实 验 报 告

课程名称: Java语言程序设计 姓 名: 安民

学 号: P181713164

班 级: 2018级计算机科学与技术1班

数学与计算机科学学院

西北民族大学数学与计算机科学学院实验报告

**实验名称：Java语言基础**

**指导教师**：蔡正琦 **日期：**2020年 4月13日

|  |
| --- |
| 实验内容  1.输入一个16位的长整型数，利用switch语句统计其中0-9每个数字出现的次数。  2.求a+aa+aaa+…+a…a(n个)的和，其中a为1-9之间的整数。例如，当a=3，n=4时，求3+33+333+3333的和。  3.给定一个正整数m，判断它的具体位数，分别打印每一位数，再按照逆序打印出各位数字。  4.编写程序将16进制数转换为10进制数。  5.编写程序对10个整数进行排序。 |
| 实验目的  (1)掌握条件语句的使用。  (2)掌握循环语句的使用。  (3)掌握数组的定义和使用方法。  (4)熟悉数组的排序、查找等算法。 |
| 实验设备  硬件：x64 处理器，内存8GB  软件：Win10操作系统，IDE Eclipse 2019-12和VS CODE  VS CODE插件：Language Support for Java(TM) by Red Hat、Debugger for Java、Java Test Runner、Maven for Java、Java Dependency Viewer |
| 实验过程  1.实验代码  **package** test;  **import** java.util.Scanner;  **public** **class** test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  //输入一个16位的长整型数，利用switch语句统计其中0~9每个数字出现的次数。  Scanner sc=**new** Scanner(System.***in***);  System.***out***.println("请输入一个16位长整型数：");  **long** s=sc.nextLong();  **if**((s+"").length()!=16)  System.***out***.println("输入有误，请重新开始。");  **else**{  **double** i;  **long** a,b;  **int** n0=0,n1=0,n2=0,n3=0,n4=0,n5=0,n6=0,n7=0,n8=0,n9=0;  //s%10=个位数，s/(10^i)=i位前的数  **for**(i=0;i<16;i++) {  **long** c=(**long**)Math.*pow*(10, i);  a=s/c;  b=a%10;  **switch**((**int**)b){  **case** 0:  n0++;**break**;  **case** 1:  n1++;**break**;  **case** 2:  ++n2;**break**;  **case** 3:  ++n3;**break**;  **case** 4:  ++n4;**break**;  **case** 5:  ++n5;**break**;  **case** 6:  ++n6;**break**;  **case** 7:  ++n7;**break**;  **case** 8:  ++n8;**break**;  **case** 9:  ++n9;**break**;  **default**:**break**;  }  }  System.***out***.println(s+"中0出现的次数是："+n0+"次");  System.***out***.println(s+"中1出现的次数是："+n1+"次");  System.***out***.println(s+"中2出现的次数是："+n2+"次");  System.***out***.println(s+"中3出现的次数是："+n3+"次");  System.***out***.println(s+"中4出现的次数是："+n4+"次");  System.***out***.println(s+"中5出现的次数是："+n5+"次");  System.***out***.println(s+"中6出现的次数是："+n6+"次");  System.***out***.println(s+"中7出现的次数是："+n7+"次");  System.***out***.println(s+"中8出现的次数是："+n8+"次");  System.***out***.println(s+"中9出现的次数是："+n9+"次");  System.***out***.println("\tThank you!\n------exiting------");  }  }  }  2.实验代码  **package** test;  **import** java.util.Scanner;  **public** **class** test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  //求a+aa+aaa+…+a…a(n个)的和，其中a为1-9之间的整数。例如，当a=3，n=4时，求3+33+333+3333的和。  Scanner sc=**new** Scanner(System.***in***);  System.***out***.println("请输入n：");  **int** n=sc.nextInt();  Scanner sc1=**new** Scanner(System.***in***);  System.***out***.println("请输入a：");  **int** a=sc1.nextInt();  **long** sum = 0;  **int** temp = 0;  **for** (**int** i = 0; i < n; i++) {  temp = temp \* 10 + a;  sum += temp;  // 将每个数给打印容出来  System.***out***.println(temp);  }  System.***out***.println(sum);  }  }  3.