# java平台

1、J2SE java开发平台标准版

2、J2EE java开发平台企业版

java程序需要在虚拟机上才可以运行，换言之只要有虚拟机的系统都可以运行java程序。不同系统上要安装对应的虚拟机才可以运行java程序

## 开发步骤

1、编写源文件 (.java)

2、编译源文件为类文件(.class)可用J2SE或J2EE编译

3、在虚拟机上运行

**注释**

//单行注释

/\* \*/多行注释

## java内容介绍

java编程可以分成三个方向：

1、java se (j2se)桌面开发 java中的基础中的基础

2、java ee (j2ee)web开发

3、java me (j2me)手机开发

java se课程介绍

java面向对象编程(基础)

java图开界面开发

java数据库编程

java文件io流编程

java网络编程

java多线程编程

java ee基础1

java面向对象编程--数据库编程-->java se

java 基础2

html--css--javascript-->div+css

java ee中级部分

Servlet--Jsp-->mvc模式

java ee高级部分

Struts--Ejb--Hibernate--Spring--Ajax(ext,dw2)-->ssh框架

java之父gosling

1990 sun启动 绿色计划

1992 创建oak语言-->java

1994 gosling参加硅谷大会演示java功能，震惊世界

1995 sun正式发布java第一个版本，目前最新是jdk7.0

java开发工具

记事本、(jcreator、jbuilder退出舞台了)、netbean、eclipse

如何选择开发工具

先选择记事本，对java有一定了解后再使用eclipse高级开发工具

为什么呢？

1、更深刻的理解java技术，培养代码感

2、有利于公司面试

## java语言的特点

1、java语言是简单的

2、java语言是面向对象的

3、java语言是跨平台(操作系统)的[即一次编译，到处运行]

4、java是高性能的

java第一个程序hello.java

运行java程序要安装和配置jdk

jdk是什么？

1、jdk全称java dvevlopment kit中文java开发工具包

2、jdk是sun公司开发的

3、jdk包括jre(java runtime envirnment)java运行环境、一堆java工具和java基础的类库(类共3600左右，常用类在150个左右)

4、可以在www.sun.com下载

\*\*开发安装jdk,用户执行需要安装jre

配置JDK

添加环境变量即可

windows下配置jdk

在计算机属性--高级设置--环境变量--添加PATH将JDK所在路径指定即可。多个环境变量设置时需要用;号进行隔开

**1、编写第一个hello.java**

**//注释 (解释) 作者：**

**//功能：在控制台显示"hello"**

**//日期：2013.11.28**

**//public：表示这个类是公共的，一个java文件中只能有一个public类**

**//class：表示这是一个类**

**//hello：类名(公共类的类名必须和文件名一致)**

**public class hello{**

**//一个主函数，相当于是程序的入口**

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

**//执行语句**

**//System：是一个包**

**//out.println是输入函数**

**System.out.println("hello!");**

**}**

**}**

**2、编译hello.java 使用javac hello.java**

**3、执行hello.java 使用java hello**

java程序运行关系

1、java源文件(.java文件)

2、java编译器即javac.exe

3、java字节码文件(.class文件)

4、由解释执行器即(java.exe)将字节码文件加载到java虚拟器(jvm)

5、字节码文件(.class)就会在java虚拟机中执行

对hello.java程序进行改过使之变为一个简单的加法运算程序

//注释 (解释) 作者：

//功能：在控制台显示"hello"

//日期：2013.11.28

//public：表示这个类是公共的，一个java文件中只能有一个public类

//class：表示这是一个类

//jiafa：类名(公共类的类名必须和文件名一致)

public class jiafa{

//一个主函数，相当于是程序的入口

public static void main(String args[]){

//执行语句

//System：是一个包

//out.println是输入函数

// System.out.println("hello!");

int a=10;//定义一个变量，变量名a，它的值10

int b=20;//定义一个变量，变量名b，它的值20

int result=a+b;//定义一个叫result变量将变量ab相加的值赋值给result

//输出结果

System.out.println("结果是"+result);

}

}

----------------------------------------------------------------

为什么有变量

不论是使用哪种高级程序语言编写程序，变量都是其程序的基本组成单位。java中的基本数据类型的定义与c/c++中大体一致。

public class Test{

public static void main(String []args){

int a=1;//定义一个整形变量，取名a，并赋初值1

int b=3;//定义一个整形变量，取名b，并赋初值3

b=89;//给变量b赋89

System.out.println(a);//输出语句，把变量a的值输出

System.out.println(b);//把变量b的值输出

}

}

----------------------------------------------------------------

# java基本语法---基本数据类型

**java基本数据类型 四大类型**

**整数类型、小数(浮点)类型、布尔类型、字符类型**

整数类型

可以表示一个整数，常用的整数类型有：byte,short,int,long

主要区别是 数据大小范围，请大家看一个小案例。

byte 占用内存 一个字节 范围：-128至127

short 占用内存 两个字节 范围：-32768至32767

int 占用内存 四个字节 范围：-2147483648至2147483647

long 占用内存 八个字节 范围：-?至?

小数(浮点)类型

可以表示一个小数，常用的小数(浮点)类型有：

float(单精度),double(双精度)

float 占用内存 四个字节 范围：3.4E-38至3.4E+38 只能提供7位有效数字

double占用内存 八个字节 范围：1.7E-308至1.7E+308 可提供16位有效数字

布尔类型

可以表示"真"或者"假"，类型是boolean

比如：

boolean spBool=true; //给变量spBool定义为boolean型并赋值为真

字符类型

可以表示 单个字符，字符类型是char。char是两个字节(可以存放汉字)

多个字符我们称为字符串，在java中String这种数据类型表示，但是String不是基本数据类型，而是类，类是复合数据类型。

结论：在java中，对char进行运算的时候，直接当做ascii码对应的整数对待。

思考：int test1='a'+'b'; 输出值195

char test2='a'+'b'; 输出值?

char test3='中'; 输出值195

## java基本语法--定义变量，初始化，赋值

定义变量

1、什么是定义变量？

int a; 这就是定义了一个变量，变量名是a

float haha;这也定义了一个变量，表示一个float类型的小数，变量名是haha

初始化变量

在定义变量的时候，就给值

int a=45;这就是初始化变量a

给变量赋值

比如你先定义了变量：int tt;

然后再给值tt=780; ->这就是给变量赋值

----------------------------------------------------------------

## java基本语法--基本数据类型转换

自动转换

int a=1.2; double b=3;

结论：数据类型可以自动的从低精度-->高精度。高精度不能转为低精度。

**byte小于<short小于<int小于<long小于<float小于<double**

在java中的小数默认是double数据类型

float赋值时要在值后加f

long赋值时要在值后加l

强制转换

如何把高精度转成低精度？

int a=(int)1.2;

练习int a=1; int a=1;

double b=4.5; double b=4.5;

a=b; b=a;

报错 成功

计算过程中的转换

int a=3;

int b=a+3.4;

结论：当一个整数和一个double运算的时候，运算结果会向高精度转换

## java 基本语法--运算符

**java中常用的算术运算符是：+加、-减、\*乘、/除、%取模**

其实%运算可以得到两个数的余数。

**算术运算符：++自加、--自减。**

这两个运算符比较特殊，请大家思考一下：

int a=90;

a++; 等同于(a=a+1)

这时a等于？ 91

int b=89;

b--; 等同于(b=b-1)

这时b等于？ 88

int a=56;

int b=a++;

System.out.println(b); 56

System.out.println(a); 57

**\*\*\*int b=++a;相当于a=a+1再将b=a；**

**int b=a++;相当于b=a再将a=a+1。**

----------------------------------------------------------------

**算术运算符：+=左加、-=左减、/=左除、%=左取模**

请大家看看几个案例，就明白了：

int a=90;

a+=90; (相当于a=a+90)

请问，a现在等于多少？ 180

float b=89.7f;

b+=a; (相当于b=b+a)

请问，b现在等于多少？ 269.7

int a=56;int b=90;

a-=34;b%=a; (相当于a=a-34,b=b%a)

