+c;  }  }  }  System.***out***.println(**"距离最近的两个点为的坐标分别为："**);  System.***out***.println(ans[0].**x**+**","**+ans[0].**y**+**","**+ans[0].**z**);  System.***out***.println(ans[1].**x**+**","**+ans[1].**y**+**","**+ans[1].**z**);  } } |

创建的Point类：

|  |
| --- |
| **public class** Point {  **public  int x**,**y**,**z**;   Point(**int** x,**int** y,**int** z){  **this**.**x**=x;  **this**.**y**=y;  **this**.**z**=z;  } } |

**结果：**



**过程中的问题与解决：**

问题：不知道java该如何输入整形数据。

解决：利用Scanner类中的输入函数。

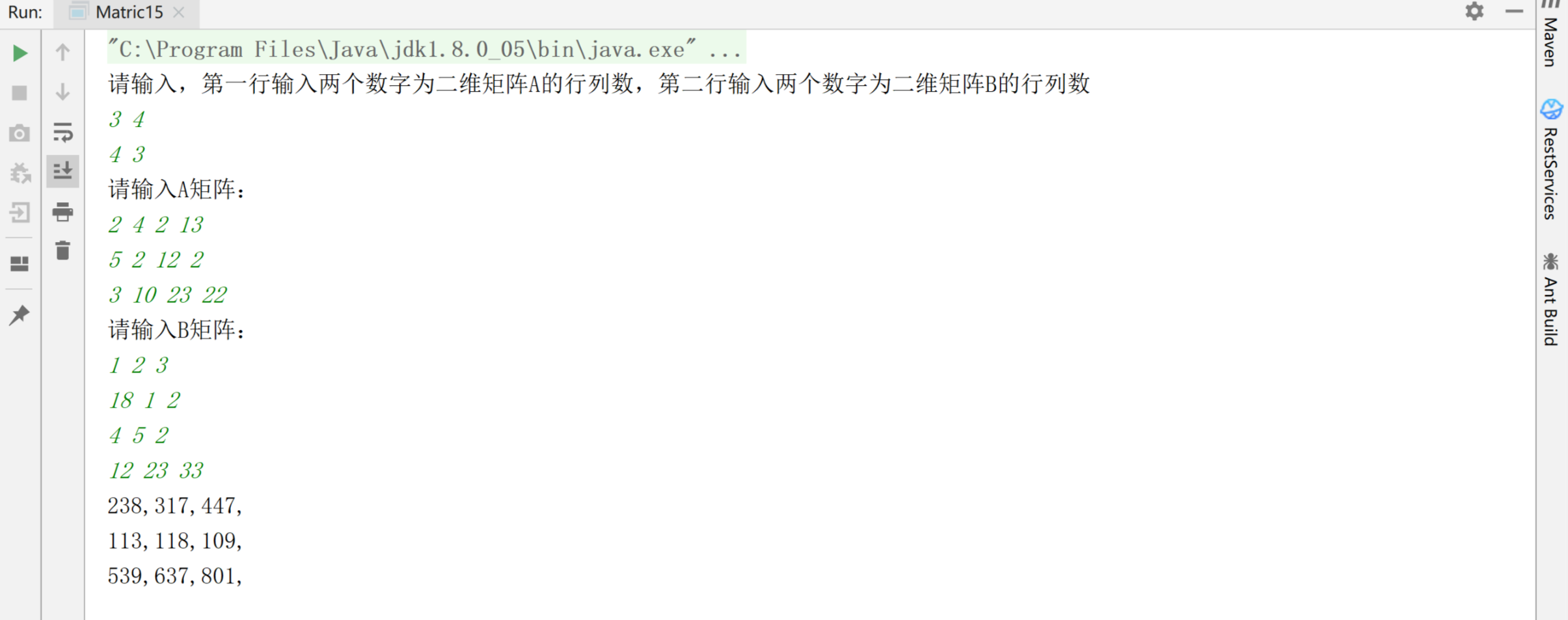
**题目：**

1.5 随机定义两个二维数组，构成两个矩阵，编写两个矩阵相乘的方法。

**源代码：**

|  |
| --- |
| **import** java.util.Scanner; **public class** Matric15 {  **void** multi(**int** a[][],**int** b[][],**int** i1,**int** j1,**int** i2,**int** j2){  **int** ans[][]=**new int**[100][100];  **if**(j1!=i2) System.***out***.println(**"矩阵无法相乘"**);  **else**{  **for**(**int** i=0;i<i1;i++){  **for**(**int** j=0;j<j2;j++){  **int** t=0;  **for**(**int** k=0;k<j1;k++){  t=t+a[i][k]\*b[k][j];  }  ans[i][j]=t;  System.***out***.print(t+**","**);  }  System.***out***.println();  }  }  }   **public static void** main(String[] args) {   **int** [][]a=**new int**[100][100];  **int** [][]b=**new int**[100][100];  System.***out***.println(**"请输入，第一行输入两个数字为二维矩阵A的行列数，第二行输入两个数字为二维矩阵B的行列数"**);  Scanner input=**new** Scanner(System.