实验代码  **package** test;  **import** java.util.Scanner;  **public** **class** test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  //给定一个正整数m，判断它的具体位数，分别打印每一位数，再按照逆序打印出各位数字。  Scanner sc1=**new** Scanner(System.***in***);  System.***out***.println("请输入数字m：");  **int** m=sc1.nextInt();  String str=m+"";  System.***out***.println("一共有"+str.length()+"位");  System.***out***.println("每位数字是：");  **for**(**int** i=0;i<str.length();i++)  {  System.***out***.print(str.charAt(i)+" ");  }  System.***out***.println("\n逆序");  **for**(**int** i=str.length()-1;i>=0;i--)  {  System.***out***.print(str.charAt(i)+" ");  }    }  }  4.实验代码  **package** test;  **import** java.util.Scanner;//导入扫描器  **public** **class** test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  //编写程序将16进制数转换为10进制数。  Scanner scan = **new** Scanner(System.***in***);  System.***out***.println("please enter a Hex:");  String hex = scan.nextLine();//读取一行  hex = hex.toUpperCase();//转换成大写字母  System.***out***.println("The hex is:" + hex);//输出一下  **int** decimal = 0;  **for** (**int** i = 0; i < hex.length(); i++) {  **if** (*hexXChangeDecimal*(hex.charAt(hex.length() - 1 - i)) != -1) {//从16进制数的最后一个字符开始获取  decimal = (**int**) (decimal + *hexXChangeDecimal*(hex.charAt(hex.length() - 1 - i)) \* Math.*pow*(16, i));//乘以16的0次幂，然后++  } **else** {  System.***out***.println("enter error, decimal will be zero!");//如果等于-1则是非法字符  **break**;  }  }  System.***out***.println("decimal=" + decimal);  }  **private** **static** **int** hexXChangeDecimal(**char** charAt) {  **if** (charAt >= 'A' && charAt <= 'F')  **return** charAt - 'A' + 10;//A~F转换成10进制数  **else** **if** (charAt >= '0' && charAt <= '9')  **return** charAt-'0';//0~9字符转换成10进制  **else**  **return** -1;    }  }  5.实验代码  **package** test;  **import** java.util.Scanner;//导入扫描器  **public** **class** test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  //10个数排序  //先定义一个数组，存放10个数  **int**[] a=**new** **int**[10];  Scanner sc=**new** Scanner(System.***in***);      **for** (**int** i = 0; i<a.length; i++) {  System.***out***.println("请输入第" + (i + 1) + "个元素:");  a[i] = sc.nextInt();  }  **int** temp;  **for**(**int** j=0;j<a.length;j++)  {  **for**(**int** i=0;i<a.length-1;i++)  {  **if**(a[j]<a[i])  {  temp=a[i];  a[i]=a[j];  a[j]=temp;  }  }  }  System.***out***.print("从小到大的排序为：");  /\*for(int i=0;i<a.length;i++)  {  System.out.print(a[i]+",");  }\*/  **for**(**int** num:a) {//增强型for循环一般用来输出数组所有元素较方便  System.***out***.print(num+" ");  }  }  } |
| 实验结果  1.    2.    3.    4.    5. |
| 结果分析与结论  实验中通过查阅资料获得了Eclipse中自动生成get和set方法的三种办法。  第一种为：1.在声明的类中，单击鼠标右键，选择source；2.选择Generate Getters and Setters；3.根据需要配置，如位置，顺序等，确定即可。  第二种为：1.在声明的数据域中按Ctrl+1；2.点击最后一个选项Create getter and setter，在弹出的对话框中点击确定。  第三种：1.声明完类的数据域之后，输入set，按住Alt+/；2.同步骤一，输入get，再按住Alt+/。 |
| 教师评阅及成绩  教师签名：  年 月 日 |

