**《面向对象程序设计(Java)》**

**实验报告**

姓 名

学 号

学 院 软件工程

专 业 软件工程

年 级 2020级

指导教师

**成都信息工程大学软件工程学院**

**2021年9月**

# 实验1：Java程序的编写、编译与运行

1. **实验目的**

1. 掌握下载Java SDK 软件包的方法；掌握Myeclipse或Eclipse或JetBrains IntelliJ IDEA等开发平台的下载安装方法

2. 掌握设置Java 程序运行环境的方法；

3. 掌握编写、编译、运行简单的Java Application程序的方法步骤，初步了解java程序的特点和基本结构。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 IntelliJ IDEA2020.2

1. **实验内容及指导**

1. 编写一个Java Application,利用JDK软件包中的工具编译并运行这个程序，使之能够输出字符串信息“Hello World! ”。

**实验步骤：**

①打开IDEA，新建一个project，设置好编译环境；

②新建一个Module，命名为JavaTest，在此module的src目录下新建一个Package取名为demo01。新建一个HelloWorld.java文件。

③编写程序，关键代码码如下：

|  |
| --- |
| public class HelloWorld {      public static void main(String[] args) {          System.out.println("HelloWorld!");      }  } |

④运行程序。

2. 编写一个应用程序求三个整数的平均数，原始数字要求从命令行输入。应用程序中main方法的参数String类型的数组args能接受用户从命令行键入的参数。

**实验步骤：**

(1)、创建类，在编辑器种编写出程序关键代码:

|  |
| --- |
| public static int AverAge(int[] nums) {      int sum = 0;  *for* (int i = 0; i < nums.length; i++) {          sum += nums[i];      }  *return* sum/nums.length;  } |

(2)编译程序、运行程序。

**参考程序运行效果：**

1. **实验总结**

本次实验完成了以下任务：

* 下载安装JDK
* 编写helloWorld.java
* 编写AverageTest.java
* 使用命令行编译运行java程序

掌握了JDK以及开发平台的安装方法，掌握设置java程序运行环境的方法

掌握编写、编译、运行简单java程序的方法步骤，初步了解java程序的特点和基本结构。

1. **附录（程序代码）**

|  |
| --- |
| // helloWorld.java  **public** **class** helloWorld {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  System.***out***.println("Hello World!");  }  }  // AverageTest.java  **public** **class** AverageTest {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **double** array[];  array = **new** **double**[args.length];  **double** sum = 0;  **for**(**int** i = 0; i < args.length; i++) {  array[i] = Integer.*valueOf*(args[i]);  sum += array[i];  }  **double** average = sum/args.length;  System.***out***.println(average);  }  } |

# 实验2：Java程序设计基础

1. **实验目的**

1. 掌握 Java 基本数据类型、运算符与表达式。

2. 理解 Java 程序语法结构，掌握顺序结构、选择结构和循环结构语法的程序设计方法。

3. 理解并掌握数组的使用

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 JetBrains IntelliJ IDEA2020.2

1. **实验内容及要求**

1. 编写一个应用程序，接受用户从键盘输入的摄氏温度，然后将其转化成等效的华氏度温度

(1) 设计思路

本题要求接收用户的输入作为摄氏温度，再将其转化为华氏温度。接收用户的输入需要用到Scanner类中的nextFloat()方法，进行温度换算时需要注意数据类型的变化（从double强制转换为float）

(2)实验步骤

①、学习Scanner类的数值读取方法

②、了解摄氏度与华氏度换算方法

③、编写代码

④、调试运行

(3) 关键代码

|  |
| --- |
| public class TemCompute {  public static void main(String[] args) {  Scanner sc1 = new Scanner(System.*in*);   System.*out*.print("Please input your temperature: ");   double in = sc1.nextDouble();  System.*out*.print("Your Fahrenheit is: ");  System.*out*.println(1.8 \* in + 32);  } } |

2. 用选择排序法，按升序顺序输出一维数组中的元素 。

(1) 设计思路

本题要求使用选择排序为数组排序并且输出数组元素，事先初始化一个一维数组，再使用排序算法排序，输出。

(2)实验步骤

①、学习选择排序

②、编写代码

③、编译运行

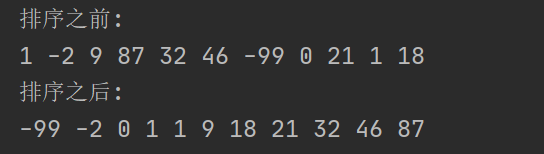
(3) 关键代码

|  |
| --- |
| **//选择排序  for(int i = 0; i < numbers.length - 1; i++){ //做第i趟排序  int k = i;  for(int j = k + 1; j < numbers.length; j++){//选最小的记录  if(numbers[j] < numbers[k]){  k = j;  }  }  if(i!=k) {//交换a[i]和a[k]  int temp = numbers[i];  numbers[i]=numbers[k];  numbers[k]=temp;  }  }  System.*out*.println();  System.*out*.println("排序之后:");  for(int num : numbers){  System.*out*.print(num + " ");  }  }** |

(4) 程序测试过程

输出排序前后数组内元素排列情况，方便比较

(5) 运行结果（运行效果）



3. (1) 设计思路

要求输出斐波那契数列，根据该数列的特点可使用数组存放数列，给数组前两个元素赋初值，使用数组存放数列，之后再次使用数列更快速便捷，但是由于需要先给两个元素赋值，数组的长度至少要为2。或者使用递归函数，将n = 1和n=2设置为递归出口，使用递归调用时主程序代码可读性更强，但是由于递归调用时有许多重复项出现，时间效率相比于数组存放更低。

(2)实验步骤

①、编写代码

②、编译运行

(3) 关键代码（至少用2种方式实现）

1. 递归实现

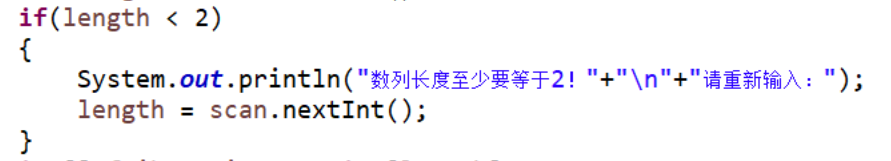
|  |
| --- |
| public static int Fibonacci(int term) {  *if*(term == 1 || term == 2) {  *return* 1;      } *else* {  *return* Fibonacci(term - 1) + Fibonacci(term - 2);      }  } |

1. 循环实现

|  |
| --- |
| *if* (n < 1) {      System.out.println("input Error!");  } *else* *if* (n == 1 ) {      System.out.println("1");  } *else* *if* (n == 2 ) {      System.out.println("1 1");  } *else* {      long a =1l, b= 1l, c =0l;      System.out.printf("%d %d",1,1);  *for* (int i = 0; i < n - 2; i++) {          c = a + b;       *//第3个数的值等于前两个数的和*          a = b;       *//第2个数的值赋值给第1个数*          b = c;       *//第3个数的值赋值给第2个数*          System.out.printf(" %d",c);      }  } |

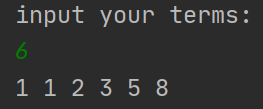
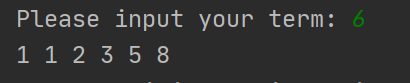
(4) 程序测试过程

当使用数组来存放数列时，数列的长度至少要为2（初始化时，给了两个值，数组长度至少为2），增加了判断语句。



(5) 运行结果（运行效果）

左：循环实现 右：递归实现



**五、 实验总结**

1. 掌握 Java 基本数据类型、运算符与表达式。

2. 理解 Java 程序语法结构，掌握顺序结构、选择结构和循环结构语法的程序设计方法。

3. 理解并掌握数组的使用

4.了解了Scanner类

**六、 附录（程序代码）**

|  |
| --- |
| **import** java.util.Scanner;  /\*\*  \* @author SaraiNoQ  \* @create 2021/9/16 - 19:26  \*/  **public** **class** Temperature {  **public** **static** **void** main(String[] args) {  // 华氏度 = 32°F（华氏温标单位）+ 摄氏度 × 1.8  System.***out***.println("------------------------------------");  Scanner scan = **new** Scanner(System.***in***);  System.***out***.println("请输入摄氏温度：");  **float** centigrade = scan.nextFloat();  **float** fahrenheit = (**float**)(32 + centigrade \* 1.8);  System.***out***.println();  System.***out***.println(centigrade+"摄氏度等于"+fahrenheit+"华氏度");  System.***out***.println("------------------------------------");  **int**[] array = {4, 7, 8, 2, 1, 0, 5, 10};  // 选择排序  System.***out***.println("排序前："+"\n");  **for**(**int** a : array)  System.***out***.println(a);  **for**(**int** i = 0; i < array.length-1; i++) {  **int** curmin = i;  **for**(**int** j = curmin + 1; j < array.length; j++) {  **if**(array[j] < array[curmin])  curmin = j;  }  **if**(i != curmin) {  **int** temp = array[i];  array[i] = array[curmin];  array[curmin] = temp;  }  }  System.***out***.println("排序后："+"\n");  // for each循环 输出数组  **for**(**int** a : array)  System.***out***.println(a);  System.***out***.println("------------------------------------");    System.***out***.println("请输入斐波那契数列长度(length>=2)：");  **int** length = scan.nextInt();  **if**(length < 2)  {  System.***out***.println("数列长度至少要等于2！"+"\n"+"请重新输入：");  length = scan.nextInt();  }  System.***out***.println();  **int**[] feibonaqi = **new** **int**[length];  feibonaqi[0] = feibonaqi[1] = 1;  **for**(**int** i = 0; i < length; i++)  {  **if**(i > 1)  feibonaqi[i] = feibonaqi[i-2] + feibonaqi[i-1];  System.***out***.println(feibonaqi[i]);  }  System.***out***.println("------------------------------------");    System.***out***.println("使用递归计算斐波那契数列：");  System.***out***.println("请输入斐波那契数列长度：");  **int** newlength = scan.nextInt();  System.***out***.println();  **for**(**int** i = 1; i <= newlength; i++)  {  System.***out***.println(*fib*(i));  }    }    **public** **static** **int** fib(**int** num) {  //判断：是否是第一个数和第二个数  **if**(num == 1 || num == 2) {  **return** 1;  }**else** {  //循环调用本函数  **return** *fib*(num - 2) + *fib*(num - 1);  }  }  } |