***in***);  **int** i1,j1,i2,j2;  i1=input.nextInt(); j1=input.nextInt(); i2=input.nextInt(); j2=input.nextInt();  System.***out***.println(**"请输入A矩阵："**);  **for**(**int** i=0;i<i1;i++){  **for**(**int** j=0;j<j1;j++){  a[i][j]=input.nextInt();  }  }  System.***out***.println(**"请输入B矩阵："**);  **for**(**int** i=0;i<i2;i++){  **for**(**int** j=0;j<j2;j++){  b[i][j]=input.nextInt();  }  }  **int** [][]ans=**new int**[100][100];  Matric15 m=**new** Matric15();  m.multi(a,b,i1,j1,i2,j2);  } } |

**结果：**



**过程中的问题与解决：**

问题：不知如何创建二维数组。

解决：利用int a[][]=new int[100][100];创建。

**题目：**

1.7 定义两个数组A和B，B数组中的元素包含在A数组中，请写一段代码把A数组中B没有的元素放到C数组中，假如数组中都是数字，而且已经按大小排序，请写一段代码最快效率把上面的元素放到C数组中。

**源代码：**

|  |
| --- |
| **public class** ABC17 {  **public static void** main(String[] args) {  **int** []a={1,2,3,4,5,6,12,24,100};  **int** []b={3,6,24};  **int** []c=**new int**[100];  *//时间复杂度为n,设两个指针i,j分别指向数组a,b;* **int** i=0,j=0;  **int** index=0;  **while**((i<=a.**length**-1)){  **if**(j>=b.**length**){  c[index++]=a[i];  i++;  }  **else if**(a[i]<b[j]){  c[index++]=a[i];  i++;  }  **else if**(a[i]==b[j]){  i++;  j++;  }   }  **for**(i=0;i<index;i++){  System.***out***.println(c[i]);  }  } } |

**结果：**

C数组的结果：



**过程中的问题与解决：**

问题：无

解决：无

**第二次上机实验**

**题目：**

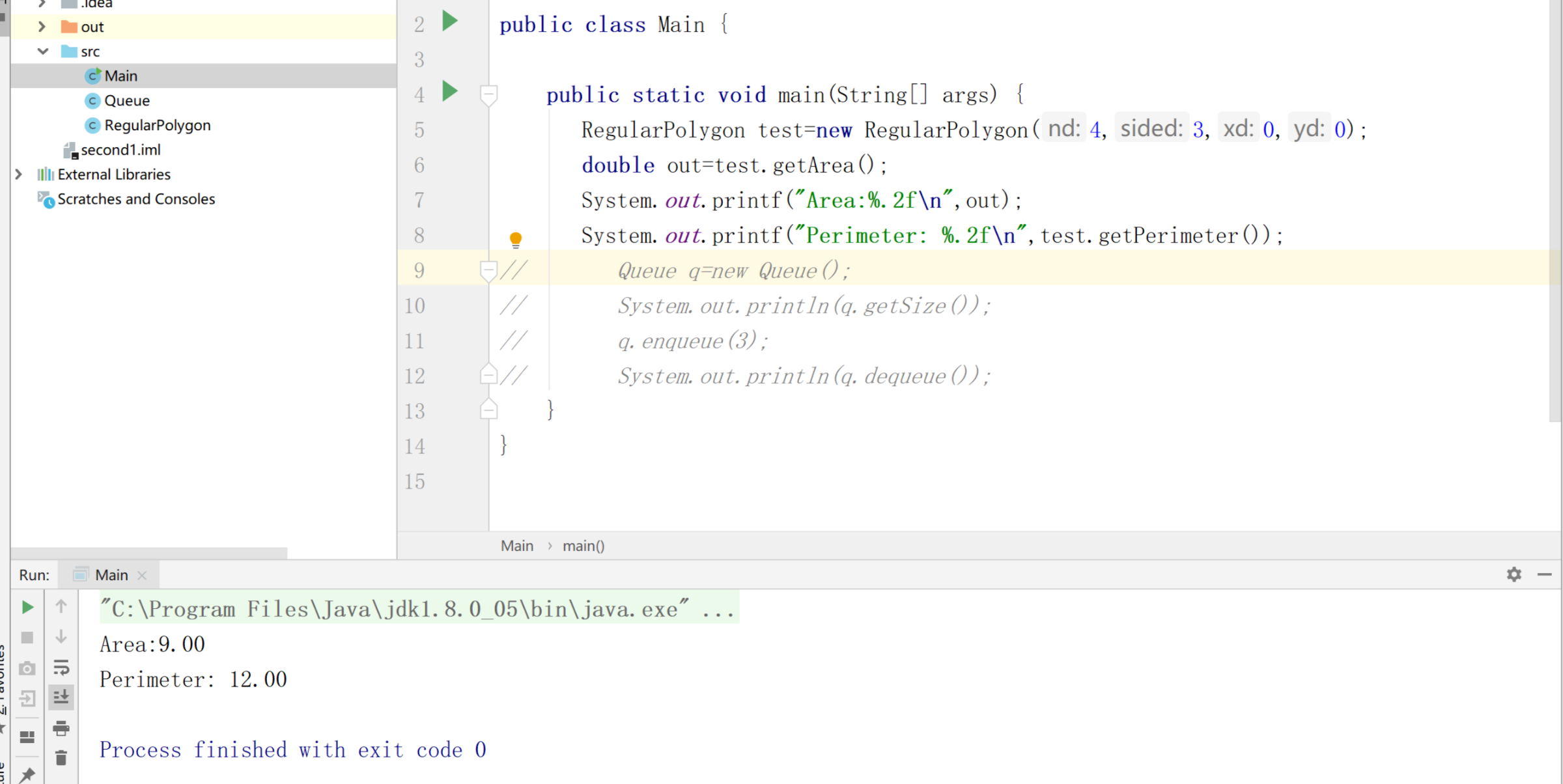
2.3在一个正n边形中，所有边的长度相等，所有角的角数都相等。创建一个名为RegularPolygon的类，该类包括：