System.out.println(b); 2

System.out.println(a); 22

----------------------------------------------------------------

运算符

**关系运算符**

**1、==等于；2、>大于；3、<小于；4、>=大于等于；5、<=小于等于；6、!=不等于**

int a=90;int b=90;

if(a==b){System.out.println("ok1");}

b--;

if(a>b){System.out.println("ok2");}

if(a>=b){System.out.println("ok3");}

请编写一个程序，该程序可以接收两个数(可以是整数，也可是小数)并判断两个数是大于？小于？还是等于？

程序代码：

import java.io.\*;//载入IO流包

public class Demo5{

public static void main(String []args){

try{

//输入流，从键盘接收数

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

//给出提示

System.out.println("请输入第一个数");

//从控制台读取一行数据

String a1=br.readLine();

System.out.println("请输入第二个数");

String a2=br.readLine();

//把String转为float

float num1=Float.parseFloat(a1);

float num2=Float.parseFloat(a2);

if(num1>num2){System.out.println("第一个大");}

if(num1==num2){System.out.println("相等");}

if(num1<num2){System.out.println("第二个大");}

}catch(Exception e){

e.printStackTrace();

}

}

}

----------------------------------------------------------------

## 运算符--逻辑运算符

**用于判断逻辑关系的运算符**

**1、&&与；2、||或；3、!非**

&&与的运算关系，真真为真，其它都为假。

||或的运算关系，假假为假，其它都为真。

!非的运算关系，真取假，假取真

请大家看个案例，请问输出什么：

int a=90;int b=90;

if(a==b || a>8){System.out.println("ok1");}

b--;

if(a>b && a>45){System.out.println("ok2");}

if(!(a<=b)){System.out.println("ok3");}

# java基本语法----三大流程控制

顺序控制

听其名而知其意，让程序可以顺序的执行。

请大家看个案例[demo11.java]:

int a=7;

System.out.println("a="+a);

System.out.println("hello!");

a++;

System.out.println("a="+a);

System.out.println("0(∩\_∩)0");

a++;

System.out.println("a="+a);

分支控制

让程序有选择的执行，分支控制有三种：

1. 单分支；2、双分支；3、多分支a

1、单分支语法：

if(条件表达式){

语句;

}

2、双分支语法：

if(条件表达式){

语句;

}else{

语句;

}

3.1、多分支语法：

if(条件表达式){

语句;

}else if(条件表达式){

语句;

}else if(条件表达式){

语句;

}else{

语句;

}

3.2、多分支语法：

switch(条件表达式){ //\*\*\*条件表达式数据类型，就和case常量表达式一致，否则就会报错。

case 常量1:

语句1;

//语句1至break中相当于{}可以加入相应程序代码

break; //break，表示跳出switch语句

case 常量2:

语句1;

break;

...

case 常量n:

语句n;

break;

default:

语句;

break;

}

**\*\*\*\*\*:switch中的可用数据类型主要是：允许byte,short,int,char,enum等**

1、单分支控制作业：请大家看个案例[demo22.java]:

编写一个程序，可以输入人的年龄，如果该同学的年龄大于18岁，则输出“你年龄大于18，要为自己的行为负责!”

2、双分支控制作业：请大家看个案例[demo33.java]:

编写一个程序，可以输入人的年龄，如果该同志的年龄大于18岁，则输出“你年龄大于18岁，要对自己的行为负责!”否则，输出“你的年龄不大这次放过你了”

3、多分支控制作业：请大家看个案例[demo44.java]

编写一个程序，可以输入人的年龄，如果该同志的年龄大于18岁，则输出“你年龄大于18，要对自己的行业负责，送入监狱”。如果该同志的年龄大于10岁并且小于等于18，则输出“你的年龄也不小了，把你送少管所”。其它情况，则输出“小孩子，下次注意”。

4、多分支控制作业：请大家看个案例[demo55.java]

请编写一个程序，该程序可以接收一个字符，比如：a,b,c,d,e,f,g

a表示星期一，b表示星期二...根据用户的输入显示相应的信息，要求使用switch语句完成。

5、多分支控制作业：请大家看个案例[demo66.java]

对比switch与if语句在执行分支语句的区别。

**\*\*\*switch分支语句不支持判断类型即boolean类型，switch只支持byte,short,int,char,enum等数据类型**

-------------------------------------------------------------------------------

## 循环控制

听其名而知其意，就是让你的代码可以循环的执行。

循环控制有三种

1、for循环 语法：

for(循环初值;循环条件;步长){

语句; //循环体

}

2、while循环 语法：

while(循环条件){

语句; //循环体

}

**特别说明：while循环是先判断在执行语句。**

3、do while循环 语法：

do{

语句; //循环体

}while(循环条件);

**特别说明：do while循环是先执行，再判断。**

请大家看个案例[demo77-88-99.java]:

编写一个程序，可以打印10句“你好，我是刘德华同志!”。请大家想想怎么做？

1、for循环

for (int a=0;a<10;a++){System.out.println("你好，我是刘德华同志!");}

2、while循环

int i=0;while(i<10){System.out.println("你好，我是刘德华同志!");i++;}

3、do while循环

int i=0;do{System.out.println("你好，我是刘德华同志!");i++;}while(i<10)

请编写一个程序，可以接收一个整数，表示层数，打印出金字塔一半。[demo100.java]

\*\*\*把一个复杂的问是先拆解成简单问题

//三角形[Demo100.java]

public class Demo100{

public static void main(String []args){

//死去活来法-->打四次

int lay=4;

//表示有多少层

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

//打印\*

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

System.out.print("\*");

}

System.out.println();//换行

}

}

}

-------------------------------------------------------------------------------

//金字塔[Demo101.java]

public class Demo101{

public static void main(String []args){

int lay=10;//表示有多少层

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

//找出规律

//1->3 2->2 3->1 4->0找出空格

for(int k=1;k<=lay-i;k++){

System.out.print(" ");

}

//打印\*

//1->1 2->3 3->5 4->7找出星的规律

for(int j=1;j<=(i-1)\*2+1;j++){

System.out.print("\*");

}

System.out.println();//换行

}

}

}

-------------------------------------------------------------------------------

//镂空金字塔[Demo102.java]

public class Demo102{

public static void main(String []args){

int lay=10;//表示有多少层

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

//找出规律

//1->3 2->2 3->1 4->0找出空格

for(int k=1;k<=lay-i;k++){

System.out.print(" ");

}

//打印\*

//1->1 2->3 3->5 4->7找出星的规律

for(int j=1;j<=(i-1)\*2+1;j++){

//判断该层是否是顶层或者是底层

if(i==1||i==lay){

System.out.print("\*");

}else{

//如果是第一个\*

if(j==1||j==(i-1)\*2+1){

System.out.print("\*");

}else{

System.out.print(" ");

}

}

}

System.out.println();//换行

}

}

}

-------------------------------------------------------------------------------

//空心菱形[Demo103.java]

public class Demo103{

public static void main(String []args){

int lay=4;//菱形上半部行数

for(int i=1;i<=lay-1;i++){ //判断循环上半部分

for(int k=1;k<=lay-i;k++){ //找行内\*号前输出空格

System.out.print(" ");

}

for(int j=1;j<=(i-1)\*2+1;j++){ //找行内输出星的位置

if(j==1||j==(i-1)\*2+1){ //判断行最小长度和最大长度，并在最小长度和最大长度上输出\*号，行内输入出空格。

System.out.print("\*");

}else{

System.out.print(" ");

}

}

System.out.println(); //换行

}

for(int i=1;i<=lay;i++){ //判断循环菱形下半部分

for(int k=1;k<=i-1;k++){ //判断循环找\*号前要输出空格处

System.out.print(" ");

}

for(int j=1;j<=(lay-i)\*2+1;j++){ //判断循环行长度

if(j==1||j==(lay-i)\*2+1){ //判断行最小长度和最大长度，并在最小长度和最大长度上输出\*号，行内输入出空格。

System.out.print("\*");

