附件3：

湖南科技大学计算机科学与工程学院

面向对象程序设计 实验报告

**专业班级：** 22级计科七班

**姓 名：** 周俊哲

**学 号：** 2205010711

**指导教师：** 彭珍连

**时 间**：

**地 点**： 逸夫楼418

|  |  |  |  |
| --- | --- | --- | --- |
| 指导教师评语：    **成绩： 等级：**  **签名：**  **年 月 日** | | | |
| 实验名称 | | Java入门—方法的使用 | | | |
| 实验性质  （必修、选修） | |  | 实验类型（验证、设计、创新、综合） |  | |
| 实验课时 | |  | 实验日期 | 1、  2、  3、  4、  。。。 | |
| 实验仪器设备以及实验软硬件要求 | |  | | | |
| 实验目的 | |  | | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法）  第3关： 掌握有参数无返回值方法的调用  算法思想:求平均数:读入两个数据,求和后除以它们的数量  Scanner sc = new Scanner(System.in);  int score1 = sc.nextInt(); //第一门成绩  int score2 = sc.nextInt(); //第二门成绩  System.out.println("平均分："+(score1+score2)/2); 第4关：掌握有参数有返回值方法的调用 算法思想:求最大值,读入两个数据,利用if语句判断哪个数大,然后输出  Scanner sc = new Scanner(System.in);  int i = sc.nextInt(); //获取第一个输入的整数  int j = sc.nextInt(); //获取第二个输入的整数  if(i>j) max=i;  else max=j; | | | | | |
| 实验结果与分析  学会了在java中如何使用方法 | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java面向对象—类与对象 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第1关：什么是类，如何创建类 算法思想:利用class关键字创建类,类中可以声明变量和方法  class Dog{  public String name;  public String color;  public String variety;  public Dog(String name,String color,String variety){  this.name=name;  this.color=color;  this.variety=variety;  }  public void eat()  {  System.out.println("啃骨头");  }  public void run(){  System.out.println("叼着骨头跑");  }  } 第5关：类与对象练习 算法思想:声明类,利用类中的构造方法来创建对象  public WuMingFen(String a,int b, boolean c){  theMa=a;  quantity=b;  likeSoup=c;  } | | | |
| 实验结果与分析  认识了java中如何创建类和使用类 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java面向对象 - 封装、继承和多态 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第1关：什么是封装，如何使用封装 算法思想:将类中的变量私有化,利用get和set函数封装变量  class Person {  /\*\*\*\*\*\*\*\*\* begin \*\*\*\*\*\*\*\*\*/  private String name;  private int age;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  } 第2关：什么是继承，怎样使用继承 算法思想:利用extends关键字来继承父类  class Dog extends Animal {  Dog(String name,int age){  this.name=name;  this.age=age;  } | | | |
| 实验结果与分析  认识了java中如何使用封装和继承 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java面向对象 - String类 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第1关：length()方法与compareTo()方法的使用 - 花名册 算法思想:利用length方法得到String变量的长度,然后利用compareTo方法判断数据是否相同  Scanner scanner = new Scanner(System.in);  /\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*/  String s1=scanner.nextLine();  String s2=scanner.nextLine();  System.out.println(s1.length());  System.out.println(s2.length());  int a=s1.compareTo(s2);  if(a!=0)  System.out.println("不相同");  else  System.out.print("相同"); 第2关：substring()方法与indexOf()方法的使用 - 姓名查找 算法思想:利用substring截取字符串,然后利用indexOf来获取索引  Scanner scanner = new Scanner(System.in);  /\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*/  String data=scanner.nextLine();  String name=scanner.nextLine();  int i=0,a=0;  while(i<=data.length()){  a=data.indexOf(name,i);  if(a!=-1){  System.out.println(a);  i=a+1;  }  else  i++;  } | | | |
| 实验结果与分析  认识了java中String类的使用方法 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java面向对象-Java中的异常 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第2关：捕获异常 算法思想:利用try-catch语句块来捕获异常  try{  System.out.println(num1/num2);}  catch(ArithmeticException e){  System.out.print("除数不能为0");  } 第3关：抛出异常 算法思想:利用throws关键字抛出异常  public static void test() throws FileNotFoundException{  File file = new File("abc");  if(!file.exists()){ //判断文件是否存在  //文件不存在，则 抛出 文件不存在异常  throw new FileNotFoundException("该文件不存在");  }else{  FileInputStream fs = new FileInputStream(file);  }  } | | | |
