**《面向对象程序设计(Java)》**

**实验报告**

姓 名 游李滨

学 号 2020081142

学 院 软件工程学院

专 业 软件工程

年 级 2020级

指导教师 曹亮

**成都信息工程大学软件工程学院**

**2019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 (填写你自己的实验环境)**

JDK1.8, 命令行

1. **设计思路与具体实现**

1.

1. 设计思路

使用System.out.println()输出hello world

(2)实验步骤

①、创建Hello.java文件，并在main方法里编写输出语句 System.out.println(“Hello World!”)；

②、使用命令行javac Hello.java编译

③、使用java Hello.class运行文件

(3) 关键代码

System.out.println(“Hello World!”);

(4) 程序测试过程

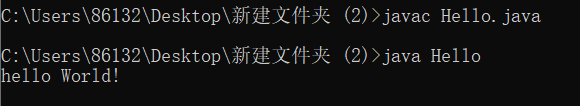


编译通过



运行结果正确

(5) 运行结果（运行效果）



2.

(1) 设计思路

命令行中输入的数字，利用args[]的形式得到字符串，再利用Integer.valueOf( )方法将字符串转为整型进行计算

(2)实验步骤

①、创建java文件，并编写代码

②、使用命令行编译得到class文件

③、运行class文件

(3) 关键代码

**int** number1=Integer.*valueOf*(args[0]);

**int** number2=Integer.*valueOf*(args[1]);

**int** number3=Integer.*valueOf*(args[2]);

System.***out***.println((number1+number2+number3)/3);

(4) 程序测试过程

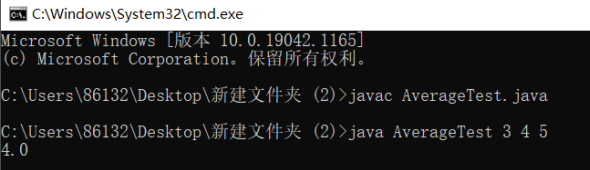


编译通过



程序运行成功，且结果符合预期

1. 运行结果（运行效果）



1. **实验总结**
2. 可以使用System.out.prinln()；输出字符串内容
3. Args[]可以访问命令行字符串
4. Integer.valueof（String）可以将字符串转换成Integer类型
5. Integer类型会自动类型转换成int类型
6. **附录（程序代码）**
7. public class Hello{

public static void main(String[] args) {

System.out.prinln(“Hello World!”);

}

}

1. public class AverageTest {

public static void main(String[] args) {

int number1=Integer.*valueOf*(args[0]);

int number2=Integer.*valueOf*(args[1]);

int number3=Integer.*valueOf*(args[2]); System.*out*.println((number1+number2+number3)/3);

}

}

# 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**

Eclipse，JDK1.8

1. **设计思路与具体实现**

1.

1. 设计思路

提示输入摄氏度,用户输入摄氏度使用Scanner类，计算华氏温度，System.out.println()输出华氏温度

(2)实验步骤

①、请求用户输入

②、计算华氏温度

③、输出结果

(3) 关键代码

Scanner scanner=new Scanner(System.*in*);

System.*out*.println("请输入摄氏温度:");

float sheShi=scanner.nextFloat();

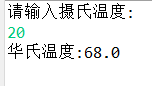
double huaShi=32+sheShi \*1.8;

System.*out*.println("华氏温度:"+huaShi);

1. 程序测试过程



(5) 运行结果（运行效果）



2.

(1) 设计思路

定义数组并赋值，使用选择排序方法进行排序

(2)实验步骤

①、定义数组并赋值

②、使用选择排序方式对数组内容进行排序

③、输出结果

(3) 关键代码

**for**(i=0;i<array.length-1;i++)

{

**int** min=i;

**for**(**int** j=i+1;j<array.length;j++)

{

**if**(array[min]>array[j])

{

min=j;

}

**if**(i!=min)

{

**int** temp;

temp=array[i];

array[i]=array[min];

array[min]=temp;

}

}

}

(4) 程序测试过程



**输出检查**



**选择排序后**



(5) 运行结果（运行效果）



3.

（1）设计思路

建立数组，数组大小由Scanner用户输入决定，通过循环枚举法将数n从1开始至无限，若arrra[i]等于array[i-1]+array[i-2]，则将n加入数组，当前索引i++，最多i至array.length-1

(2)实验步骤

①、建立数组，判断输入数组长度

②、循环枚举寻找，并添加进数组

③、输出结果

1. 关键代码

第一种： **int** index=2;

array[0]=array[1]=1;

**while**(index<=array.length-1)

{

array[index]=array[index-1]+array[index-2];

index++;

}

**for**(**long** x : array)

{

System.***out***.print(x+" ");

}

第二种：int index=2;

array[0]=array[1]=1;

while(index<=array.length-1)

{

for(long j=1;;j++)

{

if(j==(array[index-2]+array[index-1]))

{

array[index]=j;

++index;

break;

}

}

}

1. 程序测试过程

输入0报错



输入1



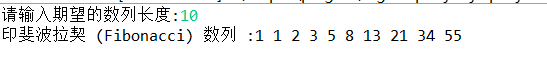
输入2



输入5



(5) 运行结果（运行效果）



1. **实验总结**
2. scanner的使用方法
3. 选择排序的方法
4. 印斐波拉契 (Fibonacci) 数列
5. 增强for()遍历数组
6. **附录（程序代码）**

（1）Scanner scanner=new Scanner(System.*in*);

System.*out*.println("请输入摄氏温度:");

float sheShi=scanner.nextFloat();

double huaShi=32+sheShi \*1.8;

System.*out*.println("华氏温度:"+huaShi);

（2）

int []array= {5,3,4,1,2};

int i;

System.*out*.print("选择排序前: ");

for(i=0;i<array.length;i++)

{

System.*out*.print(array[i]+" ");

}

System.*out*.print("\n选择排序后: ");

for(i=0;i<array.length-1;i++)

{

int max=i;

for(int j=i+1;j<array.length;j++)

{

if(array[j]>array[max])

{

max=j;

}

if(i!=max)

{

int temp;

temp=array[i];

array[i]=array[max];

array[max]=temp;

}

}

}

for(int a : array)

{

System.*out*.print(a+" ");

}

（3）

第一种：递归

System.*out*.print("请输入期望的数列长度:");

Scanner scanner=new Scanner(System.*in*);

int n=scanner.nextInt();

long []array=new long[n];

System.*out*.print("印斐波拉契 (Fibonacci) 数列 :");

if(n<3)

{

if(n==0){ System.*out*.print("error: Should Be n>0");return 0; }

if(n==1) { System.*out*.print(1);return 0; }

if(n==2) { System.*out*.print(1+" "+1);return 0;}

}else {

int index=2;

array[0]=array[1]=1;

while(index<=array.length-1)

{

array[index]=array[index-1]+array[index-2];

index++;

}

for(long x : array)

{

System.*out*.print(x+" ");

}

}

return 0;

第二种：枚举法

System.*out*.print("请输入期望的数列长度:");

Scanner scanner=new Scanner(System.*in*);

int n=scanner.nextInt();

long []array=new long[n];

System.*out*.print("印斐波拉契 (Fibonacci) 数列 :");

if(n<3)

{

if(n==0){ System.*out*.print("error: Should Be n>0");return 0; }

if(n==1) { System.*out*.print(1);return 0; }

if(n==2) { System.*out*.print(1+" "+1);return 0;}

}else {

int index=2;

array[0]=array[1]=1;

while(index<=array.length-1)

{

for(long j=1;;j++)

{

if(j==(array[index-2]+array[index-1]))

{

array[index]=j;

++index;

break;

}

}

}

for(long x : array)

{

System.*out*.print(x+" ");

}

}

return 0;

# 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**

JDK，eclipse

1. **设计思路与具体实现**

1. （1）设计思路

变量内半径 innerRadius, 外半径 outerRadius 和颜色 color可以set也可以get，则设置为private，面积由公式S=PAI\*（R\*R-r\*r）求得。

(2)实验步骤

①、创建Ring类，建立变量

②、编写set、get、ringArea方法

③、创建对象测试

(3) 关键代码

public double getArea() {

return *PAI*\*(outerRadius\*outerRadius-innerRadius\*innerRadius);

}

public String getColor() {

return color;

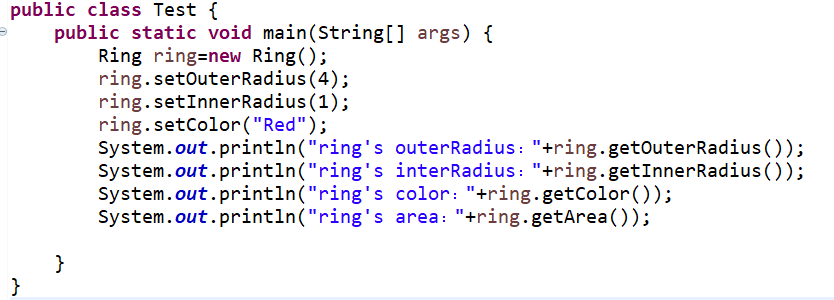
}

public void setColor(String color) {

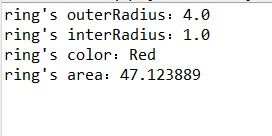
this.color = color;

}

(4) 程序测试过程



(5) 运行结果（运行效果）



2.