1. 一个名为n的int型私有数据域定义多边形的边数，默认值为3；
2. 一个名为side的double型私有数据域存储边的长度，默认值为1；
3. 一个名为x的double型私有数据域定义多边形中点的x坐标，默认值为0；
4. 一个名为y的double型私有数据域定义多边形中点的y坐标，默认值为0；
5. 一个创建带默认值的正多边形的无参构造方法
6. 一个能创建带指定边数和边长度、中心在(0,0)的正多边形的构造方法
7. 一个能创建带指定边数和边长度、中心在(x,y)的正多边形的构造方法
8. 所有数据域的访问器和修改器
9. 一个返回多边形周长的方法getPerimeter()；
10. 一个返回多边形面积的方法getArea(); 面积=（n\*s2）/ [4\* tan(pi/n)]

**源代码：**

|  |
| --- |
| **public class** RegularPolygon {  **private int n**=3;*//边数* **private double side**=1;*//边长* **private double x**=0;  **private double y**=0;  **private double pi**=Math.*toRadians*(180);  *//所有数据域的访问器和修改器* **public void** setN(**int** n) {  **this**.**n** = n;  }   **public void** setSide(**double** side) {  **this**.**side** = side;  }   **public void** setX(**double** x) {  **this**.**x** = x;  }   **public void** setY(**double** y) {  **this**.**y** = y;  }   **public int** getN() {  **return n**;  }   **public double** getSide() {  **return side**;  }   **public double** getX() {  **return x**;  }   **public double** getY() {  **return y**;  }   RegularPolygon(){  }  RegularPolygon(**int** nd,**double** sided){  **n**=nd;  **side**=sided;  }  RegularPolygon(**int** nd,**double** sided,**double** xd,**double** yd){  **n**=nd;  **side**=sided;  **x**=xd;  **y**=yd;  }   *//一个返回多边形周长的方法getPerimeter()；* **public double** getPerimeter(){  **return n**\***side**;  }   *//返回多边形面积* **public double** getArea(){  **return n**\*Math.*pow*(**side**,2)/(4\*Math.*tan*(**pi**/**n**));  } } |

**结果：**



**过程中的问题与解决：**

问题：

不知道如何输出时输出保留小数后两位数字。

解决：

System.***out***.printf(**"Perimeter: %.2f\n"**,test.getPerimeter());