}else{

System.out.print(" ");

}

}

System.out.println();//输出换行

}

}

}

-------------------------------------------------------------------------------

//实心菱形[Demo104.java]

public class Demo104{

public static void main(String []args){

int lay=4;//菱形上半部行数

for(int i=1;i<=lay-1;i++){ //判断循环上半部分

for(int k=1;k<=lay-i;k++){ //找行内\*号前输出空格

System.out.print(" ");

}

for(int j=1;j<=(i-1)\*2+1;j++){ //找行内输出星的位置

System.out.print("\*");

}

System.out.println(); //换行

}

for(int i=1;i<=lay;i++){ //判断循环菱形下半部分

for(int k=1;k<=i-1;k++){ //判断循环找\*号前要输出空格处

System.out.print(" ");

}

for(int j=1;j<=(lay-i)\*2+1;j++){ //判断循环行长度

System.out.print("\*");

}

System.out.println();//输出换行

}

}

}

-------------------------------------------------------------------------------

# 1、java面向对象编程(1)-类与对象

一个问题？[Demo107.java]

张老太养了两只猫猫：一只名字叫小白，今年3岁，白色。还有一只叫小花，今年100岁，花色。请编写一个程序，当用户输入小猫的名字时，就显示该猫的名字，年龄，颜色。如果用户输入的小猫名错误，则显示张老太没有这只猫猫。

//用前面学习过的知识写出代码

public class Demo107{

public static void main(String []args){

int a=49;//输入的名字49，50

int cat1age=3; //第一只猫

String cat1name="小白";

String cat1color="白色";

int cat2age=100; //第二只猫

String cat2name="小花";

String cat2color="花色";

switch(a){

case '1':

System.out.println(cat1age+cat1color);

break;

case '2':

System.out.println(cat2age+cat2color);

break;

default:

System.out.println("张老太没有这只猫!");

}

}

}

java语言是面向对象的

计算机语言的发展向接近人的思维方式演变

汇编语言 [面向机器]

c语言 [面向过程]

java语言 [面向对象]

类和对象的关系

把猫的特性提取出来-->猫类-->对象(实例)

-->对象(实例)

-->...

注意：从猫类到对象，目前有几种说法：1、创建一个对象；2、实例化一个对象；3、对类实例化...以后大家听到这些说法，不要模糊。**(对象就是实例，实例就是对象)java最大的特点就是面向对象。**

//定义猫类

public class Demo105{

public static void main(String []args){

//创建一个猫对象

Cat cat1=new Cat();//Cat是定义的一个数据类型

**//Cat cat1;**

**//cat1=new Cat();// 等同于 Cat cat1=new Cat();**

**//访问属性的 对象名.属性名字**

cat1.age=3;

cat1.name="小白";

cat1.color="白色";

//创建第二只猫

Cat cat2=new Cat();

cat2.age=100;

cat2.name="小花";

cat2.color="花色";

}

}

//java中如何定义一个类？**[类名的首写字母大写]**可根据程序的需要定义类

class Cat{

//下面的就是类的成员变量/属性

int agr;

String name;

String color;

Master myMaster;

}

//引用类型，比如建个主人类

class Master{

int age;

String name;

String address;

}

## 类和对象的区别和联系

1、类是抽象的，概念的，代表一类事物，比如人类，猫类..

2、对象是具体的，实际的，代表一个具体事物

3、类对象的模板，对象是类的一个个体，实例

**类--如何定义类**

一个全面的类定义比较复杂，如：

**package 包名;**

**class 类名 extends 父类 implements**

**接口名{**

**成员属性;**

**构造方法;**

**成员方法;**

**}**

要透彻的掌握类，必须要了解类的构成

class 类名{ ----> 待定...

成员变量;

}

## 类--类的成员变量

成员变量是类的一个组成部分，一般是基本数据类型，也可是引用类型。比如我们前面定义猫类的int age 就是成员变量。

## 对象--如何创建对象

创建一个对象有两种方法

1、先声明再创建

**1、对象声明：类名 对象名**

**2、对象创建：对象名=new 类名()**

2、一步到位法

**类名 对象名=new 类名()**

对象--如何访问(使用)对象的成员变量

**对象名.变量名; //简单先这样理解，以后加下控制符此表达就不准确了。**

\*\*\*对象总是存在内存中的

一个小思考题[Demo.105.java]

为了让大家加深印象，我们定义一个人类(Person)(包括名字、年龄)。用一步到位法去创建一个对象

↓

我们看看下面一段代码: System.out.printlin(b.age);

Person a=new Person(); → 请问：b.age究竟是多少？

a.age=10;

a.name="小明";

Person b;

b=a;

\*\*\*对象总是存在内存中的

一个小思考题[Demo106.java]

在明白对象是如何在内存中存在后，请大家再看看下面的思考题，请问会输出什么信息？

Person1 a=new Person1();

a.age=10;

a.name="小明";

Person1 b;

b=a;

System.out.println(b.name);//输出“小明”

b.age=200;

System.out.println(a.age);//输出a.age为200

\*\*\*\*\*重点也是难点

类--成员方法的初步介绍

在某些情况下，我们要需要定义成员方法。比如人类：除了有一些属性外(成员变量表示的年龄、姓名...)，我们人类还有一些行为比如：可以说话、跑步..，通过学习，我们人类还可以做算术题。这时就要用成员方法才能完成。现在要求对Person类完善：

1、添加speak成员方法，输入出：我是一个好人

2、添加jisuan成员方法，可以计算从1+..+1000的结果

3、修改jisuan成员方法，该方法可以接收一个数n，计算从1+..+n的结果

4、添加add成员方法，可以计算两个数的和

## 类--类的成员方法(成员函数)定义

**成员方法也叫成员函数**，这里希望大家不要被两个名词搞晕了。

**public 返回数据类型 方法名(参数列表)**

**{**

**语句;//方法(函数)主体**

**}**

1、参数列表：表示成员函数输入

2、数据类型(返回类型)：表示成员函数输出

3、函数主体：表示为了实现某一功能代码块

//类的调用及方法调用[Demo108.java]

public class Demo108{

public static void main(String []args){

Person p1=new Person();

p1.speak();//调用speak方法

p1.jiSuan();//调用计算方法

p1.jiSuan(200);//调用可以传入参数的计算方法

p1.add(12,10);//调用两个数的和

int res=p1.add2(23,34);//调用两个数的和并返回值到res中

System.out.println("res返回值是："+res);

System.out.println("num1+num2+num3="+p1.add3(2,2.3f,4.5f));//返回类型一定要一致否则报错。

}

}

**//定义名字的几个方法：**

**//1、驼峰法 如myCry；2、下划线法my\_cry**

**//方法名在有不同参数的情况下可以使用同一个方法名，即有参数和没参数的方法可以同名**

class Person{ **//请注意类名首写字母应为大写如Person为类名**

int age;

String name;

//1、可以输出我是好人方法

public void speak(){ **//请注意方法名的首写字母应为小写如speak为方法名**

System.out.println("我是一个好人");

}

//2、可以计算1+..+1000的方法

public void jiSuan(){

int result=0;

for(int i=1;i<=1000;i++){

result=result+i;

}

System.out.println("1+..+1000结果是"+result);

}

//3、带参数的成员方法,可以输入n值并计算1+..+n

public void jiSuan(int n){

int result=0;

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

result+=i;

}

System.out.println("1+..+n结果是"+result);

}

//4、计算两个数的和

public void add(int num1,int num2){

int result=0; **//与下面一句等同于return num1+num2;**

result=num1+num2;

System.out.println("num1+num2="+result);

}

//5、计算两个数的和，并将结果返回给主调(调用它的)函数

**//注意：返回类型和返回结果的类型要一致**

**//注意：在调用某个成员方法的时候，给出的具体数值的个数**

**//和类型要相匹配。**

public int add2(int num1,int num2){

return num1+num2;

}

//6、计算两个float数的和，并将结果返给主调函数

public float add3(int num1,float num2,float num3){

return num1+num2+num3;