1. 设计思路

PAI、ringNumber设置成static 静态变量，同时PAI=3.1415926，ringNumber初始化为0，通过构造函数ringNumber++。

(2)实验步骤

①、设置静态变量

②、编写get、set以及构造函数

③、创建多个对象测验

1. 关键代码

public static double *PAI*=3.1415926;

public static int *ringNumber*=0; //圆对象个数

public Ring() {

*ringNumber*++;

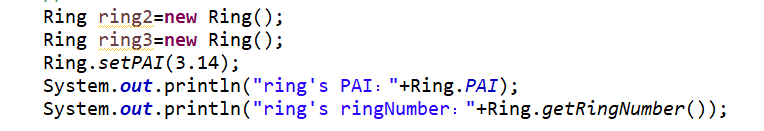
}

public static void setPAI(double tempPAI) {

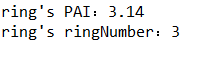
*PAI*=tempPAI;

}

(4) 程序测试过程



(5) 运行结果（运行效果）



3.

1. 设计思路

创建类，定义类成员，编写get、set以及构造函数。

(2)实验步骤

①、创建类，定义类成员，编写方法

②、创建多个类，设置属性

③、测试

(3) 关键代码

public String bookName;

public String authorName;

public String language;

public String type;

public String className;

public String type;

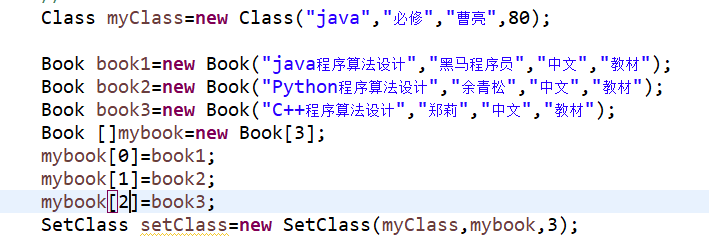
public String teacherName;

public int stuNumber;

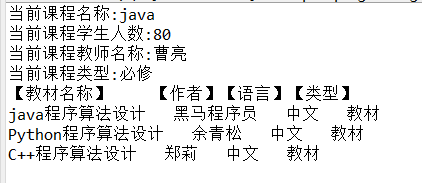
public SetClass(Class setClass,Book []setBoook,int bookNumber) {

}

(4) 程序测试过程



(5) 运行结果（运行效果）



1. **实验总结**

1静态成员通过类名.访问，实例对象通过对象.访问

1. 可以通过类内置方法设置属性
2. 可以重写构造函数
3. **附录（程序代码）**
4. **、（2）**

private float innerRadius; //内半径

private float outerRadius; //外半径

private String color;

public static double *PAI*=3.1415926;

public static int *ringNumber*=0; //圆对象个数

public Ring() {

*ringNumber*++;

}

public Ring(float innerRadius, float outerRadius, String color) {

super();

this.innerRadius = innerRadius;

this.outerRadius = outerRadius;

this.color = color;

*ringNumber*++;

}

public double getArea() {

return *PAI*\*(outerRadius\*outerRadius-innerRadius\*innerRadius);

}

public static void setPAI(double tempPAI) {

*PAI*=tempPAI;

}

public static int getRingNumber() {

return *ringNumber*;

}

public float getInnerRadius() {

return innerRadius;

}

public void setInnerRadius(float innerRadius) {

this.innerRadius = innerRadius;

}

public float getOuterRadius() {

return outerRadius;

}

public void setOuterRadius(float outerRadius) {

this.outerRadius = outerRadius;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

（3）

Class类：

public String className;

public String type;

public String teacherName;

public int stuNumber;

public Class() {

}

public Class(String className) {

this.className = className;

}

public Class(String className, String type) {

this.className = className;

this.type = type;

}

public Class(String className, String type, String teacherName) {

this.className = className;

this.type = type;

this.teacherName = teacherName;

}

public Class(String className, String type, String teacherName, int stuNumber) {

super();

this.className = className;

this.type = type;

this.teacherName = teacherName;

this.stuNumber = stuNumber;

}

public String getClassName() {

return className;

}

public void setClassName(String className) {

this.className = className;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getTeacherName() {

return teacherName;

}

public void setTeacherName(String teacherName) {

this.teacherName = teacherName;

}

public int getStuNumber() {

return stuNumber;

}

public void setStuNumber(int stuNumber) {

this.stuNumber = stuNumber;

}

Book类：

public String bookName;

public String authorName;

public String language;

public String type;

public Book() {

}

public Book(String bookName) {

this.bookName = bookName;

}

public Book(String bookName, String authorName) {

this.bookName = bookName;

this.authorName = authorName;

}

public Book(String bookName, String authorName, String language) {

this.bookName = bookName;

this.authorName = authorName;

this.language = language;

}

public Book(String bookName, String authorName, String language, String type) {

this.bookName = bookName;

this.authorName = authorName;

this.language = language;

this.type = type;

}

public String getBookName() {

return bookName;

}

public void setBookName(String bookName) {

this.bookName = bookName;

}

public String getAuthorName() {

return authorName;

}

public void setAuthorName(String authorName) {

this.authorName = authorName;

}

public String getLanguage() {

return language;

}

public void setLanguage(String language) {

this.language = language;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

SetClass类：

public SetClass(Class setClass,Book []setBoook,int bookNumber) {

System.*out*.println("当前课程名称:"+setClass.getClassName());

System.*out*.println("当前课程学生人数:"+setClass.getStuNumber());

System.*out*.println("当前课程教师名称:"+setClass.getTeacherName());

System.*out*.println("当前课程类型:"+setClass.getType());

System.*out*.println("【教材名称】 【作者】【语言】【类型】");

for(int i=0;i<bookNumber;i++)

{

System.*out*.print(setBoook[i].getBookName()+" ");

System.*out*.print(setBoook[i].getAuthorName()+" ");

System.*out*.print(setBoook[i].getLanguage()+" ");

System.*out*.println(setBoook[i].getType()+" ");

}

}

Test类：

public static void main(String[] args) {

Class myClass=new Class("java","必修","曹亮",80);

Book book1=new Book("java程序算法设计","黑马程序员","中文","教材");

Book book2=new Book("Python程序算法设计","余青松","中文","教材");

Book book3=new Book("C++程序算法设计","郑莉","中文","教材");

Book []mybook=new Book[3];

mybook[0]=book1;

mybook[1]=book2;

mybook[2]=book3;

SetClass setClass=new SetClass(myClass,mybook,3);

}

# 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**

JDK，eclipse，idea

1. **设计思路与具体实现**

1.

(1)设计思路

创建学生、老师类，添加属性和set、get方法，创建控制类，传入学生、老师两个ArrayList对象，对两个ArrayList进行操作

(2)实验步骤

①、创建学生、老师类，完成类的编写

②、创建控制类，先查询，后添加、删除、修改方法实现

③、创建Test类，传入对象一一测验功能

1. 关键代码

查询：

int hasBeSearchedStuNumber=0;

int index=0;

ArrayList<Integer> array=new ArrayList<Integer>();

for(Student x :stu){

//如果 名字符合

if(x.getStuName().equals(id) ){

if(kaiguan){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Student["+index+"]");

}

array.add(index);

hasBeSearchedStuNumber++;

}

//编号 符合

if(x.getStuNumber().equals(id)) {

if (kaiguan) {

System.out.println((hasBeSearchedStuNumber + 1) + "." + x + "index: Student[" + index + "]");

}

array.add(index);

return array;

}

index++;

}

if(hasBeSearchedStuNumber==0){

System.out.println("查无此人");

return null;

}

return array;

添加：

public void addStudent(Student student){

this.stu.add(student);

}

删除：

public void deleteStudent(int index){

if(index<=stu.size()-1 && index>=0){

deleteStudent(stu.get(index));

}else{

try {

throw (Throwable) IndexOutOfBoundsException;

} catch (Throwable throwable) {

throwable.printStackTrace();

}

}

}

修改：

public Student setStudent(String id){

ArrayList array=searchStudent(id,false);

if(array!=null){

return stu.get((int)array.get(0));

}

else{

return null;