**题目：**

2.4设计一个队列类Queue，用于存储整数，在队列中，元素以“先进先出”的方式获取。该类包含：

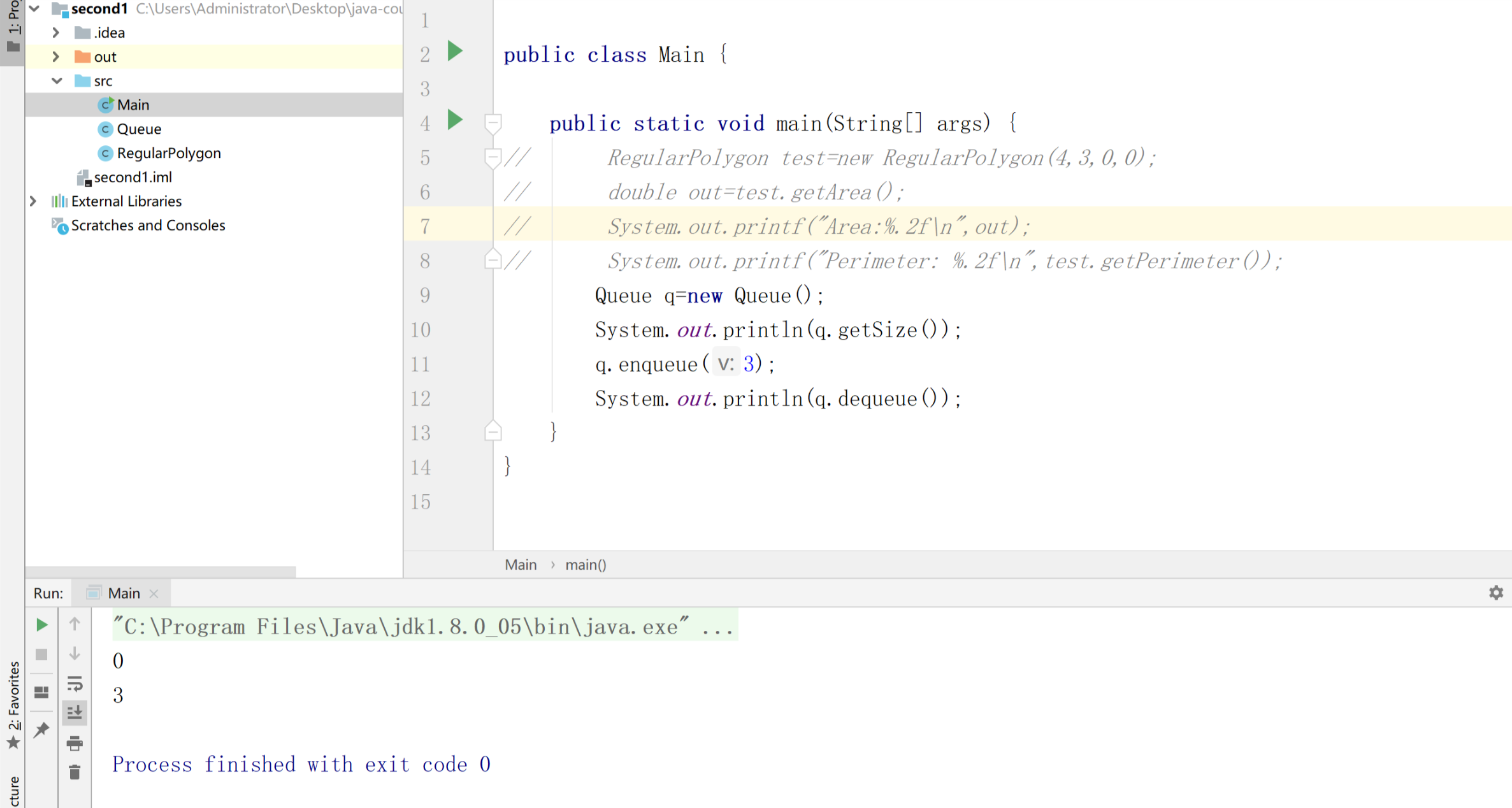
1. 一个名为element的int[]类型的数据域，保存队列中的int值
2. 一个名为size的数据域，保存队列中元素的个数
3. 一个构造方法，使用默认的容量8来创建一个Queue对象
4. 一个enqueue(int v)方法，用于将v加入队列中
5. 一个dequeue（）方法，用于从队列中移除元素并返回该元素
6. 方法isEmpty()，判断队列是否为空