}

}

## 类--类的成员方法(函数)--如何理解

如何理解方法这个概念，给大家举个通俗的示例：

程序员调用方法：给方法必要的输入，方法返回结果。

## 类的成员方法--声明

**public int test(int a);**/\*方法声明\*/

这句话的作用是声明该方法，声明的格式为：

**访问修饰符 数据类型 函数名(参数列表);**

在给Person类添加add方法的例题中，我们看到的关键字**return，它的功能是把表达式的值返回的值返回给主调函数的方法。**

**return 表达式;**

类的成员方法(函数)--特别说明

1、方法的参数列表可以是多个

案例：在Person类中编写一个成员方法，从键盘输入三个数，返回最大的那个数。

**参数列表可以是多个，并且数据类型可以是任意的类型int float double char..**

**访问修饰符 返回数据类型 函数名(参数列表){**

**语句; //函数主体**

**}**

2、方法可以没有返回值

案例：编写一个函数，从控制台输入一个整数打印出对应的金字塔。

**返回类型可以是任意的数据类型(int,float,double,char..)也可以没有返回值void表示没有返回值**

**访问修饰符 返回数据类型 函数名(形参列表){**

**语句; //函数主体**

**}**

类的成员方法(函数)--小练习

案例：编写一个成员函数，从键盘输入一个整数(1-9)，打印出对应的乘法表[Demo110.java]

//实例键盘输入打印乘法表[Demo110.java]

import java.io.\*;

public class Demo110{

public static void main(String []args){

Cfb jiu=new Cfb();

jiu.cf();

}

}

class Cfb{

public void cf(){

try{

//输入流，从键盘接收数

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

//给出提示

System.out.println("请输入1-9，按0退出:");

//从控制台读取一行数据

String a1=br.readLine();

//把String转为int

int num1=Integer.decode(a1);

for(int i=1;i<=num1;i++){

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

System.out.print(i+"×"+j+"="+(i\*j)+"\t");

}

System.out.println();

}

}catch(Exception e){

e.printStackTrace();

}

}

}

案例：编写函数，使给定的一个二维数组(3×3)转置

类定义的完善

在介绍了成员方法后，我们类的定义就可以完善一步：

class 类名{ class 类名{ 待定

成员变量; → 成员变量; →

} 成员方法;

}

小结：1.先设计类；2.然后根据类创建对象。

小练习：

1、设计计算机类，要求如下：[Demo109.java]

属性：品牌(Brand)、颜色(Color)、cpu型号(CPU)、内存容量(Memory)、硬盘大小(Harddisk)、价格(Price)、工作状态(Work)

方法：打开(Open)、关闭(Close)、休眠(Sleep)

创建一个计算机对象，调用打开，关闭方法

//计算机类与对象的代码

import java.io.\*;//加载IO流包

public class Demo109{

public static void main(String []args){

Computer Pc=new Computer();

Pc.Brand="品牌";

Pc.Color="颜色";

Pc.Cpu="Cpu型号";

Pc.Memory="内存容量";

Pc.Hd="硬盘容量";

Pc.Price="价格";

Pc.Work="工作状态";

try{

//输入流，从键盘接收数

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

//给出提示

System.out.println("请输入0-9控制机器");

//从控制台读取一行数据

String a1=br.readLine();

//把String转为float

float num1=Float.parseFloat(a1);

if(num1==0){Pc.open();}

else if(num1==1){Pc.close();}

else if(num1==2){Pc.sleep();}

else if(num1==3){System.out.println(Pc.Brand);}

else if(num1==4){System.out.println(Pc.Color);}

else if(num1==5){System.out.println(Pc.Cpu);}

else if(num1==6){System.out.println(Pc.Memory);}

else if(num1==7){System.out.println(Pc.Hd);}

else if(num1==8){System.out.println(Pc.Price);}

else if(num1==9){System.out.println(Pc.Work);}

else {System.out.println("输入错误!");}

}catch(Exception e){

e.printStackTrace();

}

}

}

class Computer{

String Brand;

String Color;

String Cpu;

String Memory;

String Hd;

String Price;

String Work;

public void open(){

System.out.println("开机");

}

public void close(){

System.out.println("关机");

}

public void sleep(){

System.out.println("休眠");

}

}

1. 采用面向对象思想设计超级马里奥游戏人物

-------------------------------------------------------------------------------

# 2、java面向对象编程(1)-构造方法(函数)

类的构造方法介绍

什么是构造方法呢？在回答这个问题之前，我们来看一个需求：前面我们在创建人类的对象时，是先把一个对象创建好后，再给他的年龄和姓名属性赋值，如果现在我要求，在创建人类的对象时，就直接指定这个对象的年龄和姓名，该怎么做？

↓

你可以在定义类的时候，定义一个构造方法即可。

构造方法是类的一种特殊的方法，它的主要作用是完成对新对象的初始化。它有几个特点：

1、方法名和类名相同

2、没有返回值

3、在创建一个类的新对象时，系统会自动的调用该类的构造方法完成对新对象的初始化。

**特别说明：**

**一个类可以定义多个不同的构造方法。**

//例人类构造方法[Demo111.java]

public class Demo111{

public static void main(String []args){

Person p1=new Person(12,"顺平");//给予不同的初始值，调用的构造方法不同，构造方法虽同名，但系统会根据初始值来选定构造方法。

}

}

//定义一个人类

class Person{

int age;

String name;

**//默认构造方法**

**public Person(){**

**}**

//构造方法的主要用处是：初始化你的成员属性(变量)

//构造方法

public Person(int age,String name){

System.out.println("我是构造1");

age=age;

name=name;

}

//构造方法2

public Person(String name){

System.out.println("我是构造2");

name=name;

}

}

## 类的默认构造方法

有些同志可能会问？亲爱的老师，我们在没有学习构造函数前不是也可以创建对象吗？

是这样的，如果程序员没有定义构造方法，系统会自动生成一个默认构造方法。比如Person类Person (){};

当创建一个Person对象时Person per1=new Person();**默认的构造函数就会被自动调用。**

**类的构造方法小结：**

**1、构造方法名和类名相同；**

**2、构造方法没有返回值；**

**3、主要作用是完成对新对象的初始化；**

**4、在创建新对象时，系统自动的调用该类的构造方法；**

**5、一个类可以有多个构造方法；**

**6、每个类都有一个默认的构造方法。**

类定义的改进

在提出构造方法后，我们类的定义就应该更加完善了：

class 类名{ class 类名{ class 类名{

成员变量; 成员变量; 成员变量;

} → 成员方法; → 构造方法; → 待定..

} 成员方法

}

## java面向对象编程(2)--this

一个问题？

请大家看一段代码：(Demo112.java)

**\*重点\*：this是属于一个对象，不属于类的。**

java虚拟机会给每个对象分配this,代表当前对象。坦白的讲，要明白this不是件容易的事

**注意事项：this不能在类定义的外部使用，只能在类定义的方法中使用**

/\*

this的必要性

\*/

public class Demo112{

public static void main(String []args){

Dog dog1=new Dog(2,"大黄");

Person p1=new Person(dog1,23,"郭德纲");

Person p2=new Person(dog1,24,"刘谦");

p1.showInfo();

p1.dog.showInfo();

}

}

//定义一个人类

class Person{

//成员变量

int age;

String name;

Dog dog;//引用类型

public Person(Dog dog,int age,String name){

//可读性不好

//age=age;

//name=name;

this.age=age; //this.age指this代词指定是成员变量age

this.name=name; //this.name指this代词指定是成员变量name

this.dog=dog;

}

//显示人名字

public void showInfo(){

System.out.println("人名是："+this.name);

}

}

class Dog{

int age;

String name;

public Dog(int age,String name){

this.age=age;

this.name=name;

}

//显示狗名

public void showInfo(){

System.out.println("狗名叫"+this.name);

}

}

-------------------------------------------------------------------------------

类变量--提出问题？

提出问题的主要目的就是让大家思考解决之道。

public class Demo113{

public static void main(String []args){

/\* int total=0;