实 验 报 告

课程名称: Java语言程序设计 姓 名: 安民

学 号: P181713164

班 级: 2018级计算机科学与技术1班

数学与计算机科学学院

西北民族大学数学与计算机科学学院实验报告

**实验名称：类和对象**

**指导教师**：蔡正琦 **日期：**2020年 4月27日

|  |
| --- |
| 实验内容  1．设计一个Dog类，有名字、颜色和年龄属性，定义构造方法初始化这些属性，定义输出方法show()显示其属性。编写应用程序使用Dog类。  2．(1)编写一个学校类，其中包含成员变量scoreLine(录取分数线)和对该变量值进行设置和获取的方法。  (2)编写一个学生类，它的成员变量有考生的name(姓名)、id(考号)、total(综合成绩)、sports(体育成绩)。它还有获取学生的综合成绩和体育成绩的方法。  (3)编写一个录取类，它的一个方法用于判断学生是否符合录取条件。其中录取条件为：综合成绩在录取分数线之上，或者体育成绩在96以上并且综合成绩大于300。在该类的main()方法中，建立若干个学生对象，对符合录取条件的学生，输出其信息及“被录取”。  3. 设计一个表示用户的类user，有用户名、口令(私有)和记录用户数(静态)的成员变量。定义类的构造方法、设置或获取口令的方法及返回类对象信息的方法。编写应用程序测试user类。  4. (1)设计一个表示二维平面上点的类Point，包含有表示坐标位置的protected类型的成员变量x和y，获取和设置x和y值的public方法。  (2)设计一个表示二维平面上圆的类Circle，它继承自类Point，还包含有表示圆半径的protected类型的成员变量r、获取和设置r值的public方法、计算圆面积的public方法。  (3)设计一个表示圆柱体的类Cylinder，它继承自类Circle，还包含有表示圆柱体高的protected类型的成员变量h、 获取和设置h值的public方法、计算圆柱体体积的public方法。  (4)建立若干个Cylinder对象，输出其轴心位置坐标、半径、高及其体积的值。 |
| 实验目的  （1）掌握类和构造方法的定义  （2）理解静态和非静态成员的区别  （3）掌握创建类实例的方法  （4）掌握类的继承机制 |
| 实验设备  硬件：x64 处理器，内存8GB  软件：Win10操作系统，IDE Eclipse 2019-12和VS CODE  VS CODE插件：Language Support for Java(TM) by Red Hat、Debugger for Java、Java Test Runner、Maven for Java、Java Dependency Viewer |
| 实验过程  1.实验代码  **package** test;  //文档注释  /\*\*  \* 宠物狗类  \* **@author** AM  \*  \*/  **public** **class** Dog{  //成员属性 名字、颜色和年龄  String name;  String color;  **int** month;//年龄    //无参构造方法  **public** Dog() {  System.***out***.println("无参构造方法");  }  //带参构造方法  **public** Dog(String name,String color,**int** month) {  **this**.name=name;  **this**.color=color;  **this**.month=month;  }  //成员方法 跑动  **public** **void** run() {  System.***out***.print("小狗快跑");  System.***out***.println();  }  //成员方法 吃东西  **public** **void** eat() {  System.***out***.print("小狗吃骨头");  System.***out***.println();  }    **public** **void** show(){  System.***out***.println("名字："+name);  System.***out***.println("颜色："+color);  System.***out***.println("年龄："+month);  }  }  **package** test;  **public** **class** test {  **public** **static** **void** main(String[] args){  // 声明对象+对象实例化  Dog one=**new** Dog("小白","write",10);  //测试  one.run();  one.eat();  one.show();  }  }  2.实验代码  （1）学校类：  **package** Luqu;  //文档注释  /\*\*  \* 学校类  \* **@author** AM  \*  \*/  **public** **class** School{  //成员属性  **private** **double** scoreLine;    **public** **double** getScoreLine() {  **return** scoreLine;  }  **public** **void** setScoreLine(**double** scoreLine) {  **this**.scoreLine = scoreLine;  }    }  （2）学生类：  **package** Luqu;  **public** **class** Student {  String name;  **int** id;  **double** total;  **double** sports;  **public** String getName() {  **return** name;  }  **public** **double** getTotal() {  **return** total;  }  **public** **double** getSports() {  **return** sports;  }  **public** Student(String name,**int** id,**double** total,**double** sports) {  **this**.name=name;  **this**.id=id;  **this**.total=total;  **this**.sports=sports;  }  }  （3）录取类：  **package** Luqu;  **import** Luqu.School;  **import** Luqu.Student;  **public** **class** Luqu {  **public** String luqu(Student stu,School sch) {  **if**((stu.getSports()>96)&&(stu.getTotal()>300)) {  System.***out***.println(stu.getName()+"被录取！");  }  **else** **if**(stu.getTotal()>sch.getScoreLine()) {  System.***out***.println(stu.getName()+"被录取！");  }  **else** {  System.***out***.println(stu.getName()+"未录取！");  }  **return** **null**;  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  School sch=**new** School();  sch.setScoreLine(483);  Student s1=**new** Student("陶喆",1,425,87);  Student s2=**new** Student("金晨",2,488,97);  Student s3=**new** Student("张艺兴",3,432,92);  Student s4=**new** Student("肖战",4,464,98);  Luqu l=**new** Luqu();  l.luqu(s1,sch);  l.luqu(s2,sch);  l.luqu(s3,sch);  l.luqu(s4,sch);  }  }  3.实验代码  **package** test;  **class** User  {  **public** String name;  **private** String password;  **private** **static** **int** *count*;//count用于统计个数必须用static 声明成全局属性  **public** User(){  *count*++;  }  **public** User(String name){  **this**.name=name;  *count*++;  }  **public** User(String name,String password){  **this**.name=name;  **this**.password=password;  *count*++;  }  **public** String getName(){  **return** name;  }  **public** **void** setName(String name){  **this**.name=name;  }  **public** String getPassword(){  **return** password;  }  **public** **void** setPassword(String password){  **this**.password=password;  }  **public** **void** print(){  System.