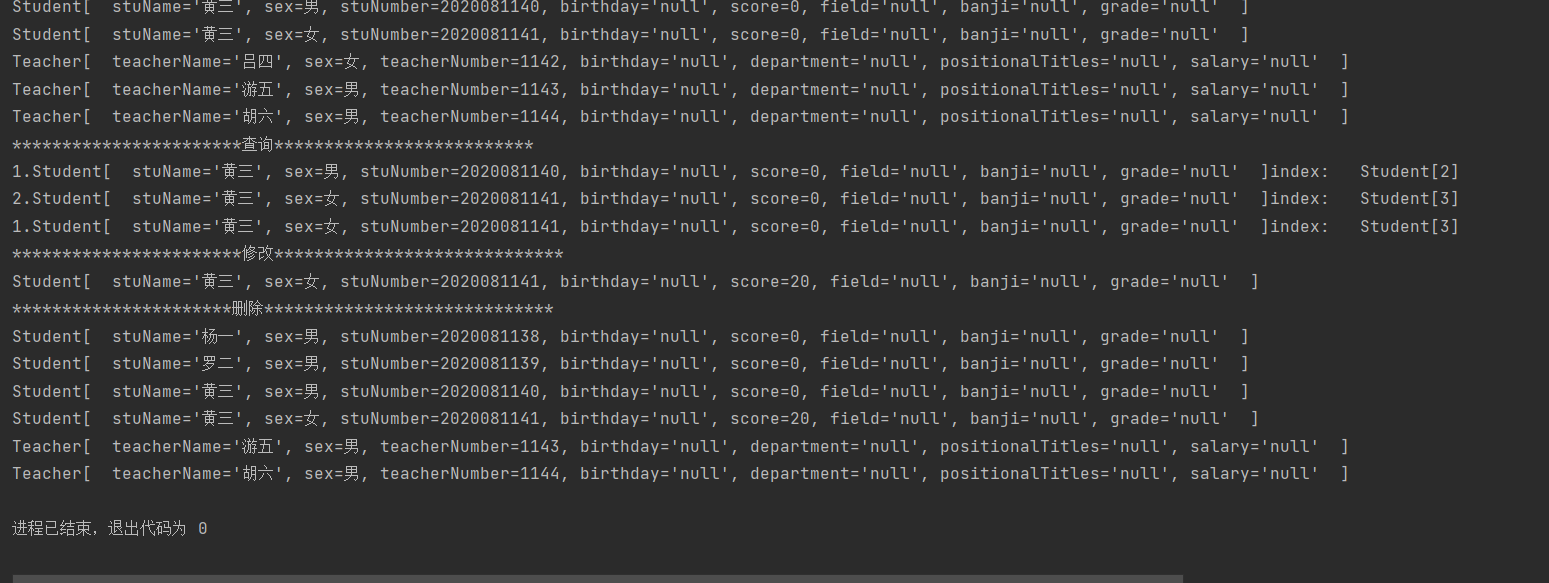
}

}

(4) 程序测试过程



(5) 运行结果（运行效果）



2.

1. 设计思路

分别设计三个类，两个子类继承父类，主函数中创建一个父类数组，将子类对象放进数组访问。

(2)实验步骤

①、创建一个父类，两个子类，编写构造函数

②、创建一个父类数组，并将子类对象放入数组，访问数组成员属性

③、统计输出结果

(3) 关键代码

Student[] stu=**new** Student[4];

Undergraduate unStu1=**new** Undergraduate("吕三","2020081141",85);

Undergraduate unStu2=**new** Undergraduate("游四","2020081142",26);

Postgraduate posStu1=**new** Postgraduate("胡五","2020081143",62);

Postgraduate posStu2=**new** Postgraduate("罗六","2020081144",81);

stu[0]=unStu1;

stu[1]=unStu2;

stu[2]=posStu1;

stu[3]=posStu2;

if(score>90&&score<=100) this.grade="优秀";

if(score>80&&score<=90) this.grade="良好";

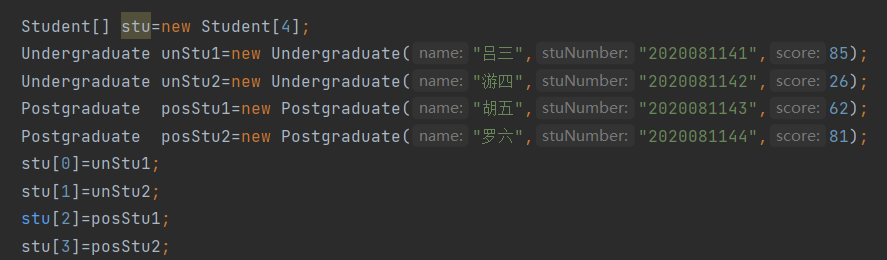
if(score>70&&score<=80) this.grade="一般";

if(score>60&&score<=70) this.grade="及格";

if(score<=60) this.grade="不及格";

if(score==0) this.grade="无成绩等级";

(4) 程序测试过程



(5) 运行结果（运行效果）



1. **实验总结**
2. **附录（程序代码）**
3. 控制类:

public class ManagementSystem {

ArrayList<Student>stu;

ArrayList<Teacher>teachers;

private Object IndexOutOfBoundsException;

public ManagementSystem() {

}

public ManagementSystem(ArrayList<Student> stu, ArrayList<Teacher> teachers) {

this.stu = stu;

this.teachers = teachers;

}

/\*添加\*/

public void addStudent(Student student){

this.stu.add(student);

}

public void addTeacher(Teacher teacher){

this.teachers.add(teacher);

}

/\*删除\*/

public void deleteStudent(Student student){

this.stu.remove(student);

}

public void deleteTeacher(Teacher teacher){

this.teachers.remove(teacher);

}

public void deleteStudent(int index){

if(index<=stu.size()-1 && index>=0){

deleteStudent(stu.get(index));

}else{

try {

throw (Throwable) IndexOutOfBoundsException;

} catch (Throwable throwable) {

throwable.printStackTrace();

}

}

}

public void deleteTeacher(int index){

if(index<=stu.size()-1 && index>=0){

deleteTeacher(teachers.get(index));

}else{

try {

throw (Throwable) IndexOutOfBoundsException;

} catch (Throwable throwable) {

throwable.printStackTrace();

}

}

}

public void deleteStudent(int startIndex,int endIndex){

if(endIndex<=stu.size()-1 && endIndex>=startIndex){

for(int i=0;i<(endIndex-startIndex+1);i++){

stu.remove(startIndex);

}

}else{

try {

throw (Throwable) IndexOutOfBoundsException;

} catch (Throwable throwable) {

throwable.printStackTrace();

}

}

}

public void deleteTeacher(int startIndex,int endIndex){

if(endIndex<=stu.size()-1 && endIndex>=startIndex){

for(int i=0;i<(endIndex-startIndex+1);i++){

teachers.remove(startIndex);

}

}else{

try {

throw (Throwable) IndexOutOfBoundsException;

} catch (Throwable throwable) {

throwable.printStackTrace();

}

}

}

/\*修改\*/

public Student setStudent(int index){

return stu.get(index);

}

public Teacher setTeachers(int index){

return teachers.get(index);

}

public Student setStudent(String id){

ArrayList array=searchStudent(id,false);

if(array!=null){

return stu.get((int)array.get(0));

}

else{

return null;

}

}

public Teacher setTeachers(String id){

ArrayList array=searchTeacher(id,false);

if(array!=null){

return teachers.get((int)array.get(0));

}

else{

return null;

}

}

/\*查询\*/

public ArrayList searchStudent(String id){

int hasBeSearchedStuNumber=0;

int index=0;

ArrayList<Integer> array=new ArrayList<Integer>();

for(Student x :stu){

//如果 名字符合

if(x.getStuName().equals(id)){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Student["+index+"]");

array.add(index);

hasBeSearchedStuNumber++;

}

//编号 符合

if(x.getStuNumber().equals(id)){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Student["+index+"]");

array.add(index);

return array;

}

index++;

}

if(hasBeSearchedStuNumber==0){

System.out.println("查无此人");

return null;

}

return array;

}

public ArrayList searchStudent(String id,Boolean kaiguan){

int hasBeSearchedStuNumber=0;

int index=0;

ArrayList<Integer> array=new ArrayList<Integer>();

for(Student x :stu){

//如果 名字符合

if(x.getStuName().equals(id) ){

if(kaiguan){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Student["+index+"]");

}

array.add(index);

hasBeSearchedStuNumber++;

}

//编号 符合

if(x.getStuNumber().equals(id)) {

if (kaiguan) {

System.out.println((hasBeSearchedStuNumber + 1) + "." + x + "index: Student[" + index + "]");

}

array.add(index);

return array;

}

index++;