# 实验3： Java的面向对象特性（1）

1. **实验目的**

1. 通过编程和上机实验理解 Java 语言是如何体现面向对象编程基本思想，了解类的封装方法，以及 如何创建类和对象，了解成员变量和成员方法的特性，掌握 OOP 方式进行程序设计的方法 。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 JetBrains IntelliJ IDEA2020.2

1. **设计思路与具体实现**

1. 题目要求设计圆环类，它具有三个成员变量以及访问成员变量的方法，还有一个返回圆环面积的方法。

（1）设计思路

为避免成员变量在外部被修改，使用private将成员变量封装在类内部，避免数据的“交叉感染”。

（2）实验步骤

①编写程序

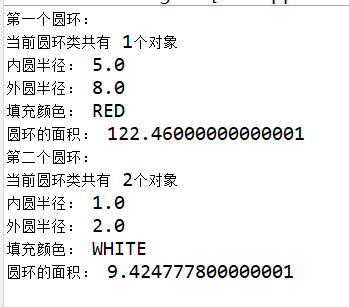
②编译、运行程序

（3）关键代码

|  |
| --- |
| public double computeArea() {  return Math.*PI* \* (outerRadius \* outerRadius - innerRadius \* innerRadius); } |

（4）程序测试过程

（5）运行结果（运行效果）



2.（1）设计思路

第1题中增加两个static 成员 ：圆周率和圆对象个数, 增加两个 static 方法，分别是设置圆周率和显示当前圆对象个数的, 仔细体会静态成员与实例成员的使用方法和区别

（2）实验步骤

①编写程序

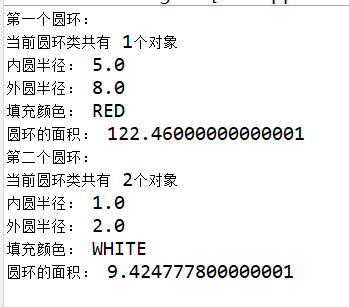
②编译、运行程序

（3）关键代码

|  |
| --- |
| public void setPi(double pi) {  this.*pi* = pi; } public void showObjectNo() {  System.*out*.println("objectNo: "+*objectNo*); } |

（4）程序测试过程

（5）运行结果（运行效果）



3. 题目要求设计一个教材类，一个课程类，及一个为某门课程指定参考教材的类。一门课程可以有多本参考教材，再设计一个测试类。

（1）设计思路

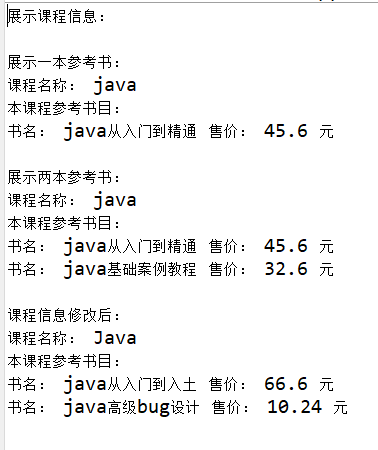
首先设计每个类都有什么成员变量和方法，如教材类应该有教材名称和价格，应该能够显示教材信息，修改教材信息。运用组合思想，在课程类中定义教材类型的成员变量，在类中设置访问成员变量的方法。通过方法的重载来实现显示不同的信息。

（2）实验步骤

①编写程序

②编译、运行程序

（3）运行结果（运行效果）



1. **实验总结**

通过本次实验，了解了类的封装方法，掌握如何创建类和对象，了解成员变量和成员方法的特性，掌握 OOP 方式进行程序设计的方法 。

在设计类的过程中，逐渐理解面向对象的编程思想。

1. **附录（程序代码）**

|  |
| --- |
| // 1，2题代码  **public** **class** RingTest {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Ring ring1 = **new** Ring(5, 8, "RED");  Ring.*count* += 1;  System.***out***.println("第一个圆环：");  Ring.*showCount*();  ring1.showInfo();  System.***out***.println("圆环的面积： " + ring1.getArea());  Ring ring2 = **new** Ring();  Ring.*count* += 1;  Ring.*setPI*(3.1415926);  System.***out***.println("第二个圆环：");  Ring.*showCount*();  ring2.showInfo();  System.***out***.println("圆环的面积： " + ring2.getArea());  }  }  **public** **class** Ring {  **private** **double** innerRadius;  **private** **double** outerRadius;  **private** String color;  // static修饰属性，使属性变为公共属性，只会分配一块内存空间,对一个对象修改就修改了全部  // 静态方法，在不创建对象的情况下调用该方法  **static** **double** *PI* = 3.14;  **static** **int** *count* = 0;  // 构造方法，用于创建圆环对象  Ring(**double** iRadius, **double** oRadius, String color){  innerRadius = iRadius;  outerRadius = oRadius;  **this**.color = color;  }  Ring(){  innerRadius = 1;  outerRadius = 2;  color = "WHITE";  }  // 显示圆环信息  **public** **void** showInfo() {  System.***out***.println("内圆半径： " + innerRadius);  System.***out***.println("外圆半径： " + outerRadius);  System.***out***.println("填充颜色： " + color);  }    // 取得信息的方法  **public** **double** getInnerRadius() {  **return** innerRadius;  }  **public** **double** getouterRadius() {  **return** outerRadius;  }  **public** String getColor() {  **return** color;  }    // 修改圆环数据  **public** **void** setInnerRadius(**double** x) {  **if**(x - 0 > 0.0001)  innerRadius = x;  }  **public** **void** setouterRadius(**double** y) {  **if**(y - 0 > 0.0001)  outerRadius = y;  }  **public** **void** setColor(String color) {  **this**.color = color;  }    // 静态方法  **static** **void** setPI(**double** value) {  *PI* = value;  }  **static** **void** showCount() {  System.***out***.println("当前圆环类共有 " + *count* + "个对象");  }      // 计算并返回圆环面积  **public** **double** getArea() {  **double** area;  area = (outerRadius \* outerRadius - innerRadius \* innerRadius) \* *PI*;  **return** area;  }  }  // 3题代码  **public** **class** BookClassTest {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  SpecifyBooks Test = **new** SpecifyBooks();    System.***out***.println("展示课程信息：\n");  Test.show();  System.***out***.println();  Test.show(1);  System.***out***.println();  Test.Change("Java","java从入门到入土", 66.6,"java高级bug设计",10.24);  }  }  **public** **class** SpecifyBooks {  Class MyClass = **new** Class("java");    // 展示课程信息  **public** **void** show() {  System.***out***.println("展示一本参考书：");  MyClass.showInfo();  }  **public** **void** show(**int** i) {  System.***out***.println("展示两本参考书：");  MyClass.showInfo(1);  }    // 修改课程信息  **public** **void** Change(String className,String bookName1,**double** price1,String bookName2,**double** price2) {  System.***out***.println("课程信息修改后：");  MyClass.setClassName(className);  MyClass.changeBook(bookName1, price1,bookName2,price2);  MyClass.showInfo(1);  }      }  **public** **class** Class {  String className;  Book book1 = **new** Book("java从入门到精通",45.6);  Book book2 = **new** Book("java基础案例教程",32.6);    Class(String name) {  className = name;  }    // 显示信息  **public** **void** showInfo() {  System.***out***.println("课程名称： " + className );  System.***out***.println("本课程参考书目： ");  book1.showInfo();  }  **public** **void** showInfo(**int** i) {  System.***out***.println("课程名称： " + className );  System.***out***.println("本课程参考书目： ");  book1.showInfo();  book2.showInfo();  }    **public** **void** setClassName(String name) {  className = name;  }    **public** **void** changeBook(String name, **double** myprice) {  book1.set(name, myprice);  }  **public** **void** changeBook(String name1, **double** myprice1,String name2, **double** myprice2) {  book1.set(name1, myprice1);  book2.set(name2, myprice2);  }      }  **public** **class** Book {  **private** String bookName ;  **private** **double** price;    Book(String name, **double** price){  bookName = name;  **this**.price = price;  }  Book(){  bookName = "Java程序设计";  price = 66.6;  }  // 显示教材信息  **public** **void** showInfo() {  System.***out***.println("书名： " + bookName + " 售价： " + price +" 元");  }  // 修改信息  **public** **void** setBookName(String name) {  bookName = name;  }  **public** **void** setPrice(**double** price) {  **this**.price = price;  }  **public** **void** set(String name,**double** myprice) {  price = myprice;  bookName = name;  }  // 获取信息  **public** String getName() {  **return** bookName;  }    **public** **double** getPrice() {  **return** price;  }  } |

# 实验4：Java的面向对象特性（2）

1. **实验目的**

1. 掌握Java的继承与多态，能编写体现类的继承性（成员变量、成员方法、成员变量隐藏）、类的多态性（成员方法重载、构造方法重载）的程序。

2. 掌握抽象类与接口

3. 掌握Object类的equals方法和toString方法的使用

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 JetBrains IntelliJ IDEA2020.2