***out***.println("用户名："+name+"\t"+"口令："+password+"\t"+"用户个数："+*count*);  }  **public** **void** count(){  System.***out***.println("用户总个数："+(*count*-1));//count()被调用时先执行了无参构造，故count需减1  }  }  **package** test;  **public** **class** test  {  **public** **static** **void** main(String args[]){  User user1=**new** User("李四","131415");  user1.print();  User user2=**new** User("张三","215451");  user2.print();  **new** User().count();  }  }  4.  （1）Point类：  **package** Graph;  **public** **class** Ponit {  **protected** **double** x;  **protected** **double** y;  **public** Ponit() {  x=y=0;  }  **public** Ponit(**double** x,**double** y) {  **this**.x=x;  **this**.y=y;  }  **public** **void** setX(**double** x) {  **this**.x = x;  }  **public** **double** getX() {  **return** x;  }  **public** **void** setY(**double** y) {  **this**.y = y;  }  **public** **double** getY() {  **return** y;  }  }  （2）Circle类：  **package** Graph;  **public** **class** Circle **extends** Ponit{  **public** **static** **final** **double** ***PI***=3.14;  **protected** **double** r;  **public** Circle(**double** x,**double** y,**double** r) {  **super**(x,y);//调用父类构造  **this**.r=r;  }  **public** Circle() {  r=0;  }  **public** **void** setR(**double** r) {  **this**.r = r;  }  **public** **double** getR() {  **return** r;  }  **public** **double** Area() {  **return** ***PI***\*r\*r;  }  }  （3）Cylinder类：  **package** Graph;  **public** **class** Cylinder **extends** Circle{  **protected** **double** h;  **public** Cylinder(**double** x,**double** y,**double** r,**double** h) {  **super**(x,y,r);  **this**.h=h;  }  **public** **void** setH(**double** h) {  **this**.h = h;  }  **public** **double** getH() {  **return** h;  }  **public** **double** Volume() {  **return** Circle.***PI***\*r\*r\*h;  }  }  （4）实例化测试  **package** Graph;  **import** Graph.Cylinder;  **public** **class** Graph {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Cylinder cy1=**new** Cylinder(1,3,1,4);  Cylinder cy2=**new** Cylinder(1,5,2,2);  Cylinder cy3=**new** Cylinder(5,2,4,6);  Cylinder cy4=**new** Cylinder(7,7,5,8);  System.***out***.println("cy1的圆心坐标："+"("+cy1.getX()+","+cy1.getY()+")");  System.***out***.println("cy1的半径："+cy1.getR());  System.***out***.println("cy1的高："+cy1.getH());  System.***out***.println("cy1的体积："+cy1.Volume());  System.***out***.println("-------------------------------------");  System.***out***.println("cy2的圆心坐标："+"("+cy2.getX()+","+cy2.getY()+")");  System.***out***.println("cy2的半径："+cy2.getR());  System.***out***.println("cy2的高："+cy2.getH());  System.***out***.println("cy2的体积："+cy2.Volume());  System.***out***.println("-------------------------------------");  System.***out***.println("cy3的圆心坐标："+"("+cy3.getX()+","+cy3.getY()+")");  System.***out***.println("cy3的半径："+cy3.getR());  System.***out***.println("cy3的高："+cy3.getH());  System.***out***.println("cy3的体积："+cy3.Volume());  System.***out***.println("-------------------------------------");  System.***out***.println("cy4的圆心坐标："+"("+cy4.getX()+","+cy4.getY()+")");  System.***out***.println("cy4的半径："+cy4.getR());  System.***out***.println("cy4的高："+cy4.getH());  System.***out***.println("cy4的体积："+cy4.Volume());  System.***out***.println("-------------------------------------");  }  } |
| 实验结果  1.    2.    3.    4. |
| 结果分析与结论  本次实验中出现了未找到主类的错误，经过多次不同方法的尝试，最终通过重建工程解决了。本次实验中还对this关键字进行了使用，this在类中就是代表当前对象，可以通过this关键字完成当前对象的成员属性、成员方法和构造方法的调用。在定义类中方法时，如果需要调用该类对象，就可以使用this表示该对象。例如在带参构造方法中对参数进行赋值时，name=name;这个语句完成的是赋值给参数本身，而非类中的属性name。用this();完成构造函数间相互调用时，this();只能放在构造函数的第一行。  实验中还通过查阅资料获得了Eclipse中自动生成get和set方法的三种办法。第一种为：1.在声明的类中，单击鼠标右键，选择source；2.选择Generate Getters and Setters；3.根据需要配置，如位置，顺序等，确定即可。第二种为：1.在声明的数据域中按Ctrl+1；2.点击最后一个选项Create getter and setter，在弹出的对话框中点击确定。第三种：1.声明完类的数据域之后，输入set，按住Alt+/；2.同步骤一，输入get，再按住Alt+/。  本次实验中还涉及到了super关键字，super关键字是Java语言中类和继承部分所使用的，它与this关键字有着异曲同工之处，super关键字主要是子类用于访问父类成员属性、访问父类成员方法、访问父类构造方法。子类的构造过程中必须调用其父类的构造方法，如果子类的构造方法中未显式调用父类的构造方法，则系统默认调用父类无参构造。使用super调用父类的构造方法必须放在子类的构造方法的第一行。如果子类构造方法中未显式调用父类构造且父类无无参构造，编译就会出错。构造方法调用时，super和this不能同时出现。  this关键字：当前类对象的引用，可以访问当前类成员方法、成员属性、构造方法，不能在静态方法中使用。  super关键字：父类对象的引用，可以访问父类成员属性、成员方法、构造方法，不能在静态方法中使用。 |
| 教师评阅及成绩  教师签名：  年 月 日 |