}

if(hasBeSearchedStuNumber==0){

System.out.println("查无此人");

return null;

}

return array;

}

public ArrayList searchTeacher(String id){

int hasBeSearchedStuNumber=0;

int index=0;

ArrayList<Integer> array=new ArrayList<Integer>();

for(Teacher x : teachers){

//如果 名字或者编号 符合

if(x.getTeacherName().equals(id)){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Teacher["+index+"]");

array.add(index);

hasBeSearchedStuNumber++;

}

if(x.getTeacherNumber().equals(id)){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Teacher["+index+"]");

array.add(index);

return array;

}

index++;

}

if(hasBeSearchedStuNumber==0){

System.out.println("查无此人");

return null;

}

return array;

}

public ArrayList searchTeacher(String id,Boolean kaiguan){

int hasBeSearchedStuNumber=0;

int index=0;

ArrayList<Integer> array=new ArrayList<Integer>();

for(Teacher x : teachers){

//如果 名字或者编号 符合

if( x.getTeacherNumber().equals(id)){

if(kaiguan){

System.out.println((hasBeSearchedStuNumber+1)+"."+x+"index: Teacher["+index+"]");

}

array.add(index);

hasBeSearchedStuNumber++;

}

if(x.getTeacherNumber().equals(id)) {

if (kaiguan) {

System.out.println((hasBeSearchedStuNumber + 1) + "." + x + "index: Teacher[" + index + "]");

}

array.add(index);

return array;

}

index++;

}

if(hasBeSearchedStuNumber==0){

System.out.println("查无此人");

return null;

}

return array;

}

public void showAll(){

for(Student x:stu){

System.out.println(x);

}

for(Teacher x:teachers){

System.out.println(x);

}

}

public void showAllStudents(){

for(Student x:stu){

System.out.println(x);

}

}

public void showAllTeachers(){

for(Teacher x:teachers){

System.out.println(x);

}

}

}

学生类:

public void setStuName(String stuName) {

this.stuName = stuName;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public String getStuNumber() {

return stuNumber;

}

public void setStuNumber(String stuNumber) {

this.stuNumber = stuNumber;

}

public String getBirthday() {

return birthday;

}

public void setBirthday(String birthday) {

this.birthday = birthday;

}

public int getScore() {

return score;

}

public void setScore(int score) {

this.score = score;

}

public String getField() {

return field;

}

public void setField(String field) {

this.field = field;

}

public String getBanji() {

return banji;

}

public void setBanji(String banji) {

this.banji = banji;

}

public String getGrade() {

return grade;

}

public void setGrade(String grade) {

this.grade = grade;

}

@Override

public String toString() {

return "Student[ " +

"stuName='" + stuName + '\'' +

", sex=" + sex +

", stuNumber=" + stuNumber +

", birthday='" + birthday + '\'' +

", score=" + score +

", field='" + field + '\'' +

", banji='" + banji + '\'' +

", grade='" + grade + '\'' +

" ]";

}

}

老师类:

public class Teacher {

public String teacherName;

public String sex;

public String teacherNumber;//编号

public String birthday;//出生日期

public String department;//部门

public String positionalTitles;//职称

public String salary;//薪水

public Teacher() {

}

public Teacher(String teacherName) {

this.teacherName = teacherName;

}

public Teacher(String teacherName, String sex) {

this.teacherName = teacherName;

this.sex = sex;

}

public Teacher(String teacherName, String sex, String teacherNumber) {

this.teacherName = teacherName;

this.sex = sex;

this.teacherNumber = teacherNumber;

}

public String getTeacherName() {

return teacherName;

}

public void setTeacherName(String teacherName) {

this.teacherName = teacherName;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public String getTeacherNumber() {

return teacherNumber;

}

public void setTeacherNumber(String teacherNumber) {

this.teacherNumber = teacherNumber;

}

public String getBirthday() {

return birthday;

}

public void setBirthday(String birthday) {

this.birthday = birthday;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public String getPositionalTitles() {

return positionalTitles;

}

public void setPositionalTitles(String positionalTitles) {

this.positionalTitles = positionalTitles;

}

public String getSalary() {

return salary;

}

public void setSalary(String salary) {

this.salary = salary;

}

@Override

public String toString() {

return "Teacher[ " +

"teacherName='" + teacherName + '\'' +

", sex=" + sex +

", teacherNumber=" + teacherNumber +

", birthday='" + birthday + '\'' +

", department='" + department + '\'' +

", positionalTitles='" + positionalTitles + '\'' +

", salary='" + salary + '\'' +

" ]";

}

}

（2）

Student类:

public String stuName;

public char sex;

public String stuNumber;//编号

public String grade;

public int score;

Undergraduate 类：

public Undergraduate() {

}

public Undergraduate(String name){

this.stuName=name;

}

public Undergraduate(String name,String stuNumber){

this.stuName=name;

this.stuNumber=stuNumber;

}

public Undergraduate(String name,String stuNumber,int score){

this.stuName=name;

this.stuNumber=stuNumber;

this.score=score;

setGrade(this.score);

}

private void setGrade(int score){

if(score>80&&score<=100) this.grade="优秀";

if(score>70&&score<=80) this.grade="良好";

if(score>60&&score<=70) this.grade="一般";

if(score>50&&score<=60) this.grade="及格";

if(score<=50) this.grade="不及格";

if(score==0) this.grade="无成绩等级";

}

Postgraduate,类：

public Postgraduate() {

}

public Postgraduate(String name){

this.stuName=name;

}

public Postgraduate(String name,String stuNumber){

this.stuName=name;

this.stuNumber=stuNumber;

}

public Postgraduate(String name,String stuNumber,int score){

this.stuName=name;

this.stuNumber=stuNumber;

this.score=score;

setGrade(this.score);

}

private void setGrade(int score){

if(score>90&&score<=100) this.grade="优秀";

if(score>80&&score<=90) this.grade="良好";

if(score>70&&score<=80) this.grade="一般";

if(score>60&&score<=70) this.grade="及格";

if(score<=60) this.grade="不及格";

if(score==0) this.grade="无成绩等级";

}

# 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**

JDK，eclipse

1. **设计思路与具体实现**

1.

(1) 设计思路

编写带有int 除数和 int 被除数 两个形参的方法，try中执行输出二者相除结果，分别catch异常：ArithmeticException、NumberFormatException、Exception.

(2)实验步骤

①、编写带有除数和被除数形参的方法

②、方法中使用try catch三种异常，并打印异常

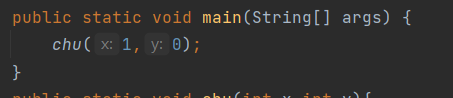
③、在main函数中调用方法测验。

(3) 关键代码

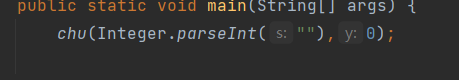
try{  
 System.out.println(x/y);  
  
}catch(ArithmeticException e){  
 System.out.println("算数异常:除数是0");  
 e.printStackTrace();  
}  
catch(NumberFormatException e){  
 System.out.println("数字格式异常");  
 e.printStackTrace();  
}  
catch (Exception e){  
 System.out.println("通用型异常");  
 e.printStackTrace();  
}

(4) 程序测试过程

算数异常：除数为0

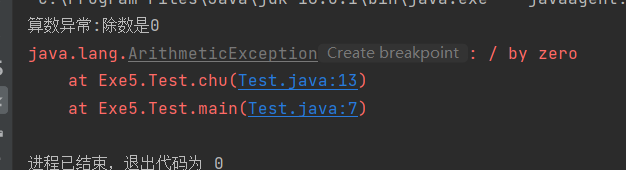


数字格式异常



(5) 运行结果（运行效果）





2.

(1) 设计思路

将 1. 的方法形参改为 double型，主函数中利用控制变量法多次运行方法

(2)实验步骤

①、修改方法形参

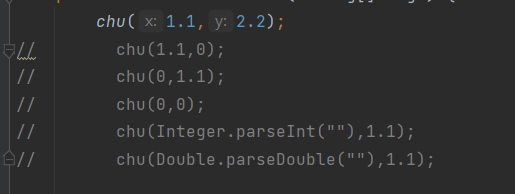
②、主函数中多次运行不同实参的方法

③、运行测试，观察

(3) 关键代码

try{  
 System.out.println(x/y);  
  
}catch(ArithmeticException e){  
 System.out.println("双精度算数异常:除数是0");  
 e.printStackTrace();  
}  
catch(NumberFormatException e){  
 System.out.println("双精度数字格式异常");  
 e.printStackTrace();  
}  
catch (Exception e){  
 System.out.println("双精度通用型异常");  
 e.printStackTrace();  
}

(4) 程序测试过程

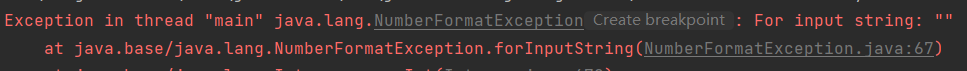


(5) 运行结果（运行效果）









3.