1. **设计思路与具体实现**

1.（1）设计思路

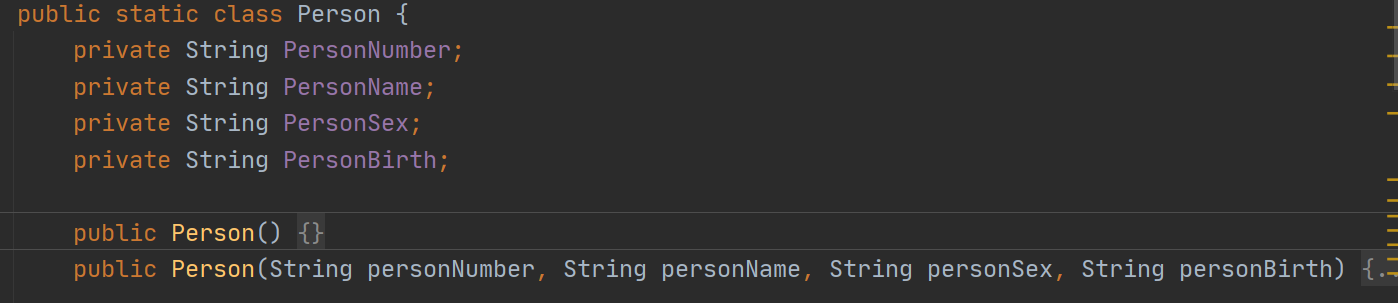
编写一个名为Person的父类，然后通过继承Person类实现Teacher和Student两个子类。Person类定义私有成员人的编号、姓名、性别以及出生日期，实现对应的get和set方法。Teacher类和Student类继承Person类，另外定义私有成员部门、职称、工资和高考分数、专业、班级，实现对应的get和set方法。

（2）实验步骤

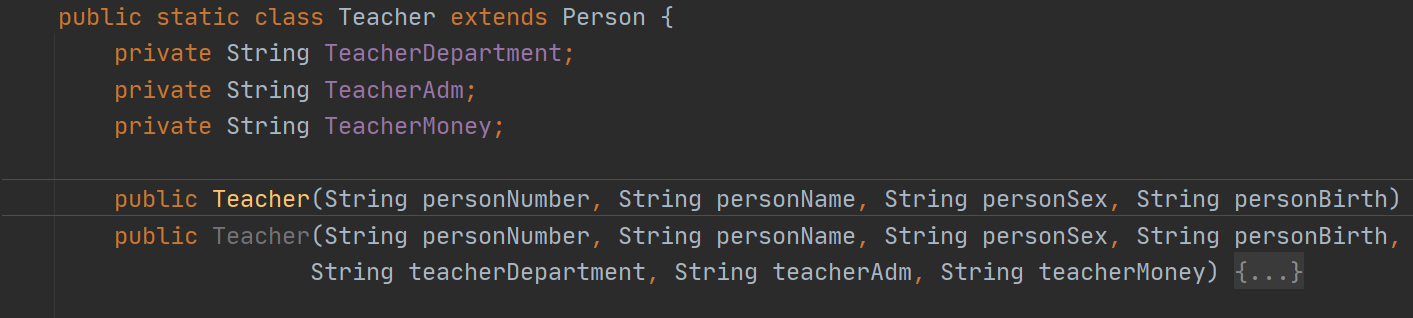
①编写程序

②编译、运行程序

（3）关键代码

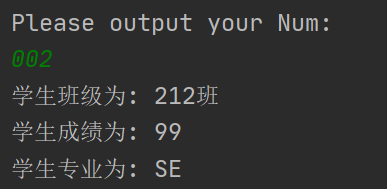






（4）程序测试过程

（5）运行结果（运行效果）



1.（1）设计思路

定义一个名为Student的抽象类，包含一些基本的学生信息，定义成员变量stuName和subScore来表示。定义两个抽象方法getType和getGrade，分别用来获取学生的类型和学生的成绩等级。另外定义实现获取学生成绩的方法setStuScore。

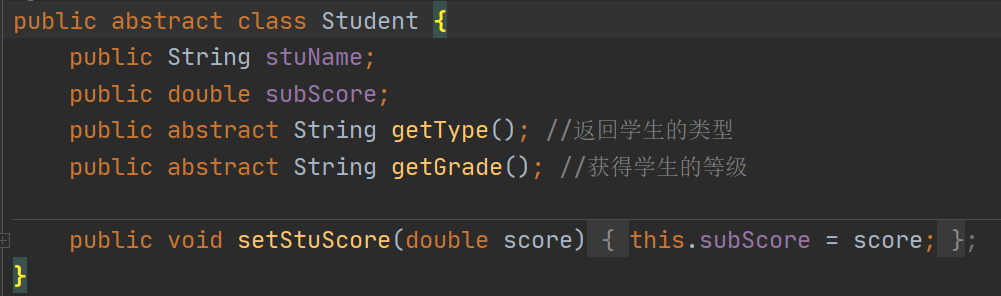
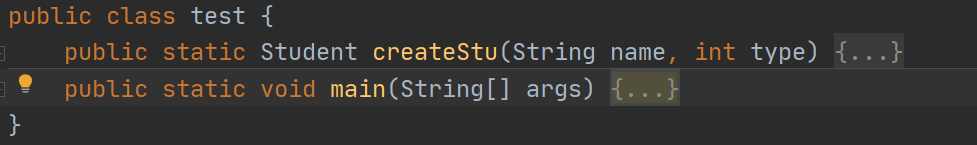
定义两个类Undergraduate和Postgraduate继承Student类，重写Student类中的getType和getGrade方法。设计一个构建学生的函数createStu(StuName, StuType)，根据StuType新建一个对应类型的学生，StuName学生的名字。

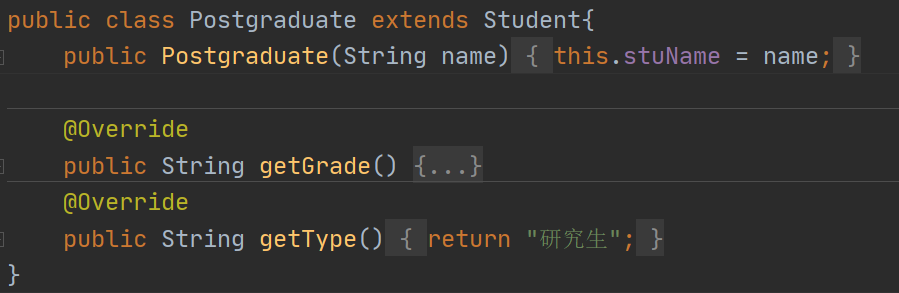
（2）实验步骤

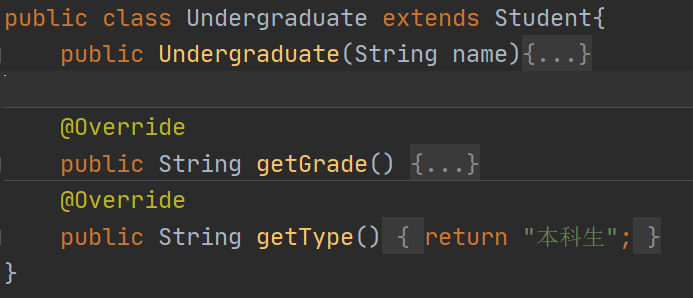
①编写程序

②编译、运行程序

（3）关键代码







（4）程序测试过程

（5）运行结果（运行效果）



1. **实验总结**

1. 主要涉及Java面向对象的特性，要以面向对象的思想去实现对代码的设计和构建。

2. 普通类的继承，子类的构造函数以及get和set方法，以及抽象类的继承，子类对父类抽象方法的实现。

1. **附录（程序代码）**

|  |
| --- |
| **import** java.util.ArrayList;  **class** Person {  **int** num;  String name;  String gender;  String birthday;    Person(**int** num,String name,String gender,String birthday){  **this**.num = num;  **this**.name = name;  **this**.gender = gender;  **this**.birthday = birthday;  }    }  // 继承  **class** Teacher **extends** Person{  String department;  String position;  **int** salary;    Teacher(**int** num,String name,String gender,String birthday,String department,  String position ,**int** salary){    **super**(num,name,gender,birthday);  **this**.department = department;  **this**.position = position;  **this**.salary = salary;    }    **public** **void** showInfo() {    System.***out***.println("\n教师信息：");  System.***out***.println("-----------------------------------");  System.***out***.println("编号： "+num);  System.***out***.println("姓名： " + name);  System.***out***.println("性别： "+ gender);  System.***out***.println("出生日期： " + birthday);  System.***out***.println("部门： " + department);  System.***out***.println("职位： " + position);  System.***out***.println("薪水： " + salary);  System.***out***.println("-----------------------------------");    }    }  **class** Student **extends** Person{  **int** score;  String field;  String iclass;    Student(**int** num,String name,String gender,String birthday,**int** score,String filed, String iclass){  **super**(num,name,gender,birthday);  **this**.score = score;  **this**.field = filed;  **this**.iclass = iclass;  }    **public** **void** showInfo() {    System.***out***.println("\n学生信息：");  System.***out***.println("-----------------------------------");  System.***out***.println("编号： "+num);  System.***out***.println("姓名： " + name);  System.***out***.println("性别： "+ gender);  System.***out***.println("出生日期： " + birthday);  System.***out***.println("高考成绩： " + score);  System.***out***.println("专业： " + field);  System.***out***.println("班级： " + iclass);  System.***out***.println("-----------------------------------");    }  }  **class** OP{  // 声明ArrayList集合  **static** ArrayList<Teacher> *te* = **new** ArrayList<Teacher>();  **static** ArrayList<Student> *st* = **new** ArrayList<Student>();    // 增  **public** **static** **void** add\_t(Teacher teacher1) {  *te*.add(teacher1);  }  **public** **static** **void** add\_s(Student student1) {  *st*.add(student1);  }    // 删  **public** **static** **void** del\_t(String name) {  **for**(**int** i = 0; i < *te*.size(); i++) {  **if**(*te*.get(i).name.equals(name)) {  *te*.remove(i);  }  }  }    **public** **static** **void** del\_s(String name) {  **for**(**int** i = 0; i < *st*.size(); i++) {  **if**(*st*.get(i).name.equals(name)) {  *st*.remove(i);  }  }  }    // 改  **public** **static** **void** change\_t(String name, Teacher teacher1) {  **int** index = -1;// 修改位置  **for**(**int** i = 0; i < *te*.size(); i++) {  **if**(*te*.get(i).name.equals(name)) {  index = i;  **break**;  }  }  **if**(index != -1)  *te*.set(index, teacher1);  }    **public** **static** **void** change\_s(String name, Student student1) {  **int** index = -1;// 修改位置  **for**(**int** i = 0; i < *st*.size(); i++) {  **if**(*st*.get(i).name.equals(name)) {  index = i;  **break**;  }  }  **if**(index != -1)  *st*.set(index, student1);  }  // 查  **public** **static** **void** show\_t(String name) {  **for**(**int** i = 0; i < *te*.size();i++) {  **if**(*te*.get(i).name.equals(name)) {  *te*.get(i).showInfo();  **break**;  }  }  }    **public** **static** **void** show\_s(String name) {  **for**(**int** i = 0; i < *st*.size();i++) {  **if**(*st*.get(i).name.equals(name)) {  *st*.get(i).showInfo();  **break**;  }  }  }    }  //测试  **public** **class** PersonManagTest {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Teacher teacher1 = **new** Teacher(1,"Bob","male","1990-10-5","软工","讲师",10000);  Teacher teacher2 = **new** Teacher(2,"Jan","famale","1990-5-5","软工","讲师",10000);  Student student1 = **new** Student(2020001,"Wan","female","2002-05-31",571,"软工","软工205");  Student student2 = **new** Student(2020002,"Yan","female","2002-03-21",611,"软工","软工205");    // 初始信息：  OP.*add\_t*(teacher1);  OP.*add\_t*(teacher2);  OP.*add\_s*(student1);  OP.*add\_s*(student2);  System.***out***.println("初始信息如下：");  PersonManagTest.*show*();  // 修改  Teacher teacher3 = **new** Teacher(3,"Tim","male","1990-11-3","软工","讲师",10000);  Student n\_student = **new** Student(2020006,"Lan","male","2001-03-31",670,"软工","软工205");  OP.*add\_t*(teacher3);  OP.*change\_s*("Yan",n\_student);  System.***out***.println("增加一位教师，修改一位学生的信息后：");  PersonManagTest.*show*();    // 删除  OP.*del\_t*("Bob");  System.***out***.println("删除一位教师后：");  PersonManagTest.*show*();  }      // 查询  **static** **void** show() {  **for**(**int** i = 0; i < OP.*te*.size(); i++) {  OP.*show\_t*(OP.*te*.get(i).name);  }  **for**(**int** i = 0; i < OP.*st*.size(); i++) {  OP.*show\_s*(OP.*st*.get(i).name);  }  }  } |