Child ch1=new Child(3,"妞妞");

ch1.joinGame();

total++;

Child ch2=new Child(4,"小小");

ch2.joinGame();

total++;

\*/

Child ch1=new Child(3,"妞妞");

ch1.joinGame();

Child ch2=new Child(4,"小小");

ch2.joinGame();

Child ch3=new Child(5,"大大");

ch3.joinGame();

System.out.println("共有="+Child.total);

}

}

//定义小孩类

class Child{

int age;

String name;

//static公共函数,total是静态变量，因此它可以被任何一个对象访问

static int total=0;

public Child(int age,String name){

this.age=age;

this.name=name;

}

public void joinGame(){

total++;

System.out.println("有一个小孩加入了");

}

}

-------------------------------------------------------------------------------

## java面向对象编程(2)--类变量、类方法

什么是类变量？

类变量是该类的所有对象共享的变量，任何一个该类的对象去访问它时，取到的都是相同的值，同样任何一个该类的对象去修改它时，修改的也是同一个变量。

如何定义类变量？

定义语法：

**访问修饰符 static 数据类型 变量名;**

如何访问类变量？

**类名.类变量名** 或者 **对象名.类变量名**

//类变量的程序演示[Demo114.java]

public class Demo114{

static int i=1;

static{

//该静态区域块只被执行一次

i++;

System.out.println("执行一次");

}

public Demo114(){ //建立Demo114()构造函数

System.out.println("执行二次");

i++;

}

public static void main(String []args){

Demo114 t1=new Demo114(); //创建t1对象实例并调用Demo114函数

System.out.println(t1.i);

Demo114 t2=new Demo114();

System.out.println(t2.i);

}

}

-------------------------------------------------------------------------------

## 什么是类方法，为什么有类方法？

类方法是属于所有对象实例的，其形式如下：

**访问修饰符 static 数据返回类型 方法名(){}**

**注意：类方法中不能访问非静态变量(类变量)。**

使用：**类名.类方法名** 或者 **对象名.类方法名**

**\*重点\*static静态的方法可以访问static静态变量，不能访问非静态变量(类变量)**

**非静态方法可以访问非静态变量(类变量)同时也可以访问static静态变量。**

//统计总学费的程序代码，加深static静态的方法由静态变量的访问[Demo115.java]

public class Demo115{

public static void main(String []args){

//创建一个学生

Stu stu1=new Stu(29,"aa",340);

Stu stu2=new Stu(29,"aa",240);

System.out.println(Stu.getTotalFee());

}

}

//学生类

class Stu{

int age;

String name;

int fee;

static int totalFee;

public Stu(int age,String name,int fee){

this.age=age;

this.name=name;

totalFee+=fee;

}

//返回总学费[这是一个类方法(静态方法)]

**//java中规则：类变量原则上用类方法去访问或操作**

public static int getTotalFee(){

return totalFee;

}

}

## 类变量小结

1、什么时候需要用类变量

案例[Demo115.java]：定义学生类，统计学生共交多少钱？

用类变量，属于公共的属性

2、类变量与实例变量区别：

**加上static称为类变量或静态变量，否则称为实例变量**

**类变量是与类相关的，公共的属性**

**实例变量属于每个对象个体的属性**

**类变量可以通过 [类名.类变量名] 直接访问**

类方法小结

1、什么时候需要用类方法

案例[Demo115.java]：定义学生类，统计学生共交多少钱？

**类方法属于与类相关的，公共的方法**

**实例方法属于每个对象个体的方法**

**类方法可以通过 [类名.类方法名] 直接访问**

-------------------------------------------------------------------------------

**java面向对象编程的四大特征**

**封装/继承/多态/抽象**

# 封装--什么是封装

封装就是把抽象出来的数据和对数据的操作封装在一起，数据被保护在内部，程序的其它部分只有通过被授权的操作(成员方法)，才能对数据进行操作。

## 封装--访问控制修饰符

电视机的开关，对音量，颜色，频道的控制是公开的，谁都可以操作，但是对机箱后盖，主机板的操作却不是公开的，一般是由专业维修人员来玩。那么java中如何实现这种类似的控制呢？不能随便查看人的年龄，工资等隐私[Demo116.java]

//封装案例[Demo116.java]

public class Demo116{

public static void main(String []args){

//创建一个职员

Clerk clerk1=new Clerk("小花",24,4567.6f);

System.out.println("名字是"+clerk1.name+"薪水"+clerk1.getSal());

}

}

//职员

class Clerk{

public String name;

**//private私有的，public公有的**

private int age;

private float salary;

public Clerk(String name,int age,float sal){

this.name=name;

this.age=age;

this.salary=sal;

}

//通过一个成员方法去控制和访问私有的属性

public float getSal(){

return this.salary;

}

}

## 封装--访问控制修饰符

java提供四种访问控制修饰符号控制方法和变量的访问权限：

**1、公开级别：用public修饰，对外公开**

**2、受保护级别：用protected修饰，对子类和同一个包中的类公开**

**3、默认级别：没有修饰符号，向同一个包的类公开**

**4、私有级别：用private修饰，只有类本身可以访问，不对外公开**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **4种访问级别的访问范围** | | | | | |
| **访问级别** | **访问控制修饰符** | **同类** | **同包** | **子类** | **不同包** |
| **公 开** | **Public** | **√** | **√** | **√** | **√** |
| **受保护** | **Protected** | **√** | **√** | **√** | **╳** |
| **默 认** | **没有修饰符** | **√** | **√** | **╳** | **╳** |
| **私 有** | **private** | **√** | **╳** | **╳** | **╳** |

# 包--必要性

问题的提出，请看下面的一个场景[eclipse]

现在有两个程序员共同开发一个java项目，程序员xiaoming希望定义一个类取名Dog，程序员xiaoqiang也想定义一个类也叫Dog。两个程序员为此还吵了起来，怎么办？

**包--三大作用**

1、区分相同名字的类

2、当类很多时，可以很好的管理类

3、控制访问范围

**包--换包命令**

**package com.自定义名字;**

注意：打包命令一般放在文件开始处。

包--命名规范

**小写字母** 比如 com.sina.shunping

包--常用的包

一个包下，包含很多的类，java中常用的包有：

java.lang.\* 包 自动引入 java.util.\* 工具包

java.net.\* 包 网络开发包 java.awt.\* 包 窗口工具包

## 包--如何引入包

**语法：import 包;**

比如import java.awt.\*;

我们引入一个包的主要目的要使用该包下的类

定义类的改进

在提出包后，我们类的定义就更加完善了：

class 类名{ class 类名{ class类名{ package包名; 待定..

成员变量; → 成员变量; → 成员变量; → class 类名{ →

} 成员方法; 构造方法; 成员变量;

} 成员方法; 构造方法;

} 成员方法;

}

-------------------------------------------------------------------------------

# **继承--为什么有？**

[Demo117.java]

//功能：说明继承的重要性

package com.abc;//包名

public class Demo117 {

public static void main(String[] args) {

Pupil p1=new Pupil();

p1.printName();

}

}

//将学生的共有属性提取，做一个父类

class Stu{

//定义成员属性

protected int age;

public String name;

public float fee;

private String job;//私有将不被继承

//编程中，如果你不希望子类继承某个属性或方法

//则将其声明为private即可

public void printName(){

System.out.println("名字"+this.name);

}

}

//小学生类

class Pupil extends Stu{

//交学费

public void pay(float fee){

this.fee=fee;

}

}

//幼儿

class Pre extends Pupil{

//交学费

public void pay(float fee){

this.fee=fee\*1.5f;

}

}

//中学生类

class MiddleStu extends Stu{

//交学费

public void pay(float fee){

this.fee=fee\*0.8f;

}

}

//大学生类

class ColStu extends Stu{

//交学费

public void pay(float fee){

this.fee=fee\*0.1f;

}

}

## 继承--解决之道

继承可以解决代码复用，让我们的编程更加靠近人类思维。当多个类存在相同的属性(变量)和方法时，可以从这些类中抽象出父类(比如刚才的Student)，在父类中定义这些相同的属性和方法，所有的子类不需要重新定义这些属性和方法，只需要通过extends语句来声明继承父类：