(1) 设计思路

编写异常类继承Exception或者其子类，创建带String的构造方法，整除方法try中判断如果除数等于0，则抛出自定义异常，并catch该异常。

(2)实验步骤

①、编写异常类继承Exception或者其子类

②、改写整除方法

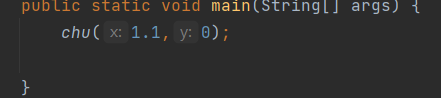
③、在main函数中测验。

(3) 关键代码

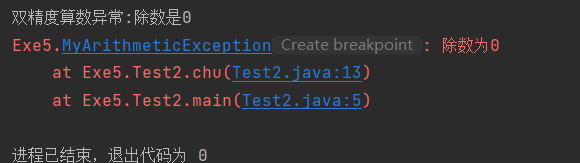
public class MyArithmeticException extends Exception{

try{  
 if(y==0){  
 throw new MyArithmeticException("除数为0");  
 }  
 System.out.println(x/y);

(4) 程序测试过程



(5) 运行结果（运行效果）



1. **实验总结**
2. catch可以多次使用，捕获多个异常。

2.自定义异常类必须继承Exception或者其子类，且必须创建构造函数。

3.thow手动抛出异常后，要捕获异常。

1. **附录（程序代码）**

**(1)**

public class Test1 {  
 public static void main(String[] args) {  
 chu(1,0);  
  
 }  
 public static void chu(int x,int y){  
  
  
 try{  
 System.out.println(x/y);  
  
 }catch(ArithmeticException e){  
 System.out.println("算数异常:除数是0");  
 e.printStackTrace();  
 }  
 catch(NumberFormatException e){  
 System.out.println("数字格式异常");  
 e.printStackTrace();  
 }  
 catch (Exception e){  
 System.out.println("通用型异常");  
 e.printStackTrace();  
 }  
  
  
 }  
}

(2)

public class Test2 {  
 public static void main(String[] args) {  
 chu(1.1,0);  
  
 }  
 public static void chu(double x,double y){  
  
  
 try{  
 System.out.println(x/y);  
 }catch(ArithmeticException e){  
 System.out.println("双精度算数异常:除数是0");  
 e.printStackTrace();  
 }  
 catch(NumberFormatException e){  
 System.out.println("双精度数字格式异常");  
 e.printStackTrace();  
 }  
 catch (Exception e){  
 System.out.println("双精度通用型异常");  
 e.printStackTrace();  
 }  
  
  
 }  
}

(3)

public class MyArithmeticException extends Exception{  
  
 public MyArithmeticException() {  
 }  
  
 public MyArithmeticException(String message) {  
 super(message);  
 }  
  
 public MyArithmeticException(String message, Throwable cause) {  
 super(message, cause);  
 }  
  
 public MyArithmeticException(Throwable cause) {  
 super(cause);  
 }  
  
 public MyArithmeticException(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace) {  
 super(message, cause, enableSuppression, writableStackTrace);  
 }  
}

# 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**

JDK，eclipse

1. **设计思路与具体实现**

1.

(1) 设计思路

创建一个只能容纳String对象名为names的ArrayList集合，添加5个字符串内容，用增加for遍历集合输出，用集合.remove删除元素，再遍历输出。

(2)实验步骤

①、创建集合

②、添加字符串内容，遍历输出

③、删除元素，遍历输出

(3) 关键代码

ArrayList<String> names=new ArrayList<>();

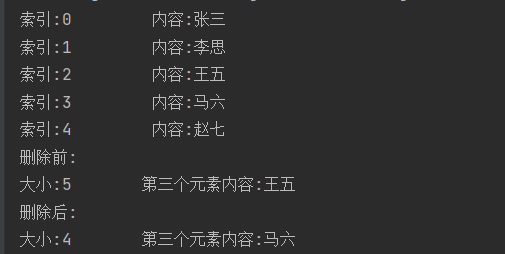
for(String str:names){  
 System.out.println("索引:"+names.indexOf(str)+" 内 容:"+str);  
}

names.remove(2);

(4) 程序测试过程



(5) 运行结果（运行效果）



2.

(1) 设计思路

创建hashMap对象，读取命令行输入字符串，并遍历字符串，字符作为键，索引作为值，使用map对象.put（）添加，若有键重复出现，则先取键对应的值，将此值拼接索引，再进行put（）方法覆盖原值。

(2)实验步骤

①、创建hashMap，指定泛型为String,String

②、读取命令行字符串，并遍历字符串，取出键值对

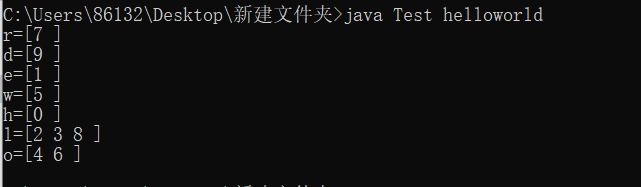
③、将键值对放入hashMap对象中,遍历输出.

(3) 关键代码

for(int i=0;i<str.length();i++){  
 String temp="";  
 x=String.valueOf(str.charAt(i));  
// 跳过空格  
 if(" ".equals(x)){  
 continue;  
 }  
 temp=tm.get(x);  
// 如果存在键,则代表已有对应值,取值加字符串,再放回hashMap自动覆盖值  
 if(temp==null){  
 temp="";  
 }  
 temp+=i+" ";  
 tm.put(x,temp);  
 }

(4) 程序测试过程

(5) 运行结果（运行效果）



1. **实验总结**
2. ArrayList需要指定一个泛型，hashMap需要指定键值对泛型。
3. 对于HashMap中存在的键，新put的键值对会覆盖原有的值。
4. hashMap使用containKey（）时需要注意,因为Key始终存在。
5. **附录（程序代码）**

**1.**

public class Test1 {  
 public static void main(String[] args) {  
 ArrayList<String> names=new ArrayList<>();  
 names.add("张三");  
 names.add("李思");  
 names.add("王五");  
 names.add("马六");  
  
 names.add("赵七");  
 for(String str:names){  
 System.out.println("索引:"+names.indexOf(str)+" 内容:"+str);  
 }  
 System.out.println("删除前:");  
 System.out.println("大小:" +names.size()+" 第三个元素内容:"+names.get(2));  
 names.remove(2);  
 System.out.println("删除后:");  
 System.out.println("大小:" +names.size()+" 第三个元素内容:"+names.get(2));  
  
  
 }  
}

2.public class Test2 {  
 public static void main(String[] args) {  
 String str=args[0];  
 String x;  
 HashMap<String, String> tm =new HashMap<>();  
 for(int i=0;i<str.length();i++){  
 String temp="";  
 x=String.valueOf(str.charAt(i));  
// 跳过空格  
 if(" ".equals(x)){  
 continue;  
 }  
 temp=tm.get(x);  
//如果存在键,则代表已有对应值,取值加字符串,再放回hashMap自动覆盖值  
 if(temp==null){  
 temp="";  
 }  
 temp+=i+" ";  
 tm.put(x,temp);  
 }  
 for(String i:tm.keySet()){  
 System.out.println(i+"=["+tm.get(i)+"]");  
 }  
  
  
 }  
}

# 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**

JDK，eclipse

1. **设计思路与具体实现**

1.

(1) 设计思路

读取命令行字符串指定源文件和目标文件地址，创建字节码输入输出流，循环读取read（），并同时写出write（）。

(2)实验步骤

①、创建源文件，写入内容

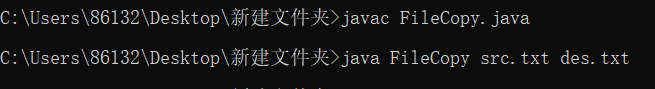
②、创建输入输出流，读取并写出

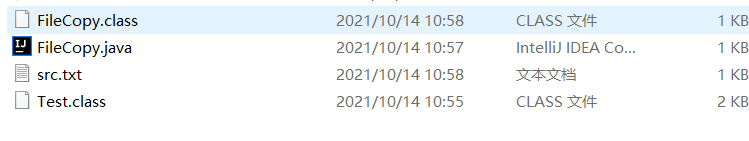
③、查看是否有目标文件生成，检查内容是否一致。

(3) 关键代码

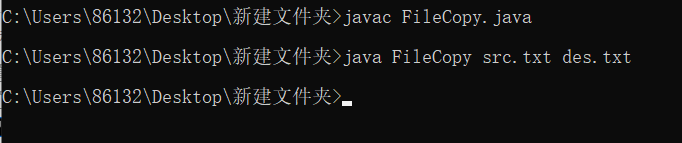
while((len=bis.read())!=-1){  
 bos.write(len);  
}

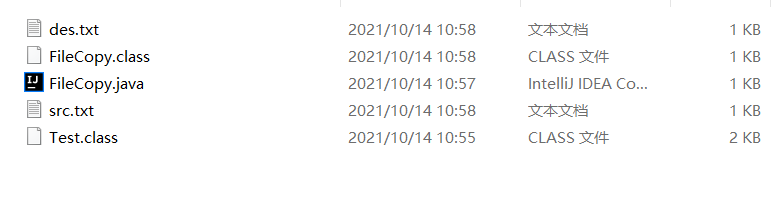
(4) 程序测试过程





(5) 运行结果（运行效果）





2.