实 验 报 告

课程名称: Java语言程序设计 姓 名: 安民

学 号: P181713164

班 级: 2018级计算机科学与技术1班

数学与计算机科学学院

西北民族大学数学与计算机科学学院实验报告

**实验名称：输入与输出**

**指导教师**：蔡正琦 **日期：**2020年 5月11日

|  |
| --- |
| 实验内容  1．利用流在屏幕上显示文本文件内容及文件的路径、修改时间、大小、长度、可读性和可修改性等属性。  2．利用流向文本文件添加记录并显示其记录内容。 |
| 实验目的  （1）熟悉流的操作。  （2）应用流读写磁盘文件。 |
| 实验设备  硬件：x64 处理器，内存8GB  软件：Win10操作系统，IDE Eclipse 2019-12 |
| 实验过程  1.实验代码  **package** experiment3\_1;  **import** java.io.\*;  **public** **class** MyFile  {  **public** **static** **void** main(String args[]) **throws** IOException  {  String filename=**null**;  **if**(args.length>0) filename=args[0];  **else**  {  System.***out***.println("无文件名");  **return**;  }  File file=**new** File(filename);  **if**(!file.exists())  {  System.***out***.println("文件不存在");  **return**;  }  FileReader fread=**new** FileReader(file);  BufferedReader bread=**new** BufferedReader(fread);  String s=bread.readLine();  **while**(s!=**null**)  {  System.***out***.println(s);  s=bread.readLine();  }  fread.close();  System.***out***.println("文件路径："+file.getAbsolutePath());  System.***out***.println("修改时间："+file.lastModified());  System.***out***.println("文件长度："+file.length());  System.***out***.println("文件可读："+file.canRead());  System.***out***.println("文件可写："+file.canWrite());  }  }  2.实验代码  **package** experiment3\_1;  **import** java.io.\*;  **public** **class** FileAdd  {  **private** **static** File *file*;  **private** **static** FileReader *fread*;  **private** **static** BufferedReader *bread*;  **private** **static** FileWriter *fwrite*;  **private** **static** BufferedWriter *bwrite*;  **private** **static** BufferedReader *keyread*;  **public** **static** **void** readFile()**throws** Exception  {  System.***out***.println("文件内容");  String s=*bread*.readLine();  **while**(s!=**null**)  {  System.***out***.println(s);  s=*bread*.readLine();  }  }  **private** **static** **void** writeFile() **throws** Exception  {  String s;  **while**(**true**)  {  System.***out***.println("输入字符串：");  System.***out***.flush();  s=*keyread*.readLine();  **if**(s.length()==0) **break**;  *bwrite*.write(s);  *bwrite*.newLine();  }  }  **public** **static** **void** main(String args[]) **throws** Exception  {  String filename=**null**;  **if**(args.length>0) filename=args[0];  **else**  {  System.***out***.println("无文件名");  **return**;  }  *file*=**new** File(filename);  **if**(!*file*.exists())  {  System.***out***.println("文件不存在");  **return**;  }  *fwrite*=**new** FileWriter(*file*);  *bwrite*=**new** BufferedWriter(*fwrite*);  *keyread*=**new** BufferedReader(**new** InputStreamReader(System.***in***));  *writeFile*();  *keyread*.close();  *bwrite*.close();  *fread*=**new** FileReader(*file*);  *bread*=**new** BufferedReader(*fread*);  *readFile*();  *bread*.close();    }  } |
| 实验结果  1.    2. |
| 结果分析与结论  通过本次实验我知道了Java中用IO流来处理设备之间的数据传输，传输是通过流的方式，Java用于操作流的对象都在java.io包中，流就是指一连串流动的字符，以先进先出的方式发送信息的通道。按照流的方向将流划分为输入流与输出流。输入流：将外部数据源的数据转换成流，程序通过读取流中的数据，完成对数据源读取的访问。输出流：将流中的数据转换到对应的数据源中，程序通过向流中写入数据，完成对数据源写入。  例如System.out.println("cocci"); 就是将字符串输出到控制台中，或者说输出到屏幕中，这里的屏幕就是输出设备，由程序将cocci通过流输出到目的地。输出设备除了屏幕，还有打印机、文件等。  使用键盘接收数据 Scanner sc = new Scanner(System.in); 这里的System.in就是输入流，程序从数据源这里指键盘去读取数据通过流输入到程序当中。输入设备除了键盘，还有扫描仪、文件等。  按照类型将流划分为字节流与字符流。字节流：以字节为单位(8位)，可以访问所有文件。字符流：以字符为单位(16位Unicode)，只能访问文本文件。 |
| 教师评阅及成绩  教师签名：  年 月 日 |