方法getSize()，返回队列的大小。

**源代码：**

|  |
| --- |
| **public class** Queue {  **private int**[] **element**=**new int**[8];  **private int size**=0;  Queue(){  }  **public void** enqueue(**int** v){  **element**[**size**++]=v;  }  **public int** dequeue(){  **if**(**size**==0) **return** -1;  **int** t;  t=**element**[--**size**];  **return** t;  }   **public boolean** isEmpty(){  **if**(**size**==0) **return true**;  **else return false**;  }   **public int** getSize(){  **return size**;  } } |

**结果：**



**过程中的问题与解决：**

问题：

不懂得如何创建数组。

解决：

**private int**[] **element**=**new int**[8];

**第三次上机实验**

**题目：**

3.1设计一个名为Person的类和它的两个名为Student和Employee的子类，Employee又有子类：教员类Faulty和职员类Staff。每个人都有姓名、地址、电话号码和电子邮箱地址。学生有班级状态（大一、大二、大三、大四）。将这些状态定义为常量。一个雇员有办公室、工资和受聘日期（定义一个MyDate类为受聘日期创建一个对象）。教员有办公时间和级别。职员有职务称号。覆盖每个类的toString()方法，利用多态机制显示相应的类别名字和人名。

**源代码：**

|  |
| --- |
| //Person  **public class** Person {  **private** String **name**;  **private** String **address**;  **private** String **phone\_number**;  **private** String **mail\_address**;   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public** String getAddress() {  **return address**;  }   **public void** setAddress(String address) {  **this**.**address** = address;  }   **public** String getPhone\_number() {  **return phone\_number**;  }   **public void** setPhone\_number(String phone\_number) {  **this**.**phone\_number** = phone\_number;  }   **public** String getMail\_address() {  **return mail\_address**;  }   **public void** setMail\_address(String mail\_address) {  **this**.**mail\_address** = mail\_address;  }   **public** Person(String name, String address, String phone\_number, String mail\_address) {  **this**.**name** = name;  **this**.**address** = address;  **this**.**phone\_number** = phone\_number;  **this**.**mail\_address** = mail\_address;  }   @Override  **public** String toString() {  **return "Person{"** +  **"人名='"** + **name** + **'\''** +  **" 类别名=Person"** +  **'}'**;  } }  //Student  **public class** Student **extends** Person {  **final private** String **status**;   @Override  **public** String toString() {  **return "Student{"** +  **"name='"** + **this**.getName() + **'\''** +  **"类别名：Staff"**+  **'}'**;  }   **public** String getStatus() {  **return status**;  }   **public** Student(String name, String address, String phone\_number, String mail\_address, String status) {  **super**(name, address, phone\_number, mail\_address);  **this**.**status** = status;  } }  //Employee  **public class** Employee **extends** Person {  **private** String **room**;  **private long salary**;   @Override  **public** String toString() {  **return "Employee{"** +  **"name='"** + **this**.getName() + **'\''** +  **"类别名：Staff"**+  **'}'**;  }   **public** String getRoom() {  **return room**;  }   **public void** setRoom(String room) {  **this**.**room** = room;  }   **public long** getSalary() {  **return salary**;  }   **public void** setSalary(**long** salary) {  **this**.**salary** = salary;  }   **public** MyDate getSday() {  **return sday**;  }   **public void** setSday(MyDate sday) {  **this**.**sday** = sday;  }   MyDate **sday**;   **public** Employee(String name, String address, String phone\_number, String mail\_address, String room, **long** salary, MyDate sday) {  **super**(name, address, phone\_number, mail\_address);  **this**.**room** = room;  **this**.**salary** = salary;  **this**.**sday** = sday;  } }  //Faulty  **public class** Faulty **extends** Employee {  **private** String **workTime**;   @Override  **public** String toString() {  **return "Faulty{"** +  **"name='"** + **this**.getName() + **'\''** +  **"类别名：Faulty"**+  **'}'**;  }   **public** String getWorkTime() {  **return workTime**;  }   **public void** setWorkTime(String workTime) {  **this**.**workTime** = workTime;  }   **public int** getRank() {  **return rank**;  }   **public void** setRank(**int** rank) {  **this**.**rank** = rank;  }   **private int rank**;   **public** Faulty(String name, String address, String phone\_number, String mail\_address, String room, **long** salary, MyDate sday, String workTime, **int** rank) {  **super**(name, address, phone\_number, mail\_address, room, salary, sday);  **this**.workTime = workTime;  **this**.rank = rank;  } }  //Staff  **public class** Staff **extends** Employee {  **private** String **position\_title**;   **public** String getPosition\_title() {  **return position\_title**;  }   **public void** setPosition\_title(String position\_title) {  **this**.**position\_title** = position\_title;  }   @Override  **public** String toString() {  **return "Staff{"** +  **"name='"** + **this**.getName() + **'\''** +  **"类别名：Staff"**+  **'}'**;  }   **public** Staff(String name, String address, String phone\_number, String mail\_address, String room, **long** salary, MyDate sday, String position\_title) {  **super**(name, address, phone\_number, mail\_address, room, salary, sday);  **this**.**position\_title** = position\_title;  } }  //主函数  **public class** Main {   **public static void** main(String[] args) {  Person p=**new** Person(**"pidan"**,**"ustb"**,**"1565278"**,**"pi@163"**);  MyDate d=**new** MyDate(**new** Date());  Staff s=**new** Staff(**"pidan"**,**"ustb"**,**"1565278"**,**"pi@163"**,**"723"**,123456,d,**"professor"**);  System.***out***.println(p.toString());  System.***out***.println(s.toString());   } } |