(1) 设计思路

创建两个字节输出流，一个字节输出流，并指定输出流构造函数为（目标文件地址，append=true）允许追加，读取的同时写出。

(2)实验步骤

①、创建两个源文件，输入内容

②、创建字节输入输出流，读取源文件的同时写出到目标文件

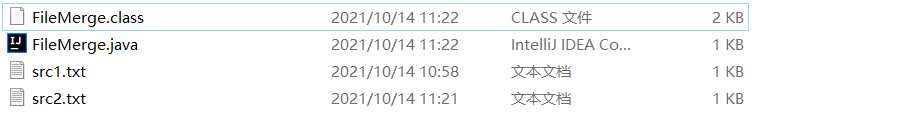
③、命令行运行java文件，并输入文件地址

(3) 关键代码

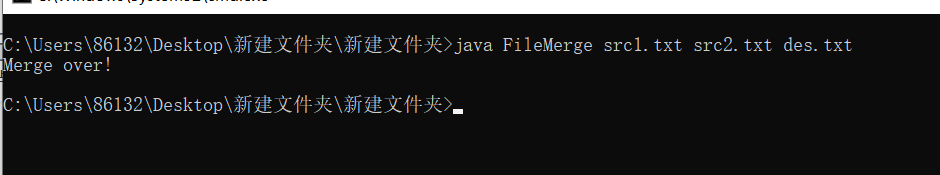
int len;  
while((len=bis1.read())!=-1){  
 bos.write(len);  
}  
while((len=bis2.read())!=-1){  
 bos.write(len);  
}

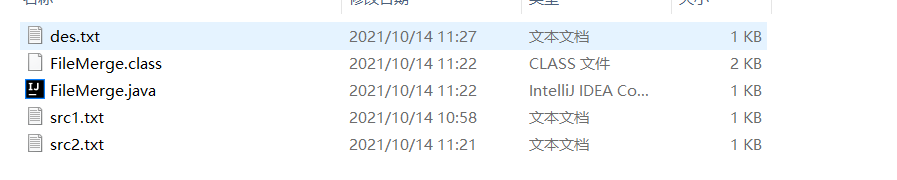
(4) 程序测试过程





(5) 运行结果（运行效果）





3.

(1) 设计思路

读取命令行字符串指定源文件和目标文件地址，创建字节码输入输出流，输出流指定运行追加，循环读取read（），并同时写出write（）。

(2)实验步骤

①、创建源文件和目标文件，写入内容

②、创建输入输出流，读取并写出

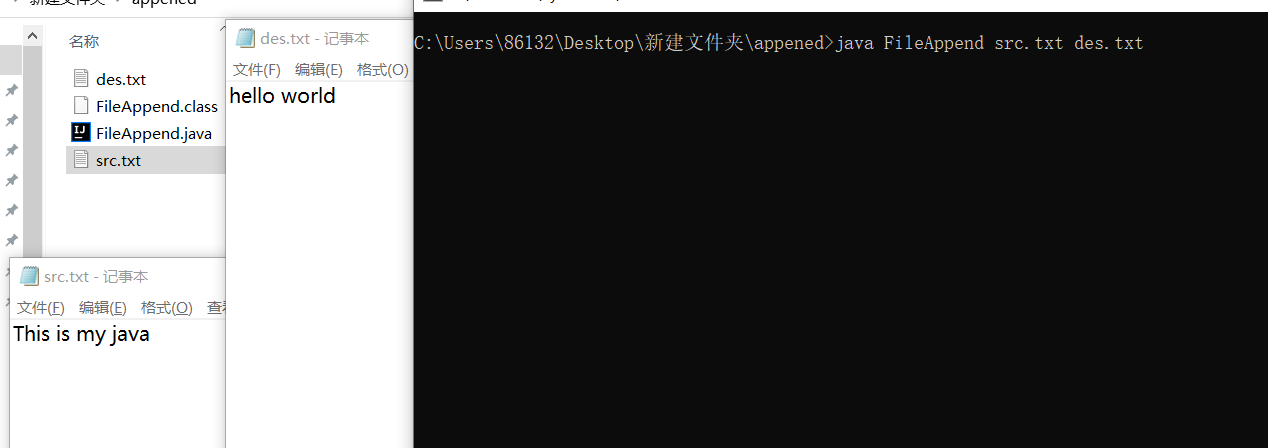
③、查看是否有目标文件是否有追加内容，检查内容是否一致。

(3) 关键代码

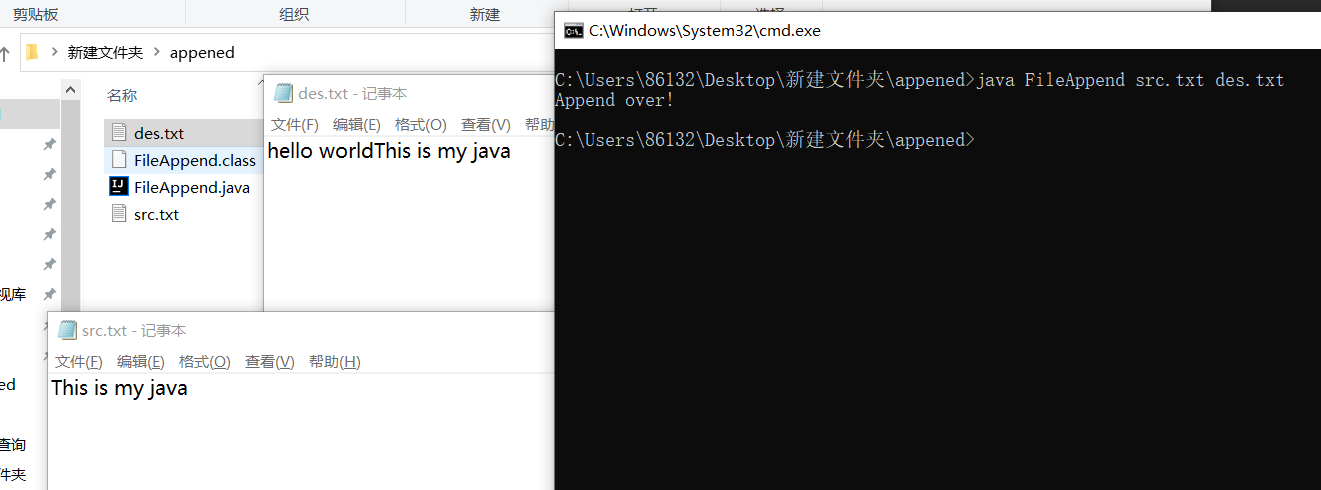
BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStream(des,true));

while((len=bis.read())!=-1){  
 bos.write(len);  
}

(4) 程序测试过程



(5) 运行结果（运行效果）



1. **实验总结**

1.字节输入流读取一个字节，并且返回一个十进制整型

2.字节输出流可以输出一个十进制整型，并且其构造函数形参除了文件地址，还可以选择追加输出true。

3.使用字节缓冲流可以提高输入输出效率。

1. **附录（程序代码）**

**1.**

public class FileCopy {  
 public static void main(String[] args) throws Exception{  
 String src=args[0];  
 String des=args[1];  
  
 BufferedInputStream bis =new BufferedInputStream(new FileInputStream(src));  
 BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStream(des));  
 int len;  
 while((len=bis.read())!=-1){  
 bos.write(len);  
 }  
 bis.close();  
 bos.close();  
 }  
}

2.

public class FileMerge {  
 public static void main(String[] args) throws Exception{  
 String src1=args[0];  
 String src2=args[1];  
 String des=args[2];  
  
 BufferedInputStream bis1 =new BufferedInputStream(new FileInputStream(src1));  
 BufferedInputStream bis2=new BufferedInputStream(new FileInputStream(src2));  
 BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStream(des,true));  
 int len;  
 while((len=bis1.read())!=-1){  
 bos.write(len);  
 }  
 while((len=bis2.read())!=-1){  
 bos.write(len);  
 }  
 System.out.println("Merge over！");  
 bos.flush();  
 bos.close();  
 bis1.close();  
 bis2.close();  
  
 }  
}

3.

public class FileAppend {  
 public static void main(String[] args) throws Exception{  
 String src=args[0];  
 String des=args[1];  
  
 BufferedInputStream bis =new BufferedInputStream(new FileInputStream(src));  
 BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStream(des,true));  
 int len;  
 while((len=bis.read())!=-1){  
 bos.write(len);  
 }  
 System.out.println("Append over!");  
 bis.close();  
 bos.close();  
 }  
}

# 实验8：多线程应用练习

# 

1. **实验目的**

1. 掌握线程的创建方法。

2. 熟悉线程的启动等方法的使用

3. 了解线程同步

4. 掌握synchronized关键字的使用

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**

JDK，eclipse

1. **设计思路与具体实现**

1.