实 验 报 告

课程名称: Java语言程序设计 姓 名: 安民

学 号: P181713164

班 级: 2018级计算机科学与技术1班

数学与计算机科学学院

西北民族大学数学与计算机科学学院实验报告

**实验名称：图形用户界面设计**

**指导教师**：蔡正琦 **日期：**2020年 5月25日

|  |
| --- |
| 实验内容  1．编写文本移动程序，窗口中有两个文本区和两个按钮，文本区分别位于窗口的左边和右边区域，两个按钮位于窗口的中间区域，当单击“→”按钮时，将左边文本区中选中的内容添加到右边文本区的末尾；当单击“←”按钮时，将右边文本区中选中的内容添加到左边文本区的末尾。  2．编写一个简单的个人简历录入程序。可以通过 输入姓名，通过单选按钮选择性别，通过组合框选择籍贯和文化程度，并在文本框中显示所填计选择的信息。 |
| 实验目的  （1）熟悉AWT标签、文本框、文本行、按钮等组件的使用方法。  （2）熟悉Swing组件的使用方法。  （3）熟悉事件处理方法。 |
| 实验设备  硬件：x64 处理器，内存8GB  软件：Win10操作系统，IDE Eclipse 2019-12 |
| 实验过程   1. 实验代码   **package** experiment4\_1;  **import** java.awt.\*;  **import** java.awt.event.\*;  **public** **class** Experiment4\_1 **extends** Frame {  **private** TextArea eastArea = **new** TextArea(7, 20);  **private** TextArea westArea = **new** TextArea(7, 20);  **private** Button toLeft = **new** Button("<--");  **private** Button toRight = **new** Button("-->");  **public** Experiment4\_1() {  **super**("MoveWord");  **this**.setLayout(**new** FlowLayout());  **this**.add(westArea);  Panel pal = **new** Panel();  pal.setLayout(**new** GridLayout(2, 1, 10, 10));  pal.add(toLeft);  pal.add(toRight);  toLeft.addActionListener(**new** Handler());  toRight.addActionListener(**new** Handler());  **this**.add(pal);  **this**.add(eastArea);  addWindowListener(**new** WindowAdapter() {  **public** **void** windowClosing(WindowEvent e) {  System.*exit*(0);  }  });  setSize(400, 200);  setVisible(**true**);  }  **class** Handler **implements** ActionListener {  **public** **void** actionPerformed(ActionEvent e) {  String copyText = "";  **if** (e.getSource() == toLeft) {  copyText = eastArea.getSelectedText();  westArea.append(copyText);  } **else** {  copyText = westArea.getSelectedText();  eastArea.append(copyText);  }  }  }  **public** **static** **void** main(String[] args) {  Experiment4\_1 word = **new** Experiment4\_1();  }  2.实验代码  **package** experiment4\_1;  **import** java.awt.\*;  **import** javax.swing.\*;  **import** java.awt.event.\*;  **class** SimpleResume **extends** JFrame {  **private** JLabel name;  **private** JTextField nameInput;  **private** JLabel sex;  **private** JRadioButton male;  **private** JRadioButton female;  **private** ButtonGroup sexSelect;  **private** JLabel provinceLab;  **private** String province[] = { "北京市", "陕西省", "河南省" };  **private** JComboBox provinceBox;  **private** JLabel degreeLab;  **private** String degree[] = { "中学", "本科", "硕士", "博士", "其他" };  **private** JComboBox degreeList;  **private** JButton ok, cancel;  **private** String[] str = **new** String[4];  **private** String output = "";  **private** JTextArea txa;  **public** SimpleResume() {  **super**("简单的个人简历程序");  Container c = **this**.getContentPane();  c.setLayout(**new** FlowLayout());  name = **new** JLabel("姓名：");  name.setHorizontalAlignment(SwingConstants.***CENTER***);  nameInput = **new** JTextField(8);  sex = **new** JLabel("性别：");  sex.setHorizontalAlignment(SwingConstants.