# 实验5： 异常处理

1. **实验目的**

1.掌握异常处理的基本概念，学会使用try...catch块捕获常见的运行期异常。

2. 学会用throws从函数中抛出异常的用法。

3. 学会创建自定义异常，以及手工抛出（产生）异常的throw语句。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 ，IDEA2020.2

1. **设计思路与具体实现**

1.（一）设计思路

编写整除运算程序，要求捕获除数为0异常、数字格式异常、通用型异常。注意要把通用型异常的捕获顺序放在最后。

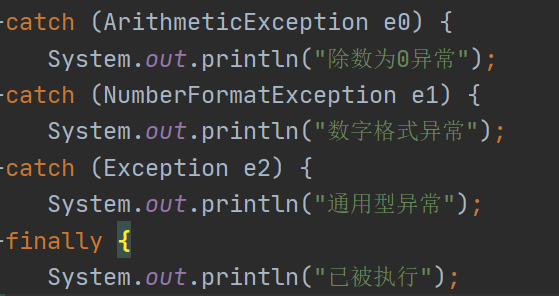
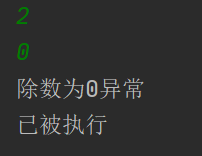
（二）实验步骤

①分别输入被除数和除数

②当除数为0时出现算法异常，打印输出“除数为0异常”

③保存代码，运行程序

（三）关键代码



2. （一）设计思路

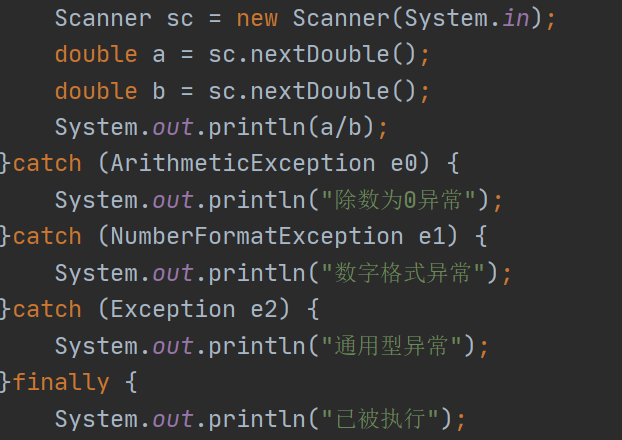
把第1题整除程序改为双精度型实数的除法运算程序，并更改有关提示信息，运行该程序若干次，每次输入不同的数据，观察分析程序的运行结果。

（二）实验步骤

①将第一题中的变量类型改为double型

②保存代码，运行程序

（三）关键代码



3. （一）设计思路

在第2题基础上编写自定义异常类（必须继承系统的Exception类），在除数为0时抛出自定义异常，并捕获处理该异常。

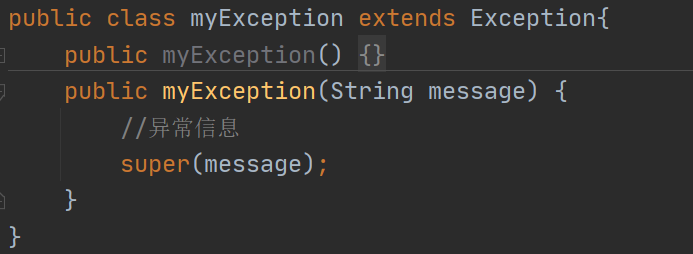
（二）实验步骤

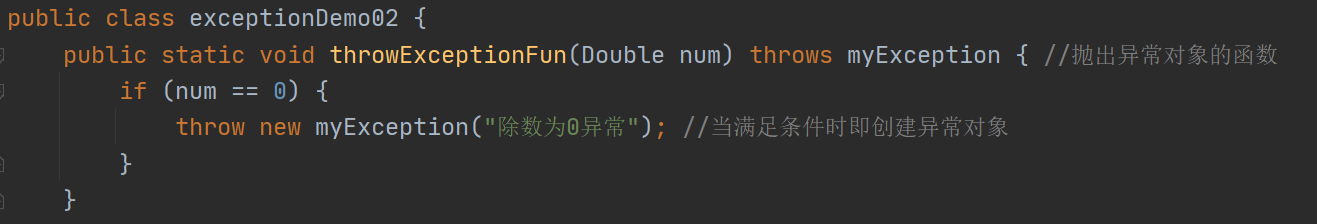
①定义一个自己的异常类，创建一个用来抛出异常的函数，当接收到0作为除数时即抛出异常。

②catch捕获抛出的异常，并打印输出异常信息。

③保存代码，运行程序。

（三）关键代码





1. **实验总结** 
   * + 1. 掌握try-catch语句，在平常写程序的时候要学会使用try-catch辅助开发。
       2. 了解掌握异常的抛出和接收
       3. 掌握一些基础的异常类
       4. 学会自定义一些简单的异常类
2. **附录（程序代码）**
   * + 1. 第一题：

****

* + - 1. 第二题：

****

* + - 1. 第三题（自定义异常类）：

****

* + - 1. 第三题（main函数+抛出异常的方法）

****

# 实验6：集合练习

1. **实验目的**

1.了解Set接口、List接口、Map接口及其主要实现类

2. 掌握ArrayList、 HashMap的使用

3. 掌握泛型类、泛型方法的创建和使用

4. 理解列表与映射、枚举和迭代 等集合类，能够创建并使用集合对象

1. **实验要求**

1. 实验在4学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 ，IDEA2020.2

1. **设计思路与具体实现**

1.（一）设计思路

编写程序练习List集合及泛型的基本使用：

1) 创建一个只能容纳String对象名为names的ArrayList集合；

2）按顺序往集合中添加5个字符串对象：“张三”、“李四”、“王五”、“马六”、“赵七”；

3）对集合进行遍历，分别打印集合中的每个元素的位置与内容；

4）首先打印集合的大小，然后删除集合中的第3个元素，并显示删除元素的内容，然后再打印目前集合中第3个元素的内容，并再次打印集合的大小。

（二）实验步骤

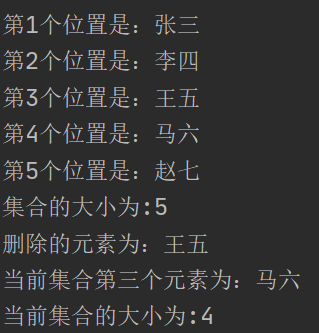
①采用增强for循环遍历数组

②调用数组中的方法完成对应要求

③保存代码，运行程序

（三）关键代码





2. （一）设计思路

编写这样一个程序，该程序创建了在某一字符串中出现的字符索引（即，哪个字符出现在字符串中的哪个地方）。从命令行读取该字符串。例如，运行该程序：

>java Concordance Hello world

将得到结果为：

{d=[9],o=[4,6],r=[7],w=[5],H=[0],l=[2,3,8],e=[1]}

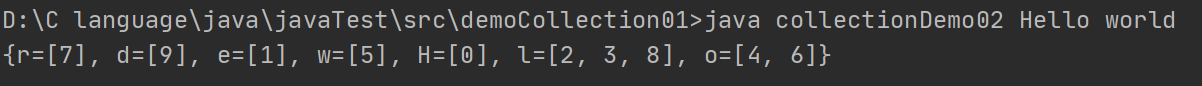
（二）实验步骤

①key-value的形式，采用map的数据结构

②遍历字符串数组来增加次数和所在位置的索引

（三）关键代码





1. **实验总结** 
   * + 1. 能够根据所需使用对应的数据结构。
       2. 掌握collection类中基础的方法
       3. 掌握一定泛型的知识。
2. **附录（程序代码）**

从上到下按顺序依次为第一题和第二题

****



# 实验7：文件操作练习

1. **实验目的**

1. 掌握Java输入输出的结构，

2. 掌握Java流的概念和利用流进行文件操作的方法。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 ，IDEA2020.2