**结果：**



**过程中的问题与解决：**

问题：

对toString函数不了解，开始时没有思绪。

解决：

翻阅课件查阅资料了解后完成了实验。

**题目：**

3.3使用下面的方法头编写方法，从一个整数的数组列表中去掉重复元素(如果有重复元素，只保留其中一个值)：  
 public static void removeDuplicate(ArrayList<Integer> list)

**源代码：**

|  |
| --- |
| **import** java.util.ArrayList; **import** java.util.HashSet; **import** java.util.Set; **public class** removeDuplicate {  **static public void** removeDuplicate(ArrayList<Integer> list){  Set<Integer> s = **new** HashSet<>();  **for**(**int** i=0;i<list.size();i++){  **int** t=list.get(i);  **if**(s.contains(t)){  list.remove(i);  i=i-1;  }  **else** {  s.add(t);  }  }  } }  //主函数  **public class** Main {  **public static void** main(String[] args) {  ArrayList<Integer> l=**new** ArrayList<>();  l.add(3);  l.add(3);  l.add(5);  l.add(3);  l.add(9);  l.add(9);  l.add(6);  l.add(30);  l.add(30);  removeDuplicate.*removeDuplicate*(l);  System.***out***.println(l);  } } |

**结果：**



**过程中的问题与解决：**

问题：不知道ArrayList中不能放int。

解决：查阅资料后，了解到，ArrayList只能存储对象，不能存放原生数据类型数据。故应该使用Integer。

**第四次上机实验**

**题目：**

4.2 使用容器来模拟枪射击。

（1）定义子弹类：Ball, 有属性：编号、型号。

默认构造方法，设编号为1， 型号为“M54”。

带参数构造方法：设定编号和型号。

方法1：取得编号

方法2：取得型号。

方法3：显示子弹的编号和型号： 编号-型号。

（2）定义枪类：Gun

有属性：编号，型号，弹夹，装弹数量。

默认构造方法：创建编号为1，型号为“M54”，装弹数量为6。

带参数构造方法：与属性相同的参数，设定属性：编号，型号，数量的值。

装弹方法：将一个子弹对象装入弹夹， 如果弹夹已满，抛出弹夹满异常。

射击方法：返回一个子弹对象，减少弹夹的一颗子弹，如果弹夹空，则抛出空异常。

显示弹夹内子弹列表信息方法：显示弹夹内的子弹信息：编号-型号

取得当前弹夹子弹数的方法。

（3）编写带main方法的测试类，模拟枪的装弹和射击功能。

**源代码：**

|  |
| --- |
| //子弹  **public class** Ball {  **private int id**;  **private** String **Type**;   **public int** getId() {  **return id**;  }   **public void** setId(**int** id) {  **this**.**id** = id;  }   **public** String getType() {  **return Type**;  }   **public void** setType(String type) {  **Type** = type;  }   **public** Ball() {  **id**=1;  **Type**=**"M54"**;  }   **public** Ball(**int** id, String type) {  **this**.**id** = id;  **Type** = type;  }   @Override  **public** String toString() {  **return id**+**"-"**+**Type**;  } }  //枪  **import** java.util.ArrayList; **import** java.util.List;  **public class** Gun {  **private int id**;  **private** String **Type**;  **private int nums**;  **private** List<Ball> **clip** = **new** ArrayList();  **static int** *ID*=1;  **public** Gun() {  **id**=1;  **Type**=**"M54"**;  **nums**=6;  }   **public** Gun(**int** id, String type, **int** nums) {  **this**.