***CENTER***);  male = **new** JRadioButton("男", **true**);  female = **new** JRadioButton("女", **false**);  sexSelect = **new** ButtonGroup();  sexSelect.add(male);  sexSelect.add(female);  provinceLab = **new** JLabel("籍贯：");  provinceLab.setHorizontalAlignment(SwingConstants.***CENTER***);  provinceBox = **new** JComboBox(province);  degreeLab = **new** JLabel("文化程度：");  degreeLab.setHorizontalAlignment(SwingConstants.***CENTER***);  degreeList = **new** JComboBox(degree);  ok = **new** JButton("确定");  cancel = **new** JButton("取消");  txa = **new** JTextArea(5, 20);  JPanel namePan = **new** JPanel();  namePan.add(name);  namePan.add(nameInput);  c.add(namePan);  JPanel sexPan = **new** JPanel();  sexPan.add(sex);  sexPan.add(female);  c.add(sexPan);  JPanel provincePan = **new** JPanel();  provincePan.add(provinceLab);  provincePan.add(**this**.provinceBox);  c.add(provincePan);  JPanel degreePan = **new** JPanel();  degreePan.add(degreeLab);  degreePan.add(degreeList);  c.add(degreePan);  JPanel buttonPan = **new** JPanel();  buttonPan.add(ok);  buttonPan.add(cancel);  ok.addActionListener(**new** Handle1());  cancel.addActionListener(**new** Handle1());  c.add(buttonPan);  c.add(txa);  **this**.setSize(350, 280);  **this**.setVisible(**true**);  }  **public** **static** **void** main(String args[]) {  SimpleResume resume = **new** SimpleResume();  resume.addWindowListener(**new** WindowAdapter() {  **public** **void** windowClosing(WindowEvent event) {  System.*exit*(0);  }  });  }  **private** **class** Handle1 **implements** ActionListener {  **public** **void** actionPerformed(ActionEvent e) {  **if** (e.getSource() == ok) {  str[0] = "姓名：" + nameInput.getText();  **if** (male.isSelected())  str[1] = "性别：男";  **else**  str[1] = "性别：女";  str[2] = "籍贯：" + province[provinceBox.getSelectedIndex()];  str[3] = "文化程度：" + province[degreeList.getSelectedIndex()];  output = "";  **for** (**int** i = 0; i < 4; i++)  output = output + str[1] + "\n";  txa.setText(output);  }  **if** (e.getSource() == cancel)  System.*exit*(0);  }  }  } |
| 实验结果  1.    2. |
| 结果分析与结论  通过本次实验对Java图形界面编程的知识有了基础了解。  1.JFrame 是容器类。  2.AWT 是抽象窗口组件工具包，是 Java 最早的用于编写图形节目应用程序的开发包。  3.Swing 是为了解决 AWT 存在的问题而新开发的包，它以 AWT 为基础的。  Swing提供许多比AWT更好的屏幕显示元素。它们用纯Java写成，所以同Java本身一样可以跨平台运行，这一点不像AWT。它们是JFC的一部分。它们支持可更换的面板和主题（各种操作系统默认的特有主题），然而不是真的使用原生平台提供的设备，而是仅仅在表面上模仿它们。这意味着你可以在任意平台上使用JAVA支持的任意面板。轻量级组件的缺点则是执行速度较慢，优点就是可以在所有平台上采用统一的行为。  通过查阅资料，我知道了：  Jframe：java的GUI程序的基本思路是以JFrame为基础，它是屏幕上window的对象，能够最大化、最小化、关闭。  Jpanel：Java图形用户界面(GUI)工具包swing中的面板容器类，包含在javax.swing 包中，可以进行嵌套，功能是对窗体中具有相同逻辑功能的组件进行组合，是一种轻量级容器，可以加入到JFrame窗体中。  JLabel：JLabel 对象可以显示文本、图像或同时显示二者。可以通过设置垂直和水平对齐方式，指定标签显示区中标签内容在何处对齐。默认情况下，标签在其显示区内垂直居中对齐。默认情况下，只显示文本的标签是开始边对齐；而只显示图像的标签则水平居中对齐。  JtextField：一个轻量级组件，它允许编辑单行文本。  JpasswordField：允许我们输入了一行字像输入框，但隐藏星号(\*) 或点创建密码(密码)。  Jbutton：JButton 类的实例。用于创建按钮类似实例中的 "Login"。 |
| 教师评阅及成绩  教师签名：  年 月 日 |