（1）设计思路

创建一个类实现Runable接口，重写.run（）方法，通过Thread.currentThread().getName()获取当前线程名字输出

(2)实验步骤

①、创建类sy6\_1

②、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

(3) 关键代码

**public** **void** run() {

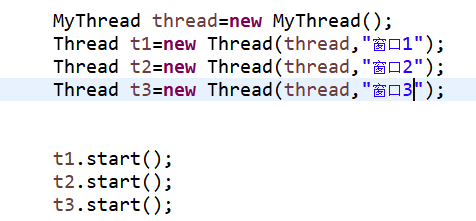
**for**(**int** i=0;i<5;i++) {

System.***out***.println("the thread is :"+Thread.*currentThread*().getName());

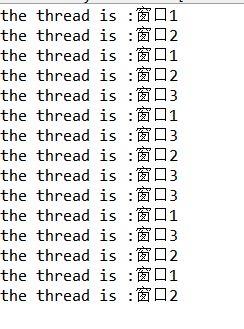
}

}

(4) 程序测试过程



(5) 运行结果（运行效果）



2.

（1）设计思路

Storage类中定义一个int x，同时添加get、set数方法；Counter和Printer类继承 Thread类，其构造方法中形参为Storage对象，并在Counter和Printer的run方法中调用get、set方法设置Storage的x。

(2)实验步骤

①、创建三个类，重写Counter和Printer类构造方法

②、Counter和Printer类继承Thread类，并且重写.run()方法

③、主函数中创建三个对象，并调用线程对象.start方法。

(3) 关键代码

Counter类：

Storage storage;

public void run() {

for(int i=1;i<=10;i++) {

storage.setValue(i);

System.*out*.println("Counter类线程正在存储："+i);

}

Printer类：

Storage storage;

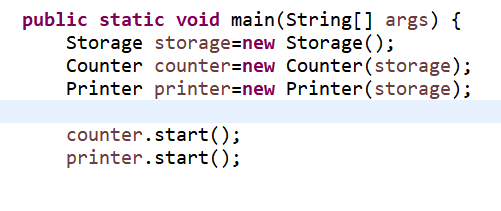
public void run() {

for(int i=1;i<=10;i++) {

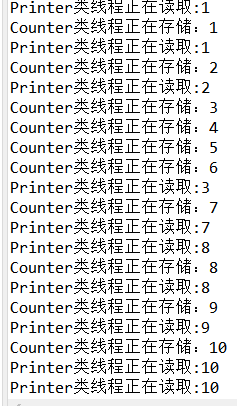
System.*out*.println("Printer类线程正在读取:"+storage.getValue());

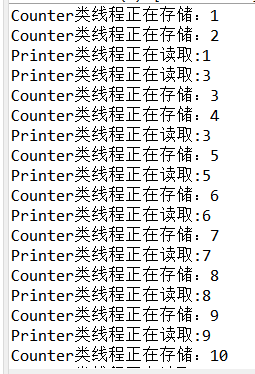
}

(4) 程序测试过程



(5) 运行结果（运行效果）





3.（1）设计思路

Storage类中定义一个int x，同时添加get、set方法,同时方法上加上synchronized关键字；Counter和Printer类继承 Thread类，其构造方法中形参为Storage对象，并在Counter和Printer的run方法中调用get、set方法。

(2)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、Counter1对象和 Printer对象，创建Counter线程和Printer线程并启动

③、在定义Storage类中的setValue(int i) 和getValue ()方法时使用synchronized关键字，将其定义为同步方法，编译运行

(3) 关键代码

public synchronized int getValue() {

if(!flag) {

try {

this.wait();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

System.*out*.println("Printer类线程正在读取:"+x);

flag=false;

this.notify();

return x;

}

public synchronized void setValue(int x) {

if(flag) {

try {

this.wait();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

this.x=x;

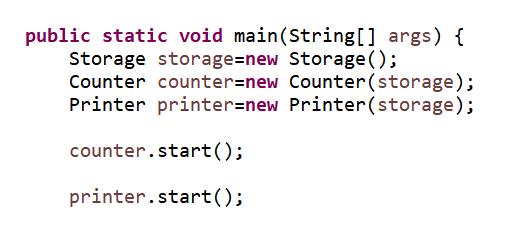
System.*out*.println("Counter类线程正在存储："+x);

flag=true;

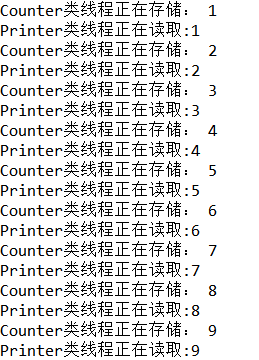
this.notify();

}

(4) 程序测试过程



(5) 运行结果（运行效果）



1. **实验总结**
2. **附录（程序代码）**

1.

public class MyThreadTest {

public static void main(String[] args) {

MyThread thread=new MyThread();

Thread t1=new Thread(thread,"窗口1");

Thread t2=new Thread(thread,"窗口2");

Thread t3=new Thread(thread,"窗口3");

t1.start();

t2.start();

t3.start();

}

}

class MyThread implements Runnable{

public void run() {

for(int i=0;i<5;i++) {

System.*out*.println("the thread is :"+Thread.*currentThread*().getName());

}

}

}

2.

public class Test {

public static void main(String[] args) {

Storage storage=new Storage();

Counter counter=new Counter(storage);

Printer printer=new Printer(storage);

counter.start();

printer.start();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Storage{

public int x;

public int getValue() {

return x;

}

public void setValue(int x) {

this.x=x;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Counter extends Thread{

Storage storage;

public Counter(Storage storage) {

super();

this.storage = storage;

}

@Override

public void run() {

for(int i=1;i<=10;i++) {

storage.setValue(i);

System.*out*.println("Counter类线程正在存储："+i);

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Printer extends Thread{

Storage storage;

public Printer(Storage storage) {

super();

this.storage = storage;

}

@Override

public void run() {

for(int i=1;i<=10;i++) {

System.*out*.println("Printer类线程正在读取:"+storage.getValue());

}

}

}

3.public class ExeTest {

public static void main(String[] args) {

Storage storage=new Storage();

Counter counter=new Counter(storage);

Printer printer=new Printer(storage);

counter.start();

printer.start();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Storage{

public int x;

private boolean flag=false;//true 有数字可以取，false 没有数可以取

public synchronized int getValue() {

if(!flag) {

try {

this.wait();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

System.*out*.println("Printer类线程正在读取:"+x);

flag=false;

this.notify();

return x;

}

public synchronized void setValue(int x) {

if(flag) {

try {

this.wait();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

this.x=x;

System.*out*.println("Counter类线程正在存储："+x);

flag=true;

this.notify();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Counter extends Thread{

Storage storage;

public Counter(Storage storage) {

super();

this.storage = storage;

}

@Override

public void run() {

for(int i=1;i<=10;i++) {

storage.setValue(i);

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Printer extends Thread{

Storage storage;

public Printer(Storage storage) {

super();

this.storage = storage;

}

@Override

public void run() {

for(int i=1;i<=10;i++) {

storage.getValue();

}

}

}

# 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**

JDK，eclipse,，javaFx

1. **设计思路与具体实现**

1.