**id** = id;  **Type** = type;  **this**.**nums** = nums;  }  *//装弹* **public void** load() **throws** ClipError{  **if**(clip.size()>=nums){  **throw new** ClipError(**"弹夹已满，无法继续装弹！"**);  }  **else**{  clip.add(**new** Ball(ID++,Type));  }  }  *//射击* **public** Ball shoot() **throws** ClipError{  **if**(clip.size()==0){  **throw new** ClipError(**"弹夹空，无法再射！"**);  }  **else** {  Ball ans=clip.get(clip.size()-1);  clip.remove(clip.size()-1);  ID--;  **return** ans;  }  }  *//显示所有弹夹的信息* **public void** showBalls(){  **if**(**clip**.isEmpty()){  System.***out***.println(**"弹夹为空"**);  }  **else** {  **for**(**int** i=0;i<**clip**.size();i++){  System.***out***.println(**clip**.get(i).toString());  }  }  }    *//取得子弹数* **private int** getBallNums(){  **return clip**.size();  } }  //自定义违例类  **public class** ClipError **extends** Exception{  *//无参构造方法* **public** ClipError(){  **super**();  }   *//用指定原因构造一个新的异常* **public** ClipError(Throwable cause) {  **super**(cause);  }  *//有参的构造方法* **public** ClipError(String message){  **super**(message);  } }  //主函数  **public class** Main {   **public static void** main(String[] args) **throws** IllegalTriangleException {  Gun g=**new** Gun();  *//装弹7次* **for**(**int** i=0;i<7;i++){  **try**{  g.load();  }  **catch** (ClipError e){  e.printStackTrace();  }  }   *//射击8次* **for**(**int** i=0;i<8;i++){  **try**{  g.shoot();  }  **catch** (ClipError e){  e.printStackTrace();  }  }   } } |

**结果：**

分别让装弹七次（会发生装满无法再装异常）和射击八次（会发生无子弹异常）。



**过程中的问题与解决：**

问题：对自定义违例不了解。

解决：进行学习后有了一定的了解。但仍不清楚其真正的具体用途，可能以后代码经验丰富了就会有所了解。

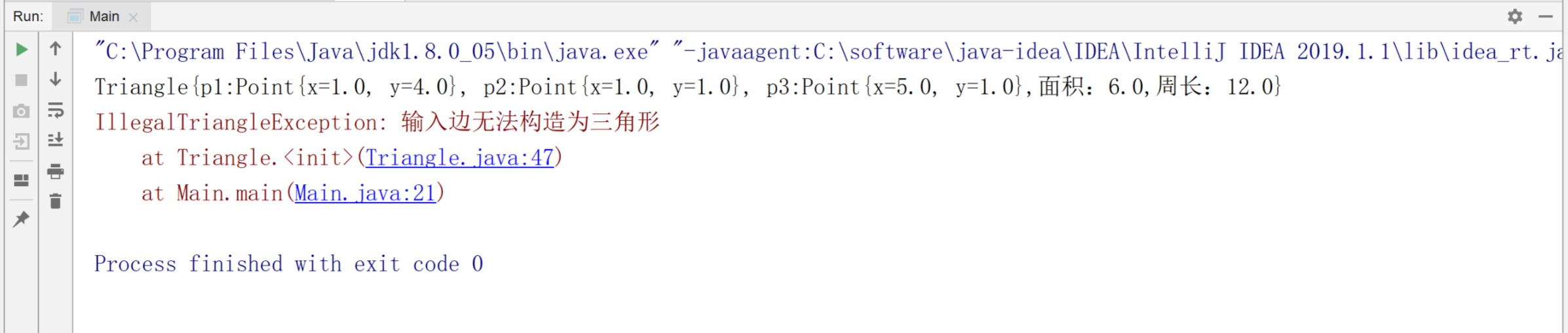
**题目：**

4.3设计一个名为Triangle类，该类包括三个名为point1，point2和point3的点对，表示二维平面三角形的三个顶点，带参数构造方法，所有数据域的访问器方法，一个名为getArea()方法返回三角形面积，一个getPerimeter()方法返回三角形周长，一个名为toString()方法返回这个三角形的字符串描述。在三角形中任意两边之和大于第三边，创建一个IllegalTriangleException类，然后修改Triangle类的构造方法，如果创建的三角形的边违反这一规则，抛出一个IllegalTriangleException对象。