语法：class 子类 **extends 父类**

这样，子类就会自动拥有父类定义的某些属性和方法。

继承--深入讨论

1、父类的哪些属性(变量)、方法被子类继承了？并不是父类的所有属性、方法都可以被子类继承

父类 子类

public 属性; public 属性;

protected 属性; 继承 protected 属性;

private 属性; → 属性;

属性;

public 方法; public 方法;

protected 方法; protected 方法;

private 方法; 方法;

方法;

2、结论

从图可以看出，父类的public修饰符的属性和方法；protected修饰符的属性和方法；默认修饰符属性和方法被子类继承了，父类的private修饰符的属性和方法不能被子类继承。

继承--注意事项

1、子类最多只能继承一个父类(指直接继承)

2、java所有类都是Object类的子类 (所有的子类都可以逐级继承，例:爷->父->子->孙)

3、JDK6中有202个包3777个类、接口、异常、枚举、注释和错误

4、在做开发的时候，强烈建议大家多查jdk帮助文档

5、在使用类时，实在不知道怎么办，多使用搜索引擎

**定义类的改进**

在提出包后，我们类的定义就更加完善了：

class 类名{ class 类名{ class类名{ package包名;

成员变量; → 成员变量; → 成员变量; → class 类名{

} 成员方法; 构造方法; 成员变量;

} 成员方法; 构造方法;

} 成员方法;

} ↓

↓←←←←←←←←←←←←←←←←←←←←←←←←←

package 包名;

class 类名 extends 父类{ 待定

成员变量; → ....

构造方法;

成员方法;

}

-------------------------------------------------------------------------------

# java面向对象编程(2)--方法重载(overload)

**方法重载(overload)**

按顺序，我们应该讲解多态，但是在讲解多态前，我们必须讲解方法重载和方法覆盖(override)。

请编写一个类(Abc)，编写方法可以接收两个整数，返回两个数中较大的数[Demo119.java]

//方法重载(overload)getMax

public class Demo119{

public static void main(String []args){

Abc2 abc1=new Abc2();

System.out.println(abc1.getMax(12,14));

System.out.println(abc1.getMax(24f,20f));

}

}

class Abc2{

//返回较大的整数

public int getMax(int i,int j){

if(i>j){

return i;

}else{

return j;

}

}

public float getMax(float a,float b){

if(a>b){

return a;

}else{

return b;

}

}

//如果只是返回类型不一样，能否构成重载？不能够构成重载

/\* public double getMax(float d,double c){

if(c>d){

return c;

}else{

return d;

}

}

//如果只是控制访问修饰符不同，能否构成重载？不能够构成重载

protected float getMax(float c,float d){

if(c>d){

return c;

}else{

return d;

}

}\*/

}

## 方法重载(overload)概念

简单的说：方法重载就是在类的同一种功能的多种实现方式，到底采用哪种方式，取决于调用者给出的参数。

**注意事项：**

1、方法名相同

2、方法的参数类型，个数，顺序至少有一项不同

3、方法返回类型可以不同(只是返回类型不一样，不能构成重载)

4、方法的修饰符可以不同(只是控制访问修饰符不同，不能构成重载)

## 方法覆盖(override)

既然子类可以继承父类的属性和方法，这样可以提高代码的复用性，这个很好，可是问题来了，假设现在我要求大家写三个类猫猫，狗狗，猪猪。我们知道这三个东东都是动物，动物必然存在相同的特点。根据类的抽象特征，我们可以把它们的相同点提取出来，形成一个父类，然后继承。

//子类方法覆盖父类方法[Demo120.java]

public class Demo120{

public static void main(String []args){

//创建一只猫

Cat cat1=new Cat();

cat1.cry();

Dog dog1=new Dog();

dog1.cry();

}

}

//动物类

class Animal{

int age;

String name;

//都会叫

public void cry(){

System.out.println("我是动物，不知道怎么叫");

}

}

//猫猫类

class Cat extends Animal{

//覆盖父类方法

public void cry(){

System.out.println("猫猫叫!");

}

}

//狗狗类

class Dog extends Animal{

//覆盖父类方法

public void cry(){

System.out.println("汪汪叫!");

}

}

## 方法覆盖(override)概念

简单的说：方法覆盖就是子类有一个方法，和父类的某个方法的名称、返回类型、参数一样，那么我们就说子类的这个方法覆盖了父类的那个方法。比如上个案例的Cat类中的cry方法就覆盖了Animal类的cry方法。

**注意事项：**

方法覆盖有很多条件，有些书上说的比较细，总的讲有两点一定注意：

1、子类的方法的返回类型，参数，方法名称，要和父类的返回类型，参数，方法名称完全一样，否则编译出错。

2、子类方法不能缩小父类方法的访问权限。

===============================================================================

作业：上机实习题目

1、Josephu问题(丢手帕问题)

Josephu问题为：设编号为1,2,...n的n个人围坐一圈，约定编号为k(1<=k<=n)的人从1开始报数，数到m的那个人出列，它的下一位又从1开始报数，数到m的那个人又出列，依次类推，直到所有人出列为止，由此产生一个出队编号的序列。

提示：用一个不带头结点的循环链表来处理Josephu问题：先构成一个有n个结点的单循环链表，然后由k结点起从1开始计数，计到m时，对应结点的人从链表中删除，然后再从被删除结点的下一个结点又从1开始计数，直到最后一个结点从链表中删除算法结束。

//Josephu问题(丢手帕问题)

//使用单向链表

public class Demo121 {

public static void main(String[] args) {

CycLink cyclink=new CycLink();

cyclink.setLen(5);//链表长度

cyclink.createLink();

cyclink.setK(2);//从第几个人开始数

cyclink.setM(2);//数几下

cyclink.show();

cyclink.play();

}

}

class Child{

int no;

Child nextChild=null;

public Child(int no){

//给一个编号

this.no=no;

}

}

//单向环形链表

class CycLink{

//先定义一个指向链表第一个小孩的引用

//指向第一个小孩的引用，不能动

Child firstChild=null;

Child temp=null;

int len=0;//表示共有多少个小孩

int k=0;

int m=0;

//设置m数几下

public void setM(int m){

this.m=m;

}

//设置环形链表大小

public void setLen(int len){

this.len=len;

}

//设置从第几个人开始数数

public void setK(int k){

this.k=k;

}

//开始play

public void play(){

Child temp=this.firstChild;

//1.先找到开始数数的人

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

temp=temp.nextChild;

}

while(this.len!=1){

//2.数m下

for(int j=1;j<m;j++){

temp=temp.nextChild;

}

//找到要出圈的前一个小孩

Child temp2=temp;

while(temp2.nextChild!=temp){

temp2=temp2.nextChild;

}

//3.将数到m的小孩，退出圈

temp2.nextChild=temp.nextChild;

//让temp指向下一个数数的小孩

temp=temp.nextChild;

this.len--;

}

//最后一个小孩

System.out.println("最后出圈的小孩:"+temp.no);

}

//初始化单向环形链表

public void createLink(){

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

if(i==1){

//创建第一个小孩

Child ch=new Child(i);

this.firstChild=ch;

this.temp=ch;

}else{

//创建最后一个小孩

if(i==len){

Child ch=new Child(i);

temp.nextChild=ch;

temp=ch;

temp.nextChild=this.firstChild;

}else{

//继续创建小孩

Child ch=new Child(i);

temp.nextChild=ch;

temp=ch;

}

}

}

}

//打印该环形链表

public void show(){

//定义一个跑龙套

Child temp=this.firstChild;

do{

System.out.print(temp.no+" ");

temp=temp.nextChild;

}while(temp!=this.firstChild);

}

}

**java面向对象编程(2)--四大特征**

# 多态--概念

所谓多态，就是指一个引用(类型)在不同情况下的多种状态。也可以理解成：多态是指通过指向父类的指针，来调用在不同子类中实现的方法。

**实现多态有两种方式：**1、继承；2、接口

多态演示[Dome122.java]