| 实验结果与分析  认识了java中如何对异常进行处理 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | java常用集合 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第2关：集合遍历方法 算法思想:利用迭代器和foreach来遍历集合  Iterator iterator = arrayListExample.iterator();  while (iterator.hasNext()) {  System.out.println(iterator.next());  }  for (Object ele : arrayListExample) {  System.out.println(ele);  } 第3关：Set接口 算法思想:创建set,  HashSet<Object> hashSet = new HashSet<>();  hashSet.add("helloHashSet");  hashSet.remove("apple");  if(hashSet.contains("cat")) {  hashSet.remove("cat");  hashSet.add("fish");  } | | | |
| 实验结果与分析  认识了java中的集合 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java高级特性 - IO流 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第2关：字节流-输入输出 算法思想:创建文件和文件输入流,利用字节数组来读入文件中的内容,创建输出流,利用字节数组将数据写入文件中  File file = new File("src/step2/input/task.txt");  FileInputStream fs = new FileInputStream(file); //定义一个文件输入流  byte[] b = new byte[8]; //定义一个字节数组  fs.read(b); //将输入流的数据读入到字节数组  String str = new String(b,"UTF-8"); //通过utf-8编码表将字节转换成字符  FileOutputStream out = new FileOutputStream("src/step2/output/output.txt"); //创建输出流  String str1 = "learning practice";  byte[] c = str1.getBytes(); //将字符串转换成字节  out.write(c); //写数据  out.flush(); //刷新缓存区数据  fs.close(); //释放资源 第3关：字符流 - 输入输出 算法思想:创建文件和字符输入流,利用字节数组来读入文件中的内容,创建输出流,利用字节数组将数据写入文件中  String file1 = "src/step3/input/input.txt";  FileReader fr = new FileReader(file1);  char[] ch = new char[8];  fr.read(ch);  String file2 = "src/step3/output/output.txt";  FileWriter fw = new FileWriter(file2);  fw.write(ch);  fr.close();  fw.flush();  fw.close(); | | | |
| 实验结果与分析  认识了java中的IO流 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java高级特性 - 多线程基础（1）使用线程 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第1关：创建线程 算法思想:继承Thread类来创建线程  public class ThreadClassOne extends Thread {  public int i=0;  public ThreadClassOne(){  super();  } 第2关：使用 Callable 和 Future 创建线程 算法思想:继承Callable接口来实现线程类,然后利用FutureTask来执行线程  Callable<Integer> callable = new ThreadCallable(num);  FutureTask<Integer> futureTask = new FutureTask<>(callable);  new Thread(futureTask).start();//开启线程  try {  Integer result = futureTask.get();  System.out.println("线程的返回值为：" + result);  } catch (Exception e) {  e.printStackTrace();  } | | | |
| 实验结果与分析  认识了java中的多线程 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java高级特性 - 多线程基础（2）常用函数 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第2关：常用函数（一） 算法思想:创建一个新线程,利用sleep函数和join函数来执行任务  class MyThread extends Thread {  private int num;  private String name;  public MyThread(String name,int num){  this.num=num;  this.name=name;  }  public void run(){  int[] arr=new int[2];  arr[0]=1;  arr[1]=1;  for(int i=2;i<num;i++){  int tmp=arr[1];  arr[1]=arr[0]+arr[1];  arr[0]=tmp;  }  System.out.println("子线程计算结果为："+arr[1]); }第3关：常用函数（二） 算法思想:继承Runnable接口,使用synchronized关键字来封锁,最后使任务正常执行  public void run(){  int count = 5;  while(count>0){  synchronized(prev){  synchronized(self){  System.out.print(name);  count--;  self.notify();  }  try {  prev.wait();  } catch (InterruptedException e) {  e.printStackTrace();  }  }  } | | | |
| 实验结果与分析  对多线程的使用有了更多认识 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | Java高级特性 - 多线程基础（3）线程同步 | | |
| 实验性质  （必修、选修） |  | 实验类型（验证、设计、创新、综合） |  |
| 实验课时 |  | 实验日期 | 1、  2、  3、  4、  。。。 |
| 实验仪器设备以及实验软硬件要求 |  | | |
| 实验目的 |  | | |
| 实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法） 第2关：使用synchronized关键字同步线程 算法思想:利用synchronized关键字对线程任务加锁来保证线程的正常运行  public synchronized void insert(Thread thread){  for (int i = 0; i <= 5; i++) {  num++;  System.out.println(num);  }  } 第3关：使用线程锁（Lock）实现线程同步 算法思想: 利用Lock类型变量对线程任务加锁来保证线程的正常运行  lock.lock();    try {  System.out.println(thread.getName()+"得到了锁");  for (int i = 0; i < 5; i++) {  num++;  System.out.println(num);  }  } catch (Exception e) {  // TODO: handle exception  }finally {  System.out.println(thread.getName()+"释放了锁");  lock.unlock();  } | | | |
| 实验结果与分析  对多线程的线程同步有了更深的理解 | | | |