1. **设计思路与具体实现**

1.（一）设计思路

编写文本文件复制程序，即把源文件复制到目标文件，运行时用参数方式输入源文件名和目标文件名，设入口主类为FileCopy，则运行方式为：

java FileCopy 源文件名 目标文件名

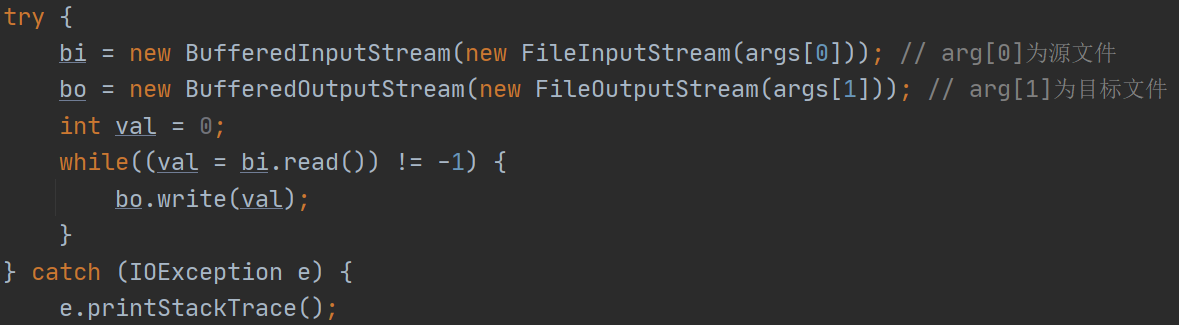
（二）实验步骤

①分别创建输入输出流对象

②从输入中逐个获取添加到输出流中

③保存代码，运行程序

（三）关键代码



2. （一）设计思路

将任意两个文件合并到一个文件，要求采用java命令行方式在控制台按照“源文件1 源文件2 目标文件” 方式录入，注意多种异常处理。

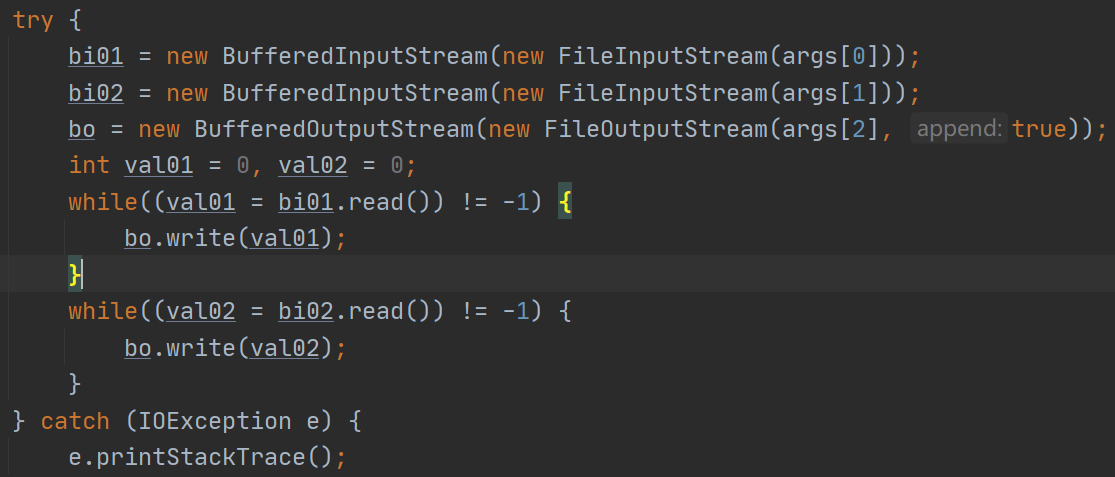
（二）实验步骤

①创建两个输入流对象和一个输出流对象

②分别从两个输入中逐个获取输入字符添加到输出中

③保存代码，运行程序

（三）关键代码



3. （一）设计思路

编写程序实现将一个文件内容追加到另一个文件内容后，如将D盘file文件夹下的f1.txt追加到E盘根目录下的f2.txt中。(必须异常处理)