//演示继承、方法覆盖、多态

public class Demo122 {

public static void main(String[] args) {

//非多态演示

Cat cat=new Cat();

cat.cry();

Dog dog=new Dog();

dog.cry();

//多态演示

Animal an=new Cat();

an.cry();

an=new Dog();

an.cry();

}

}

//动物类

class Animal{

String name;

int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

//动物会叫

public void cry(){

System.out.println("不知道怎么叫");

}

}

//创建Dog子类并继承Animal父类及覆盖cry方法

class Dog extends Animal{

//狗叫

public void cry(){

System.out.println("汪汪叫");

}

}

class Cat extends Animal{

//猫自己叫

public void cry(){

System.out.println("猫猫叫");

}

}

-------------------------------------------------------------------------------

**多重多态演示[Dome123.java]**

//演示子类继承父类、方法覆盖、多态方法

public class Demo123 {

public static void main(String[] args) {

//非多态演示

System.out.println("非多态演示:");

Cat cat=new Cat();

cat.cry();

Dog dog=new Dog();

dog.cry();

System.out.println();

//多态演示

System.out.println("多态演示:");

Animal an=new Cat();

an.cry();

an=new Dog();

an.cry();

System.out.println();

//多重多态演示

System.out.println("多重多态演示:");

Master master=new Master();

master.feed(new Dog(),new Bone());

master.feed(new Cat(),new Fish());

}

}

//主人类

class Master{

//给动物喂食物，使用多态，只要写一个方法

public void feed(Animal an,Food f){

an.eat();

f.showName();

}

}

//食物父类

class Food{

String name;

public void showName(){

System.out.println("食物");

}

}

//食物鱼子类

class Fish extends Food{

public void showName(){

System.out.println("鱼");

}

}

//食物骨头子类

class Bone extends Food{

public void showName(){

System.out.println("骨头");

}

}

//动物类Animal父类

class Animal{

String name;

int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

//动物会叫

public void cry(){

System.out.println("不知道怎么叫");

}

//动物吃东西

public void eat(){

System.out.println("不知道吃什么");

}

}

//创建Dog子类并extends继承Animal父类及覆盖cry方法

class Dog extends Animal{

//狗叫

public void cry(){

System.out.println("汪汪叫");

}

//狗吃东西

public void eat(){

System.out.println("狗爱吃骨头");

}

}

class Cat extends Animal{

//猫自己叫

public void cry(){

System.out.println("猫猫叫");

}

//猫吃东西

public void eat(){

System.out.println("猫爱吃鱼");

}

}

**多态--注意事项：**

1、java允许父类的引用变量引用它的子类的实例(对象)

Animal an=new Cat();//这种转换时自动完成的

2、关于类型转换还有一些具体的细节要求，我们在后面还要提，比如子类能不能转换成父类，有什么要求等等...

-------------------------------------------------------------------------------

**java面向对象编程(3)--抽象类**

# 抽象类--解决之道

当父类的一些方法不能确定时，可以用**abstract**关键字来修饰该方法[抽象方法]，用**abstract**来修饰该类[抽象类]。

//抽象类的必要性[Demo124.java]

public class Demo124 {

public static void main(String[] args) {

**//Animal an=new Animal();抽象类不允许实例化**

Animal an=new Cat();

an.cry();

an=new Dog();

an.cry();

}

}

//抽象类abstract关键词

abstract class Animal{

String name;

int age;

//动物会叫，使用了abstract抽象方法

abstract public void cry();//抽象类中**可以没有abstract抽象方法**

//**抽象类内可以有实现方法**

public void sx(){

System.out.println("实现方法");

}

}

//**当一个子类继承的父类是abstract抽象类的话，需要程序员把抽象类的抽象方法全部实现。**

class Cat extends Animal{

//实现父类的cry，其实类似上节学习中的子类覆盖父类

public void cry(){

System.out.println("猫猫叫");

}

}

class Dog extends Animal{

//实现父类的cry，其实类似上节学习中的子类覆盖父类

public void cry(){

System.out.println("汪汪叫");

}

}

## 抽象类--深入讨论

抽象类是java中一个比较重要的类。

**1、用abstract关键字来修饰一个类时，这个类就是抽象类。**

**2、用abstract关键字来修饰一个方法时，这个方法就是抽象方法。**

**3、abstract抽象类中的abstract抽象方法是不允许在抽象类中实现的，一旦实现就不是抽象方法和抽象类了。abstract抽象方法只能在子类中实现。**

**4、抽象类中可以拥有实现方法。**

5、抽象方法在编程中用的不是很多，但是在公司笔试时，却是考官比较爱问的知识点。

**抽象类--注意事项**

**1、抽象类不能被实例化**

**2、抽象类不一定要包含abstract方法。也就是说，抽象类可以没有abstract抽象方法。**

**3、一旦类包含了abstract抽象方法，则这个类必须声明为abstract抽象类。**

**4、抽象方法不能有主体。**

**正确的抽象方法例：abstract void abc();**

**错语的抽象方法例：abstract void abc(){}**

## java面向对象编程(3)--接口

接口--为什么有？

USB插槽就是现实中的接口。

什么是接口？

接口就是给出一些没有内容的方法，封装到一起，到某个类要使用的时候，在根据具体情况把这些方法写出来。

**接口的建立语法：interface 接口名{方法;}**

**语法：class 类名 implements 接口{**

**方法;**

**变量;**

**}**

小结：接口是更加抽象的抽象的类，抽象类里的方法可以有方法体，接口里的所有方法都没有方法体。接口体现了程序设计的**多态和高内聚低偶合**的设计思想。

## 接口--注意事项

1、接口不能被实例化

2、接口中所有的方法都不能有主体。错误语法例：void aaa(){}←(注意不能有花括号)

接口可以看作更加抽象的抽象类。

3、一个类可以实现多个接口

4、接口中可以有变量[但变量不能用private和protected修饰]

a、接口中的变量，本质上都是static的而且是final类型的，不管你加不加static修饰

b、在java开发中，我们经常把常用的变量，定义在接口中，作为全局变量使用

**访问形式：接口名.变量名**

5、一个接口不能继承其它的类，但是可以继承别的接口

//接口的实现[Demo125.java]

//电脑，相机，u盘，手机

//usb接口

interface Usb{

int a=1;**//加不加static都是静态的，不能用private和protected修饰**

//声明了两个方法

public void start();//接口开始工作

public void stop();//接口停止工作

}

//编写了一个相机类，并实现了usb接口

//一个**重要的原则：当一个类实现了一个接口，要求该类把这个接口的所有方法全部实现**

class Camera implements Usb{

public void start(){

System.out.println("我是相机，开始工作了..");

}

public void stop(){

System.out.println("我是相机，停止工作了..");

}

}

//接口继承别的接口

class Base{

}

interface Tt{

}

interface Son extends Tt{

}

//编写了一个手机，并实现了usb接口

class Phone implements Usb{

public void start(){

System.out.println("我是手机，开始工作了..");

}

public void stop(){

System.out.println("我是手机，停止工作了..");

}

}

//计算机

class Computer{

//开始使用usb接口

public void useUsb(Usb usb){

usb.start();

usb.stop();

}

}

public class Demo125 {

public static void main(String[] args) {

System.out.println(Usb.a);

//创建 Computer

Computer computer=new Computer();

//创建Camera

Camera camera1=new Camera();

//创建Phone

Phone phone1=new Phone();

computer.useUsb(camera1);

computer.useUsb(phone1);

}

}

-------------------------------------------------------------------------------

定义类的改进

有了接口，我们类的定义就更加完善了：

class 类名{ class 类名{ class类名{ package包名;

成员变量; → 成员变量; → 成员变量; → class 类名{

} 成员方法; 构造方法; 成员变量;

} 成员方法; 构造方法;

} 成员方法;

} ↓

↓←←←←←←←←←←←←←←←←←←←←←←←←←

package 包名; package 包名;

class 类名 extends 父类{ class 类名 extends 父类 implements 接口名{

成员变量; → 成员变量;