实 验 报 告

课程名称: Java语言程序设计 姓 名: 安民

学 号: P181713164

班 级: 2018级计算机科学与技术1班

数学与计算机科学学院

西北民族大学数学与计算机科学学院实验报告

**实验名称：多线程**

**指导教师**：蔡正琦 **日期：**2020年 6月22日

|  |
| --- |
| 实验内容  1．编写具有两个线程的程序，第一个线程用来计算2-1000000之间的质数及个数，第二个线程用来计算1000000-2000000之间的质数及个数。  2．编写多线程程序，显示目前正在运行的线程名和目前的活动线程数。 |
| 实验目的  （1）熟悉利用Thread类建立多线程的方法。  （2）熟悉利用Runnable接口建立多线程的方法。 |
| 实验设备  硬件：x64 处理器，内存8GB  软件：Win10操作系统，IDE Eclipse 2019-12 |
| 实验过程  1.实验代码  **package** experiment4\_1;  **class** UseThread **extends** Thread {  **int** st, en, count = 0;  **public** UseThread(**int** m, **int** n) {  **this**.st = m;  **this**.en = n;  }  **public** **void** run() {  **for** (**int** m = st; m <= en; m++) {  **boolean** isPrime = **true**;  **for** (**int** i = 2; i < m; i++)  **if** (m % i == 0) {  isPrime = **false**;  **break**;  }  **if** (isPrime)  count++;  }  System.***out***.println(st + "~" + en + "之间共有" + count + "个质数");  }  **public** **static** **void** main(String[] args) {  UseThread thread1 = **new** UseThread(2, 1000000);  UseThread thread2 = **new** UseThread(1000000, 2000000);  thread1.start();  thread2.start();  }  }  2.实验代码  **package** experiment4\_1;  **public** **class** Experiment4\_2 {  **public** **static** **void** main(String[] args) {  MyThread t1 = **new** MyThread("T1");  MyThread t2 = **new** MyThread("T2");  t1.start();  t2.start();  System.***out***.println("活动线程数：" + Thread.*activeCount*());  System.***out***.println("main()运行完毕");  }  }  **class** MyThread **extends** Thread {  **public** MyThread(String s) {  **super**(s);  }  **public** **void** run() {  **for** (**int** i = 1; i <= 3; i++) {  System.***out***.println(getName() + "第" + i + "次运行");  **try** {  *sleep*((**int**) (Math.*random*() \* 100));  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  System.***out***.println(getName() + "结束");  }  } |
| 实验结果  1.    2. |
| 结果分析与结论  本次实验中知道了进程是程序的一次动态执行过程，它需要经历从代码加载，代码执行到执行完毕的一个完整的过程，这个过程也是进程本身从产生，发展到最终消亡的过程。多进程操作系统能同时达运行多个进程（程序），由于 CPU 具备分时机制，所以每个进程都能循环获得自己的CPU 时间片。由于 CPU 执行速度非常快，使得所有程序好像是在同时运行一样。  多线程是实现并发机制的一种有效手段。进程和线程一样，都是实现并发的一个基本单位。线程是比进程更小的执行单位，线程是进程的基础之上进行进一步的划分。所谓多线程是指一个进程在执行过程中可以产生多个更小的程序单元，这些更小的单元称为线程，这些线程可以同时存在，同时运行，一个进程可能包含多个同时执行的线程。  在 Java 中实现多线程有两种手段，一种是继承 Thread 类，另一种就是实现 Runnable 接口。从 Thread 类的定义可以清楚的发现，Thread 类也是 Runnable 接口的子类，但在Thread类中并没有完全实现 Runnable 接口中的 run() 方法，下面是 Thread 类的部分定义。  Private Runnable target；  public Thread(Runnable target,String name){  init(null,target,name,0);  }  private void init(ThreadGroup g,Runnable target,String name,long stackSize){  ...  this.target=target;  }  public void run(){  if(target!=null){  target.run();  }  }  从定义中可以发现，在 Thread 类中的 run() 方法调用的是 Runnable 接口中的 run() 方法，也就是说此方法是由 Runnable 子类完成的，所以如果要通过继承 Thread 类实现多线程，则必须覆写 run()。  实际上 Thread 类和 Runnable 接口之间在使用上也是有区别的，如果一个类继承 Thread类，则不适合于多个线程共享资源，而实现了 Runnable 接口，就可以方便的实现资源的共享。要想实现多线程，必须在主线程中创建新的线程对象。任何线程一般具有5种状态，即创建，就绪，运行，阻塞，终止。  （1）创建状态  在程序中用构造方法创建了一个线程对象后，新的线程对象便处于新建状态，此时它已经有了相应的内存空间和其他资源，但还处于不可运行状态。新建一个线程对象可采用Thread 类的构造方法来实现，例如 “Thread thread=new Thread()”。  （2）就绪状态  新建线程对象后，调用该线程的 start() 方法就可以启动线程。当线程启动时，线程进入就绪状态。此时，线程将进入线程队列排队，等待 CPU 服务，这表明它已经具备了运行条件。  （3）运行状态  当就绪状态被调用并获得处理器资源时，线程就进入了运行状态。此时，自动调用该线程对象的 run() 方法。run() 方法定义该线程的操作和功能。  （4）阻塞状态  一个正在执行的线程在某些特殊情况下，如被人为挂起或需要执行耗时的输入/输出操作，会让 CPU 暂时中止自己的执行，进入阻塞状态。在可执行状态下，如果调用sleep(),suspend(),wait() 等方法，线程都将进入阻塞状态，发生阻塞时线程不能进入排队队列，只有当引起阻塞的原因被消除后，线程才可以转入就绪状态。  （5）死亡状态  线程调用 stop() 方法时或 run() 方法执行结束后，即处于死亡状态。处于死亡状态的线程不具有继续运行的能力。  Java 程序每次运行至少启动两个线程，每当使用 Java 命令执行一个类时，实际上都会启动一个 JVM，每一个JVM实际上就是在操作系统中启动一个线程，Java 本身具备了垃圾的收集机制。所以在 Java 运行时至少会启动两个线程，一个是 main 线程，另外一个是垃圾收集线程。  取得和设置线程的名称：  class MyThread implements Runnable{ //实现Runnable接口  public void run(){  for(int i=0;i<3;i++){  System.Out.Println(Thread.currentThread().getName()+"运行, i="+i); //取得当前线程的名称  }  }  };  public class ThreadDemo{  public static void main(String args[]){  MyThread my=new MyThread(); //定义Runnable子类对象  new Thread(my).start; //系统自动设置线程名称  new Thread(my,"线程A").start(); //手工设置线程名称  }  }; |
| 教师评阅及成绩  教师签名：  年 月 日 |