1. 设计思路

创建服务端，接受客户端数据，完成计算后返回给客户端；创建客户端， 首先连接服务器成功后，输入数据发送，接收数据并显示。

(2)实验步骤

①、建立服务端

②、建立客户端

③、建立测试类

(3) 关键代码

ServerSocket serverSocket=new ServerSocket(10086);

Socket socket=serverSocket.accept();

String receive=bufferedReader.readLine();

bufferedWriter.write(String.valueOf(area));

public static double triangleArea(double ...ds) {

double p=(ds[0]+ds[1]+ds[2])/2;

return Math.sqrt(p\*(p-ds[0])\*(p-ds[1])\*(p-ds[2]));

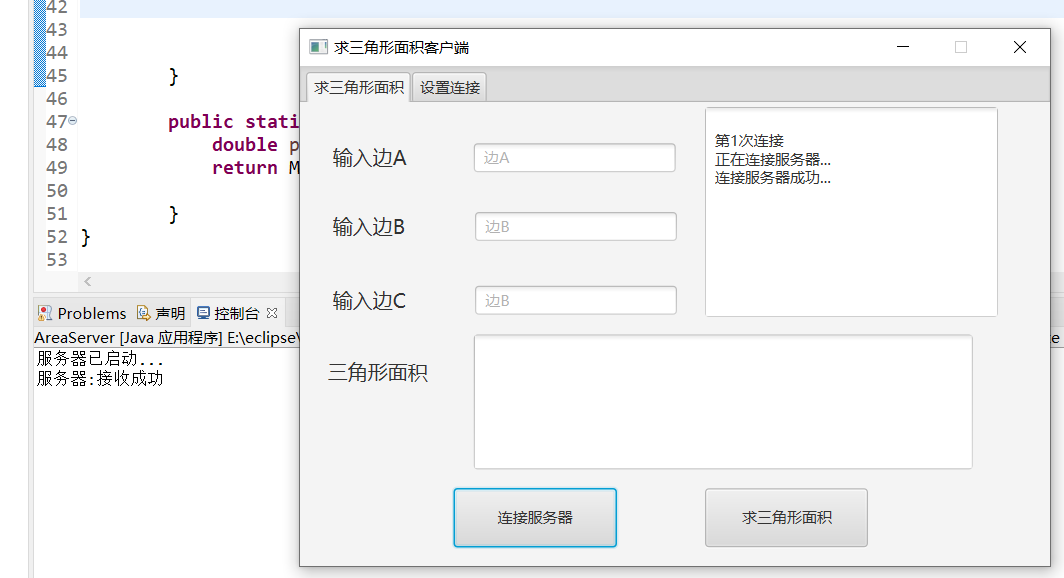
}

socket=new Socket(IP,PORT);

bw.write(text1.getText()+","+text2.getText()+","+text3.getText());

String reply=br.readLine();

(4) 程序测试过程



(5) 运行结果（运行效果）



2.

( 1 )设计思路

客户端输入字符串，使用getBytes()将字符串转换为字节组，发送数据；服务端接收数据，使用toUpperCase()方法将小写转换为大写，再次使用getBytes()将字符串转换为字节组，并发送数据给客户端，

(2)实验步骤

①、建立客户端，编写发送功能。

②、建立服务端，发现接收功能。

③、客户端和服务端功能互换

④、建立测试类，输入字符串。

(3) 关键代码

byte[] bytes=new byte[1024];

DatagramPacket dp=new DatagramPacket(bytes, bytes.length);

System.*out*.println("服务器正在等待数据包...");

ds.receive(dp);

byte[] bytess=dp.getData();

String string=new String(bytess, 0, dp.getLength());

DatagramSocket ds=new DatagramSocket();

Scanner scanner=new Scanner(System.*in*);

System.*out*.print("输入想要发送的字符串：");

String string=scanner.next();

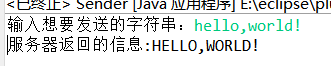
byte[] bytes=string.getBytes();

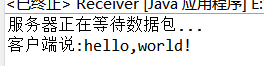
DatagramPacket dp=new DatagramPacket(bytes, bytes.length,InetAddress.*getByName*("192.168.1.102"), 2345);

(4) 程序测试过程



(5) 运行结果（运行效果）





1. **实验总结**

1.发送端：DatagramSocket（无形参）

DatagramPocket(字节数组，数组长度，IP，端口号)

send()发送

2.接收端：DatagramSocket（端口号）

DatagramPocket(字节数组，数组长度)

Receive（）接收

3.UDP协议下无客户端和服务端概念，仅有发送端和接收端，是一种不可靠的网络协议。

1. **附录（程序代码）**
2. **服务端**

**public** **static** **void** main(String[] args) **throws** IOException {

ServerSocket serverSocket=**new** ServerSocket(10086);

System.***out***.println("服务器已启动...");

Socket socket=serverSocket.accept();

InputStream inputStream=socket.getInputStream();

BufferedReader bufferedReader=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream(),"utf-8"));

BufferedWriter bufferedWriter=**new** BufferedWriter(**new** OutputStreamWriter(socket.getOutputStream(),"utf-8"));

System.***out***.println("服务器:接收成功");

String receive=bufferedReader.readLine();

String[] line=receive.split(",");

System.***out***.println("服务器:"+receive);

**double** line1=Double.*valueOf*(line[0]);

**double** line2=Double.*valueOf*(line[1]);

**double** line3=Double.*valueOf*(line[2]);

if(*isRight*(line1,line2,line3)) {

double area=*triangleArea*(line1,line2,line3);

System.*out*.println("服务器:结果为"+area);

bufferedWriter.write(String.*valueOf*(area));

}

else {

System.*out*.println("服务器:输入数据错误");

bufferedWriter.write("输入数据错误，原因：任意两边之和必须大于第三边");

}

System.***out***.println("服务器:发送成功");

bufferedWriter.flush();

bufferedReader.close();

bufferedWriter.close();

socket.close();

serverSocket.close();

}

**public** **static** **double** triangleArea(**double** ...ds) {

**double** p=(ds[0]+ds[1]+ds[2])/2;

**return** Math.*sqrt*(p\*(p-ds[0])\*(p-ds[1])\*(p-ds[2]));

}

public static boolean isRight(double...ds) {

if((ds[0]+ds[1])<=ds[2]) {

return false;

}

return true;

}

**客户端：**

public Button connect,submit;  
public TextField text1,text2,text3;  
public TextArea textArea,showArea;  
private String IP="192.168.1.102";  
private int PORT=10086;  
private String show="";  
private Socket socket;  
private int i=1;  
  
public void connectServer() throws Exception{  
 showNewInfo("第"+i+"次连接\n正在连接服务器...");

try{

socket=new Socket(IP,PORT);  
 showNewInfo("连接服务器成功...");  
 i++;

}catch(ConnectExcption e){

showNewInfo("连接失败，请检查服务器是否启动...");  
 e.printStackTrace();

}  
   
  
}  
public void submitInfo(){  
 Runnable runnable=new Runnable() {  
 @Override  
 public void run() {  
 try {  
  
 BufferedWriter bw=new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(),"utf-8"));  
 BufferedReader br=new BufferedReader(new InputStreamReader(socket.getInputStream(),"utf-8"));  
  
  
 bw.write(text1.getText()+","+text2.getText()+","+text3.getText());  
 bw.newLine();  
 bw.flush();  
  
 showNewInfo("发送数据成功...\n正在等待服务器回复...");  
 String reply=br.readLine();  
 showNewInfo("接收数据成功...");  
 textArea.setText(reply);  
  
 bw.close();  
 br.close();  
 socket.close();  
  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
 };  
 new Thread(runnable).start();  
}  
  
public void showNewInfo(String addString){  
 show=show+"\n"+addString;  
 showArea.setText(show);  
}

服务端：

public static void main(String[] args) throws Exception {

DatagramSocket ds=new DatagramSocket(2345);

byte[] bytes=new byte[1024];

DatagramPacket dp=new DatagramPacket(bytes, bytes.length);

System.*out*.println("服务器正在等待数据包...");

ds.receive(dp);

byte[] bytess=dp.getData();

String string=new String(bytess, 0, dp.getLength());

System.*out*.println("客户端说:"+string);

ds.close();

*sendInfo*(string);

}

public static void sendInfo(String str) throws Exception {

DatagramSocket ds=new DatagramSocket();

byte[] bytes=new byte[1024];

bytes=str.toUpperCase().getBytes();

DatagramPacket dp=new DatagramPacket(bytes, bytes.length,InetAddress.*getByName*("192.168.1.102"), 3456);

ds.send(dp);

ds.close();

}

客户端：

DatagramSocket ds=new DatagramSocket();

Scanner scanner=new Scanner(System.*in*);

System.*out*.print("输入想要发送的字符串：");

String string=scanner.next();

byte[] bytes=string.getBytes();

DatagramPacket dp=new DatagramPacket(bytes, bytes.length,InetAddress.*getByName*("192.168.1.102"), 2345);

ds.send(dp);

ds.close();

*getInfo*();

}

public static void getInfo() throws Exception {

DatagramSocket ds=new DatagramSocket(3456);

byte[] bytes=new byte[1024];

DatagramPacket dp=new DatagramPacket(bytes, bytes.length);

ds.receive(dp);

byte[] bytess=dp.getData();

System.*out*.println("服务器返回的信息:"+new String(bytess, 0, dp.getLength()));

ds.close();

}