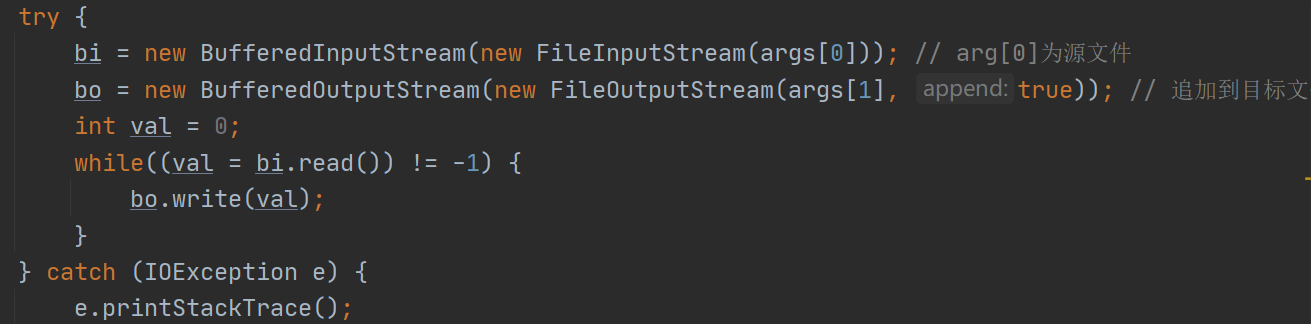
（二）实验步骤

①创建一个输入流和输出流对象，输出流模式选择为追加

②从输入流中逐个获取输入添加到输出中

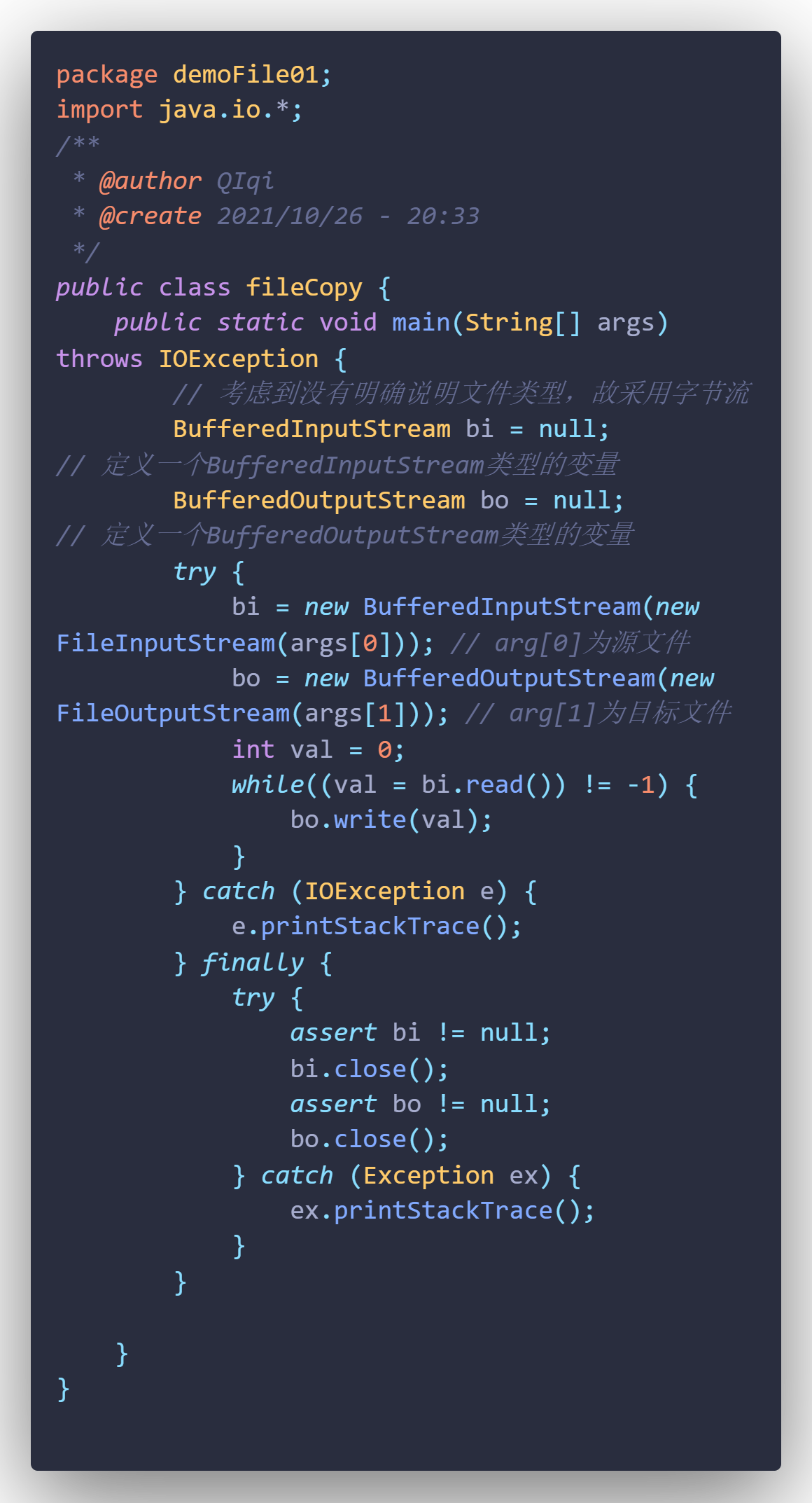
③保存代码，运行程序

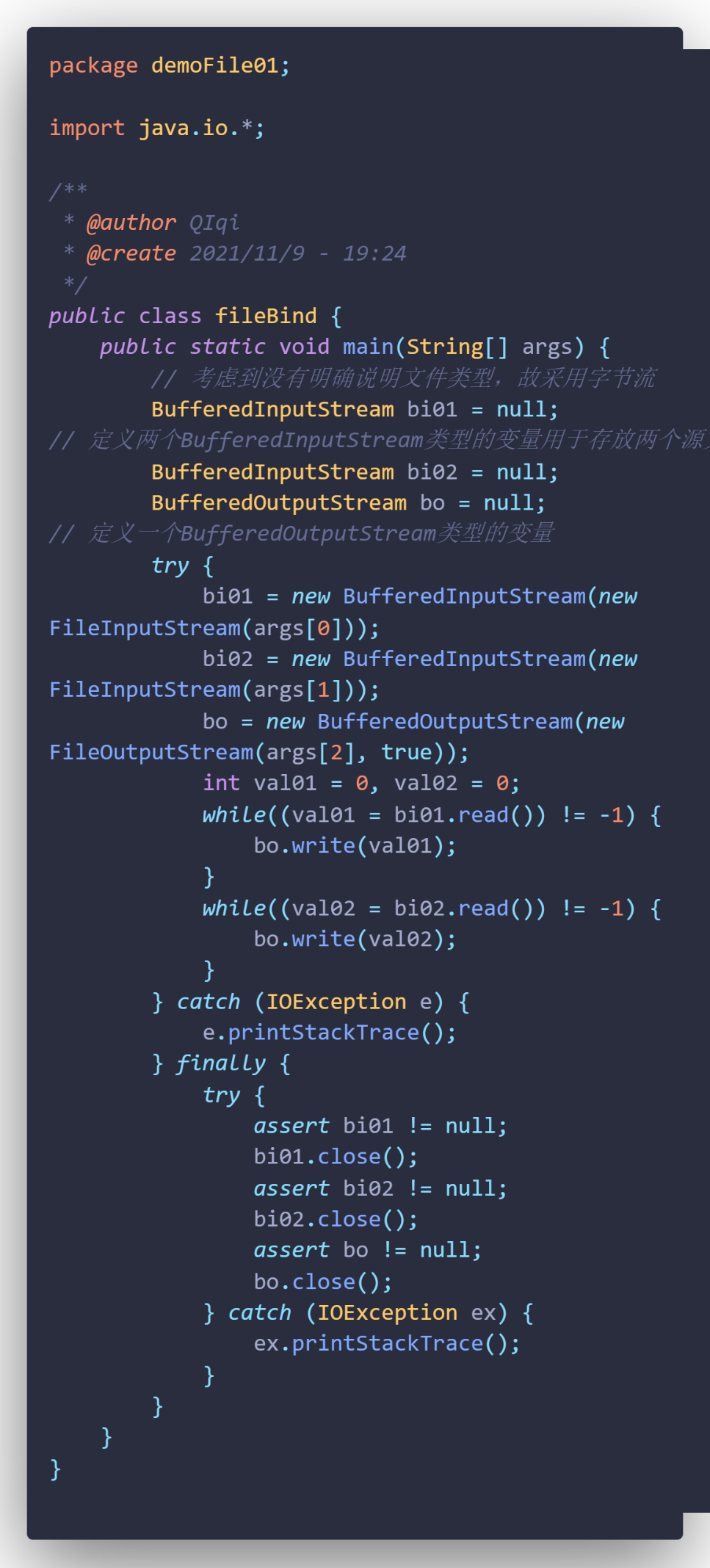
（三）关键代码

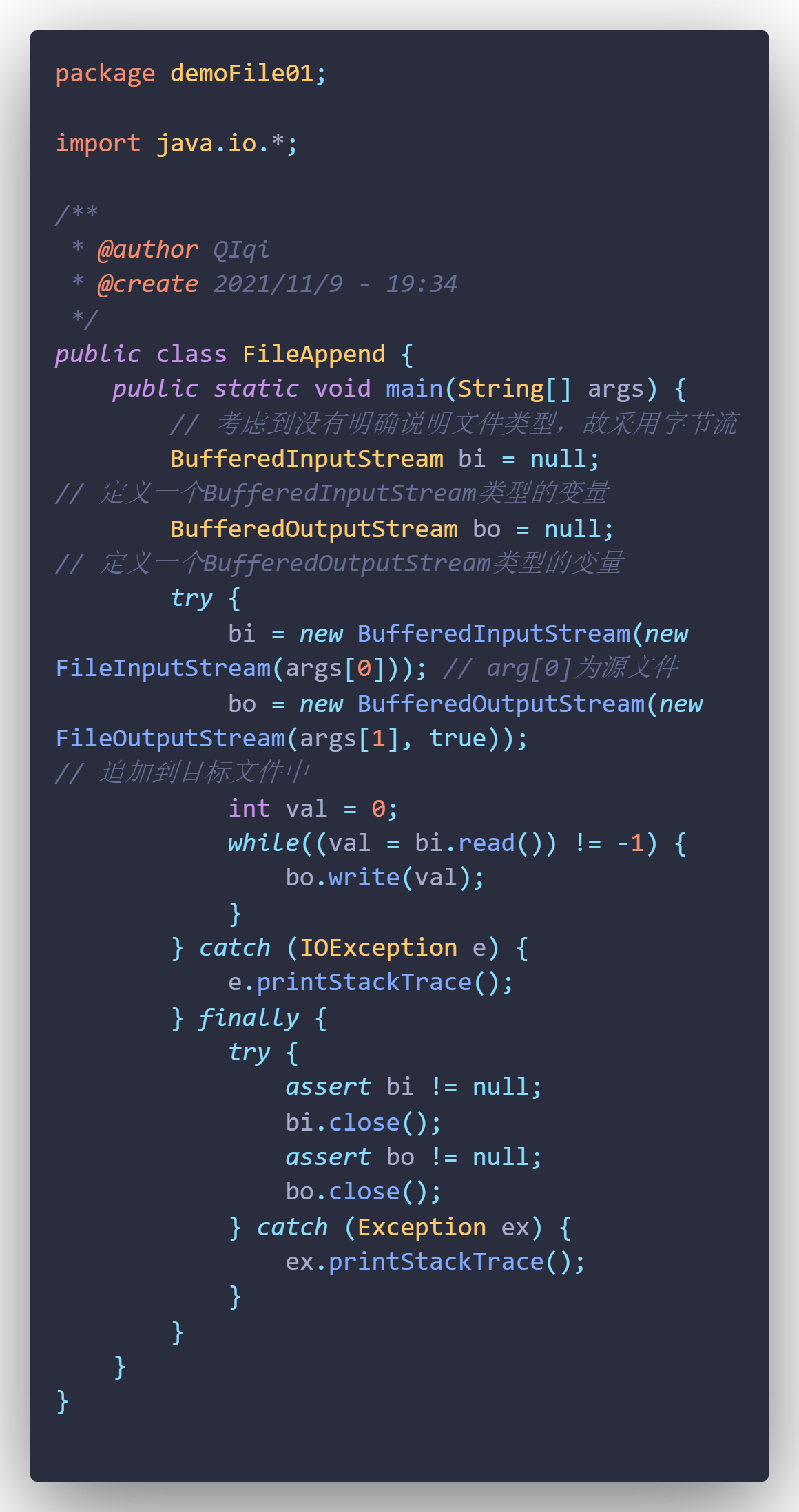


1. **实验总结**
   * + 1. 明确使用字符流还是字节流，针对性的采用对应的方法
       2. 注意对各种异常的分析
       3. java中文件路径 \\
       4. FileOutputStream方法的第二个参数若为true，则采用追加的方式。
2. **附录（程序代码）**

下面三张截图fileCopy，fileBind，fileAppend按照顺序分别为实验1,2,3的代码







# 实验8：多线程应用练习

1. **实验目的**

1. 掌握Java输入输出的结构，

2. 掌握Java流的概念和利用流进行文件操作的方法。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK11.0.2 ，IDEA2020.2

1. **设计思路与具体实现**

1.（一）设计思路

建立三个线程，并且同时运行它们。当运行时输出线程的名称。

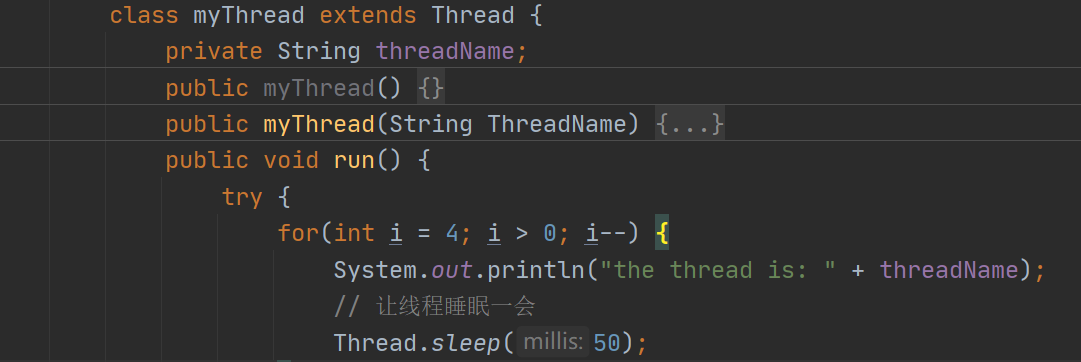
（二）实验步骤

(1)．创建一个Java类

(2)．在类中创建三个线程，调用start()方法启动这三个线程

(3)．保存文件，调试并编译运行程序。

（三）关键代码



2. 3。（一）设计思路

实现3个类：Storage、Counter和Printer。 Storage类应存储整数。 Counter应创建线程，线程从0开始计数（0,1,2,3…）并将每个值存储到Storage类中。 Printer类应创建一个线程，线程读取Storage类中的值并打印值。编写程序创建Storage类的实例，并创建一个Counter对象和Printer对象操作此实例。