**源代码：**

|  |
| --- |
| //Triangle  **public class** Triangle {  **private** Point **p1**;  **private** Point **p2**;  **private** Point **p3**;  **private double a**,**b**,**c**;  **public** Point getP1() {  **return p1**;  }   **public void** setP1(Point p1) {  **this**.**p1** = p1;  }   **public** Point getP2() {  **return p2**;  }   **public void** setP2(Point p2) {  **this**.**p2** = p2;  }   **public** Point getP3() {  **return p3**;  }   **public void** setP3(Point p3) {  **this**.**p3** = p3;  }    **public double** dis(Point a,Point b){  **return** Math.*pow*(Math.*pow*(a.getX()-b.getX(),2)+Math.*pow*(a.getY()-b.getY(),2),0.5);  }   **public** Triangle(Point p1, Point p2, Point p3) **throws** IllegalTriangleException{  **this**.**p1** = p1;  **this**.**p2** = p2;  **this**.**p3** = p3;  **this**.**a**=dis(p1,p2);  **this**.**b**=dis(p1,p3);  **this**.**c**=dis(p2,p3);  **if**(p1.equals(p2)||p1.equals(p3)||p2.equals(p3)){  **throw new** IllegalTriangleException(**"输入边无法构造为三角形"**);  }   **if**(p1.getX()==p2.getX()&&p1.getX()==p3.getX()){  **throw new** IllegalTriangleException(**"输入边无法构造为三角形"**);  }  **if**(p1.getY()==p2.getY()&&p1.getY()==p3.getY()){  **throw new** IllegalTriangleException(**"输入边无法构造为三角形"**);  }  }   @Override  **public** String toString() {  **return "Triangle{"** +  **"p1:"** + **p1**.toString() +  **", p2:"** + **p2**.toString() +  **", p3:"** + **p3**.toString() +  **",面积："**+**this**.getArea()+  **",周长："**+**this**.getPerimeter()+  **'}'**;  }   **public double** getArea(){  **double** p=getPerimeter()/2;  Double s=Math.*pow*(p\*(p-**a**)\*(p-**b**)\*(p-**c**),0.5);  **return** s;  }   **public double** getPerimeter(){  **return a**+**b**+**c**;  } }  //Point  **public class** Point {  **private double x**;  **private double y**;   **public** Point(**double** x, **double** y) {  **this**.**x** = x;  **this**.**y** = y;  }   **public double** getX() {  **return x**;  }   **public void** setX(**double** x) {  **this**.**x** = x;  }   **public double** getY() {  **return y**;  }   **public void** setY(**double** y) {  **this**.**y** = y;  }   @Override  **public** String toString() {  **return "Point{"** +  **"x="** + **x** +  **", y="** + **y** +  **'}'**;  }   **public boolean** equals(Point obj) {  **if**(**this**.getX()==obj.getX()&&**this**.getY()==obj.getY()){  **return true**;  }  **else**{  **return false**;  }  } }  //自定义违例类  **public class** IllegalTriangleException **extends** Exception {  **public** IllegalTriangleException(String message) {  **super**(message);  } }  //主函数  **public class** Main {   **public static void** main(String[] args){  *//三角形测试* Point a=**new** Point(1,4);  Point b=**new** Point(1,1);  Point c=**new** Point(5,1);  Triangle t= **null**;  **try** {  t = **new** Triangle(a,b,c);  System.***out***.println(t.toString());  } **catch** (IllegalTriangleException e) {  e.printStackTrace();  }    Point aa=**new** Point(1,3);  Point bb=**new** Point(1,1);  Point cc=**new** Point(1,10);  **try**{  Triangle tt=**new** Triangle(aa,bb,cc);  System.***out***.println(tt.toString());  }  **catch** (IllegalTriangleException e){  e.printStackTrace();  }  } } |

**结果：**



**过程中的问题与解决：**

问题：判断三个点是否可以构造成三角形比较复杂，两边之和大于第三边不能作为判别依据。

解决：三个点在一条直线上则无法构成三角形，三个点中有点重合则无法构成三角形。

**第五次上机实验**

**题目：**

5.1创建一个窗体，选择合适的布局管理器，显示一个按钮“查询”，点击该按钮以后，查询university数据库中的student表（包括学号，姓名，性别，年龄，班级，来自省份等字段）中的信息，并将查询结果返回给窗体，显示在窗体的下拉列表（或文本框）中；在窗体中显示一个按钮“注册”，以及可以填写学生各项信息的输入框，输入学生各项信息后，点击注册按钮，程序判断该学生是否已存在于数据库中，如果不存在，则注册到数据库，并显示注册成功，如果存在，则提示注册失败。

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：

**第六次上机实验**

**题目：**

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：

**题目：**

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：

**第七次上机实验**

**题目：**

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：

**题目：**

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：

**第八次上机实验**

**题目：**

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：

**题目：**

**源代码：**

|  |
| --- |
|  |

**结果：**

**过程中的问题与解决：**

问题：

解决：