（二）实验步骤

(1)．创建三个类Counter， Printer，Storage

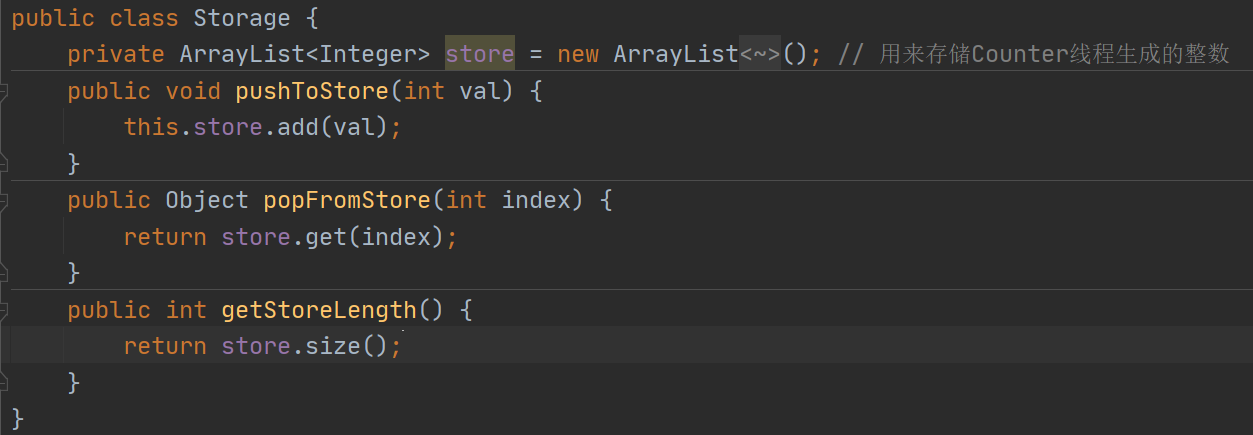
(2)．创建TestCounter类，在该类中定义main函数，在main函数中定义Storage对象、Counter对象和 Printer对象，创建Counter线程和Printer线程并启动

(3)．保存文件，调试并编译运行程序。

（三）关键代码







由于第二题已经达到第三题要求，故第三题参照第二题！

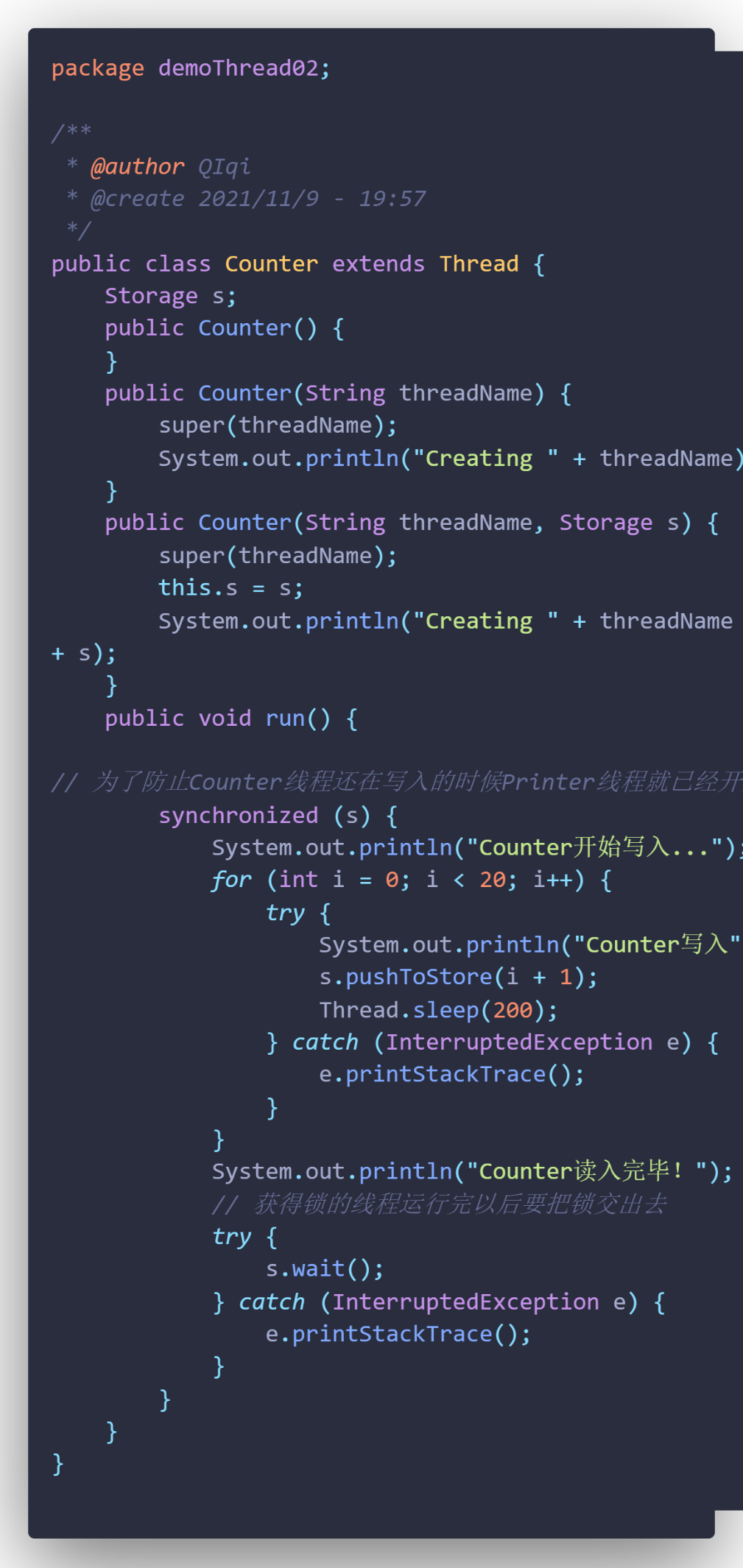
1. **实验总结**

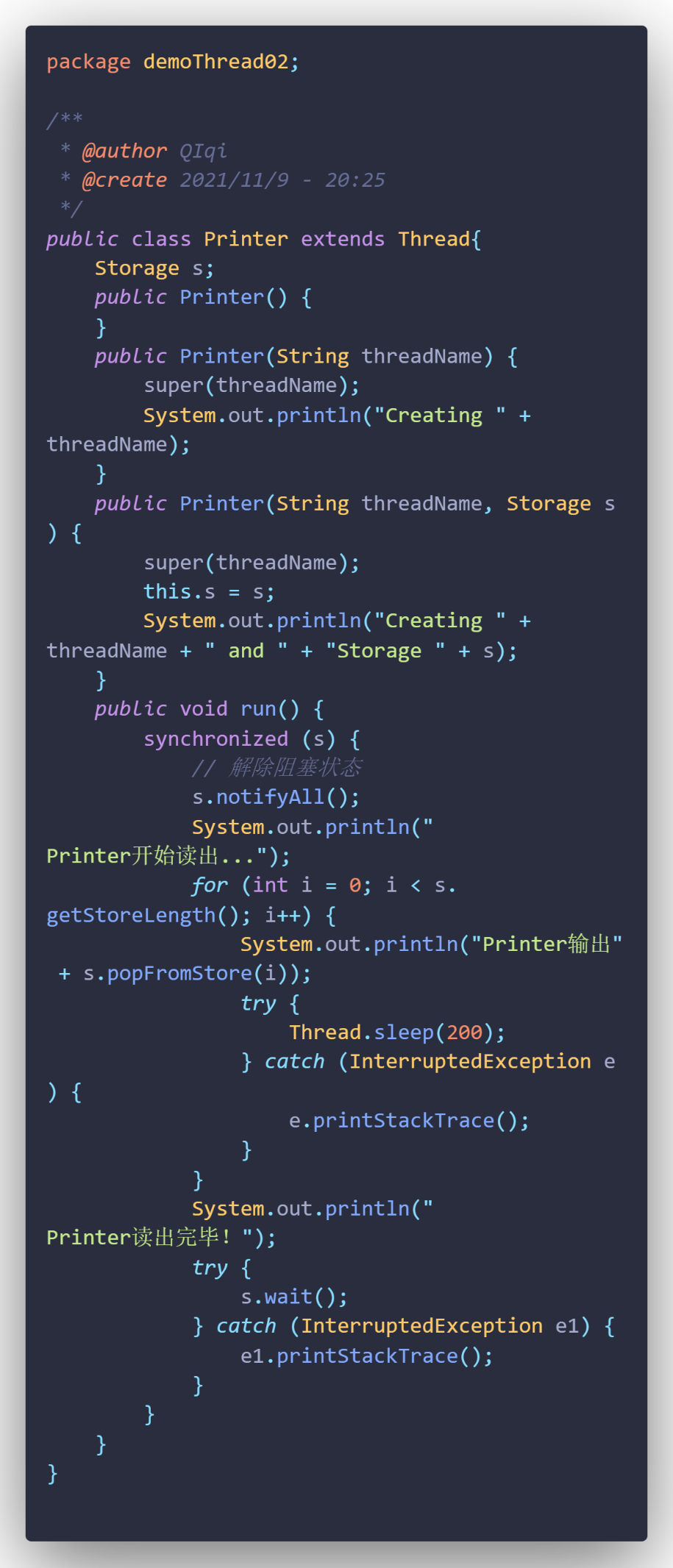
1.注意线程锁的使用，通过其来实现多个线程之间按照规则的运行。

1. **附录（程序代码）**

下面五张截图按照顺序分别为实验1,实验2的四个类的代码











# 实验9：

1. **实验目的**
   1. 理解Java网络模型。
   2. 掌握InetAddress类的使用。
   3. 理解Socket的相关概念、原理。
   4. 掌握Java Socket编程，学会创建Java网络服务器和网络客户机。
   5. 理解基于UDP协议的Socket的相关概念、原理。
   6. 掌握基于UDP协议的Socket编程的步骤及方法。
   7. 巩固Java GUI 编程和数据流的使用。
2. **实验要求**
3. 实验在2学时内完成。
4. 1人1组独立完成。
5. **实验设备与环境**

JDK11.0.2 ，IDEA2020.2

1. **设计思路与具体实现**

1．基于TCP的Socket编程。使用套接字编写客户—服务器程序，实现客户—服务器交互计算。客户将三角形3个边的长度发给服务器，服务器把计算出的三角形的面积返回给客户。

**具体实现：**

(1)． 服务器端：Server.java

(a). 调用ServerSocket(int port)创建一个服务器端套接字，并绑定到指定端口上；

(b). 调用accept()，监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字；

(c). 调用Socket类的getOutputStream()和getInputStream获取输出流和输入流，使用输入流获取客户端传来的3条边的长度；

(d). 计算面积；

(e). 将面积通过输出流传递到客户端；

(f). 最后关闭通信套接字。

(2)． 客户端Client.java

(a). 创建界面，在类的构造器中调用Socket()创建一个流套接字；

(b). 进行事件处理：按钮对象调用addActionListener注册监听，重写actionPerformed方法进行事件处理的具体操作

(c). 在事件处理过程中，客户端Socket连接到服务器端，调用Socket类的getOutputStream()和getInputStream获取输出流和输入流，开始网络数据的发送和接收。将三边传到服务器端，从服务器端获取计算出来的面积。最后关闭通信套接字。

**主要代码：**

|  |
| --- |
| // 客户端：  for (int i = 0; i < *NUM*; i++) {  // flush()方法可以强迫输出流(或缓冲的流)发送数据，即使此时缓冲区还没有填满，以此来打破这种死锁的状态。  os.flush();  // 把从终端上输入的字节流装进readLine中  readline = sin.readLine();  // 写出readLine到输出的套接字（通道中）中  os.println(readline); }  // 服务器端：  // 利用字节流缓存区，接收并暂存从通道中获取到的字节流 BufferedReader is = new BufferedReader  (new InputStreamReader(socket.getInputStream())); System.*out*.printf("从客户端接受的三角形的三条边为："); for (int i = 0; i < *NUM*; i++) {  lines = is.readLine();  a[i] = Double.*parseDouble*(lines);  System.*out*.print(a[i] + " "); } |

2. 基于UDP的Socket编程。编写一个基于UDP协议的Socket网络通信应用程序，实现如下功能：

1. 客户端能够发送任意的信息给服务器端。
2. 服务器端将收到的字符串变成大写后返还给客户端。

**实验步骤：**

(1)． 接收端

(a). 调用DatagramSocket(int port)创建一个数据报套接字，并绑定到指定端口上；

(b). 调用DatagramPacket(byte[] buf, int length)，建立一个字节数组以接收UDP包；

(c). 调用DatagramSocket类的receive()，接收UDP包；

(d). 最后关闭数据报套接字。

(2)． 发送端

(a). 调用DatagramSocket()创建一个数据报套接字；

(b). 调用DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)，建立要发送的UDP包；

(c). 调用DatagramSocket类的send()，发送UDP包；

(d). 最后关闭数据报套接字。

**主要代码：**

|  |
| --- |
| // 客户端：  byte[] receiveData = new byte[256]; DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length); // 等待服务器发送过来处理好的字符 sendSocket.receive(receivePacket); String receiveString = new String(receivePacket.getData()).substring(0, buf.length()); System.*out*.println("接收到的字符串为：" + receiveString);  // 服务器端：  // 从到来的包中得到地址和端口号 address = packet.getAddress(); port = packet.getPort(); String receiveString = new String(packet.getData()).substring(0, packet.getLength()); System.*out*.println("客户端说：" + receiveString); String sendData = receiveString.toUpperCase(); // 将数据送入缓冲区,或来自文件,或键盘输入 // 建立报文包,用来从socket上发送信息 packet = new DatagramPacket(sendData.getBytes(), sendData.length(), address, port); |

1. **实验总结** 
   * + 1. 了解网络编程相关知识，如http请求，TCP/IP协议等
       2. 掌握计算机网络对应知识，了解IP、网关、TCP和UDP协议以及两者之间的区别与联系、DNS及域名等知识
       3. 能够根据不同的情况使用更适合的网络通信协议
       4. 了解三次握手四次挥手
2. **附录（程序代码）**

1.客户端：

|  |
| --- |
| package InteDemo00; import java.io.\*; import java.net.\*;  public class TcpClient {  public static final int *NUM* = 3;  public static void main(String args[]){  try {  // 对标服务器端建立套接字（通信通道）  Socket socket = new Socket("127.0.0.1", 4700);  // 建立字节流缓冲区，以键盘输入为读入源  BufferedReader sin = new BufferedReader(new InputStreamReader(System.*in*));  // 建立输出流，以将在通信通道中发送给服务器端的流为输出源  PrintWriter os = new PrintWriter(socket.getOutputStream());  System.*out*.println("从客户端接收三角形的三条边发送到服务器：");  System.*out*.println("请输入3条边：");  String readline = "";  for (int i = 0; i < *NUM*; i++) {  // flush()方法可以强迫输出流(或缓冲的流)发送数据，即使此时缓冲区还没有填满，以此来打破这种死锁的状态。  os.flush();  // 把从终端上输入的字节流装进readLine中  readline = sin.readLine();  // 写出readLine到输出的套接字（通道中）中  os.println(readline);  }  os.close();  socket.close();  } catch (Exception e) {  e.printStackTrace();  }  } } |

1.服务器端：

|  |
| --- |
| package InteDemo00; import java.net.\*; import java.io.\*;  public class TcpServer {  public static final int *NUM* = 3;  public static void main(String[] args) {  TcpServer tcpserver = new TcpServer();  try{  // 调用服务端创建一个服务端套接字服务，绑定至4700端口  // 同时做异常判断  ServerSocket server = null;  try {  server = new ServerSocket(4700);  } catch(Exception e) {  e.printStackTrace();  }   // 创建套接字对象  // 让通信通道（套接字对象）处于监听状态，做异常判断  Socket socket = null;  try {  socket = server.accept();  } catch(Exception e) {  e.printStackTrace();  }   // 建立服务端接收流（read）  String lines = "";  double a[] = new double[*NUM*];  // 利用字节流缓存区，接收并暂存从通道中获取到的字节流  BufferedReader is = new BufferedReader  (new InputStreamReader(socket.getInputStream()));  System.*out*.printf("从客户端接受的三角形的三条边为：");  for (int i = 0; i < *NUM*; i++) {  lines = is.readLine();  a[i] = Double.*parseDouble*(lines);  System.*out*.print(a[i] + " ");  }  System.*out*.println();   try {  double area = tcpserver.countArea(a);  System.*out*.println("三角形的面积是：" + tcpserver.countArea(a));  } catch (Exception e0) {  e0.printStackTrace();  }   // 关闭字节流缓存区、通信通道和服务器  is.close();  socket.close();  server.close();  } catch (Exception e) {  e.printStackTrace();  }  }   public double countArea(double a[]) throws Exception{  double area = 0.00;  double temp = 0.00;  double aside = 0.00;  double bside = 0.00;  double cside = 0.00;   aside = a[0];  bside = a[1];  cside = a[2];  if (aside + bside <= cside || aside + cside <= bside || bside + cside <= aside) {  throw new Exception("三角形不存在！");  } else {  temp = a[0] + a[1] + a[2];  temp = temp / 2;   area = Math.*sqrt*(temp \* (temp - aside) \* (temp - bside) \* (temp - cside));   return area;  }  } } |

2.客户端

|  |
| --- |
| package InteDemo01; import java.net.\*; import java.io.\*;  public class Client {  public static void main(String[] args) {  try {  DatagramSocket sendSocket = null;  DatagramPacket sendPacket = null;  InetAddress loc = null;  try {  sendSocket = new DatagramSocket(4000);  } catch (SocketException e) {  e.printStackTrace();  }  // 建立字节流缓冲区，以键盘输入为读入源  BufferedReader sin = new BufferedReader(new InputStreamReader(System.*in*));  String inputLine = "";  String buf = "";  System.*out*.print("输入要发送的字符串：");  try {  inputLine = sin.readLine();  } catch (IOException ex) {  ex.printStackTrace();  }  while(inputLine.length() > 0) {  buf += inputLine;  try {  inputLine = sin.readLine();  } catch (IOException ex) {  ex.printStackTrace();  }  }  // System.out.println("!");   try {  loc = InetAddress.*getLocalHost*();//获取本机地址  } catch (UnknownHostException ex) {  ex.printStackTrace();  }  sendPacket = new DatagramPacket(buf.getBytes(), buf.length(), loc, 7000);  try {  sendSocket.send(sendPacket);  } catch (IOException ex) {  ex.printStackTrace();  }  // System.out.println(loc);   byte[] receiveData = new byte[256];  DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  // 等待服务器发送过来处理好的字符  sendSocket.receive(receivePacket);  String receiveString = new String(receivePacket.getData()).substring(0, buf.length());  System.*out*.println("接收到的字符串为：" + receiveString);  // 关闭socket和输入流  sendSocket.close();  try {  sin.close();  } catch (IOException ex) {  ex.printStackTrace();  }  } catch (Exception e) {  e.printStackTrace();  }  } } |

2.服务器端

|  |
| --- |
| package InteDemo01; import java.io.\*; import java.net.\*;  public class Server {  public static void main(String[] args) {  try{  // 定义数据成员  DatagramSocket socket = null;  DatagramPacket packet = null;  InetAddress address; //(用来存放接收方的地址)  int port; //(用来存放接收方的端口号)  // 创建数据报文Socket对象  try {  socket = new DatagramSocket(7000);  } catch(SocketException e) {  e.printStackTrace();  }  byte[] buf = new byte[256]; //存放从客户端接收的请求信息.  // 创建一个DatagramPacket  packet = new DatagramPacket(buf, 256); // 用来从socket接收数据,它只有两个参数   // 服务器阻塞  try {  System.*out*.println("等待客户端发送请求...");  socket.receive(packet);  } catch (IOException e) {  e.printStackTrace();  }  // 在客户的请求报道来之前一直等待   // 从到来的包中得到地址和端口号  address = packet.getAddress();  port = packet.getPort();  String receiveString = new String(packet.getData()).substring(0, packet.getLength());  // System.out.println("length:" + packet.getLength());  System.*out*.println("客户端说：" + receiveString);   String sendData = receiveString.toUpperCase();  // System.out.println(sendData);  // 将数据送入缓冲区,或来自文件,或键盘输入  // 建立报文包,用来从socket上发送信息  packet = new DatagramPacket(sendData.getBytes(), sendData.length(), address, port);  // 发送数据包  try {  socket.send(packet);  } catch (IOException e) {  e.printStackTrace();  }  // 关闭socket  socket.close();  } catch(Exception e0) {  e0.printStackTrace();  }  } } |