构造方法; 构造方法;

成员方法; 成员方法;

} }

-------------------------------------------------------------------------------

## 实现接口VS继承类

java的**继承是单继承，也就是一个类最多只能有一个父类**，这种单继承的机制可保证类的纯洁性，比C++中的多继承机制简洁。但是不可否认，对子类功能的扩展有一定影响。所以：

1、**实现接口可以看作是对继承的一种补充。**(继承是层级式的，不太灵活。修改某个类就会打破继承的平衡，而接口就没有这样的麻烦，因为它只针对实现接口的类才起作用)

2、**实现接口可在不打破继承关系的前提下，对某个类功能扩展，非常灵活。**

//实例：建立子类并继承了父类且连接多个接口[Demo126.java]

public class Demo126 {

public static void main(String[] args) {

System.out.println("继承了Monkey父类");

Monkey mo=new Monkey();

mo.jump();

LittleMonkey li=new LittleMonkey();

li.swimming();

li.fly();

}

}

//接口Fish

interface Fish{

public void swimming();

}

//接口Bird

interface Bird{

public void fly();

}

//建立Monkey类

class Monkey{

int name;

//猴子可以跳

public void jump(){

System.out.println("猴子会跳!");

}

}

//建立LittleMonkey子类并继承了Monkey父类并连接了Fish和Bird接口

class LittleMonkey extends Monkey implements Fish,Bird{

public void swimming() {

System.out.println("连接了Fish接口!");

}

public void fly() {

System.out.println("连接了Bird接口!");

}

}

-------------------------------------------------------------------------------

## **用接口实现多态**

--案例[Demo127.java]

java中多态是个难以理解的概念，但同时又是一个非常重要的概念。java三大特性之一(继承，封装，多态)，我们可以从字面上简单理解：就是一种类型的多种状态，以下通过卖小汽车的例子说明什么是多态。

//用接口实现多态

public class Demo127 {

public static void main(String []args){

CarShop aShop=new CarShop();

//卖出一辆宝马

aShop.sellCar(new BMW());

//卖出一辆奇瑞QQ

aShop.sellCar(new CheryQQ());

//卖出一辆桑塔纳

aShop.sellCar(new Santana());

System.out.println("总收入："+aShop.getMoney());

}

}

//汽车接口

interface Car{

//汽车名称

String getName();

//获得汽车售价

int getPrice();

}

//宝马

class BMW implements Car{

public String getName(){

return "BMW";

}

public int getPrice(){

return 300000;

}

}

//奇瑞QQ

class CheryQQ implements Car{

public String getName(){

return "CheryQQ";

}

public int getPrice(){

return 20000;

}

}

//桑塔纳汽车

class Santana implements Car{

public String getName(){

return "Santana";

}

public int getPrice(){

return 80000;

}

}

//汽车出售店

class CarShop{

//售车收入

private int money=0;

//卖出一部车

public void sellCar(Car car){

System.out.println("车型："+car.getName()+"单价："+car.getPrice

());

//增加卖出车售价的收入

money+=car.getPrice();

}

//售车总收入

public int getMoney(){

return money;

}

}

运行结果：

车型：BMW 单价：300000

车型：CheryQQ 单价：20000

总收入：320000

继承是多态得以实现的基础。从字面上理解，多态就是一种类型(都是Car类型)表现出多种状态(宝马汽车的名称是BMW，售价是300000；奇瑞汽车的名称是CheryQQ，售价是2000)。将一个方法调用同这个方法所属的主体(也就是对象或类)关联起来叫做绑定，分前期绑这下和后期绑定两种。下面解释一下它们的定义：

**1、前期绑定：在程序运行之前进行绑定，由编译器和连接程序实现，又叫做静态绑定。比如static方法和final方法，注意，这里也包括private方法，因为它是隐式final的。**

**2、后期绑定：在运行时根据对象的类型进行绑定，由方法调用机制实现，因此又叫做动态绑定，或者运行时绑定。除了前期绑定外的所有方法都属于后期绑定。**

**多态就是在后期绑定这种机制上实现的。**多态给我们带来的好处是消除了类之间的偶合关系，使程序更容易扩展。比如在上例中，新增加一种类型汽车的销售。只需要让新定义的类实现Car类并实现它的所有方法，而无需对原有代码做任何修改，CarShop类的sellCar(Car

car)方法就可以处理新的车型了。

新增代码如下：

java代码

//桑塔纳汽车

class Santana implements Car{

public String getName(){

return "Santana";

}

public int getPrice(){

return 80000;

}

}

-------------------------------------------------------------------------------

# java面向对象编程(3)--final

final概念

final中文意思：最后的，最终的。

final可以修饰变量或者方法。

在某些情况下，程序员可能有以下需求：

**1、当不希望父类的某个方法被子类覆盖(override)时，可以用final关键字修饰。**

**2、当不希望类的某个变量的值被修改，可以用final修饰。如果一个变量是final，则必须赋初值，否则编译出错。**

**3、当不希望类被继承时，可以用final修饰。**

//final方法的使用[Demo128.java]

public class Demo128 {

public static void main(String[] args) {

Aaa aaa=new Aaa();

aaa.show();

Bbb bbb=new Bbb();

bbb.show();

}

}

class Aaa{

int a=0;//如果a不赋初值,a是0。**定义类型后应赋值**

//圆周率不让修改

//带有**final修饰的变量命名时应有\_下划线来区分表示**。这是java程序员的标准。

final float reate\_1=3.1415926f;//**使用final可以保证，需要强制不被修改的数据一定要用final锁定**

//final int b; **//使用final定义变量时一定要赋初值否则报错。**

//b=1;

final public void sendMes(){**//给成员方法用final来修饰则表示不可以被修改，不可被覆盖。**

System.out.println("发送消息");

}

public void show(){

System.out.println("a="+a);

}

}

final class Bbb extends Aaa{**//定义类前加final表示该类不允许被继承**

public Bbb(){

a++;

//reate\_1=reate+1;

}

/\*public void sendMes(){

System.out.println("发送消息")

}\*/

}

## final--注意事项

1、final修饰的变量又叫常量，一般用XX\_XX\_XX来命名。(带下划线)

2、final修饰的变量在定义时，必须赋值，并且以后不能再赋值。

## final--什么时候用

1、因为案例的考虑，类的某个方法不允许修改。

2、类不会被其它的类继承。

3、某些变量值是固定不变的，比如圆周率3.1415926

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**作业：**

第一题：目的

1、变量定义、初始化、赋值

2、运算符和表达式

3、输出语句System.out.println();

1.作一个加法器，计算它们的和

2.作一个减法器，计算它们的差

3.作一个乘法器，计算它们的积

4.作一个除法器，计算它们的商(和余数)

[Work01.java]

public class Work01{

public static void main(String []args){

int i=9;int j=4;

int k=i+j;

int l=i-j;

int m=i\*j;

int n=i/j;

int o=i%j;

System.out.println("和："+k+" 差："+l+" 积:"+m+" 商:"+n+" 模："+o);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第二题：正方型[Work02.java]

//正方形

public class Work02{

public static void main(String []args){

for(int i=1;i<5;i++){

System.out.println("\*\*\*\*\*\*\*\*");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第三题：直角三角形[Work03.java]

//直角三角形

public class Work03{

public static void main(String []args){

for(int i=1;i<9;i++){

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

System.out.print("\*");

}

System.out.println();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第四题：三角形[Work04.java]

//三角形

public class Work04{

public static void main(String []args){

int lay=10;//行数

for(int i=1;i<=lay;i++){//循环行

for(int k=1;k<=lay-i;k++){//找空格

System.out.print(" ");

}

for(int j=1;j<=(i-1)\*2+1;j++){//找星位置并输入

System.out.print("\*");

}

System.out.println();//换行

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第五题：

已知手掌游戏机一台320元，MP3一台260元。公司举行抽奖大会，公司有18人，一等奖游戏机，二等奖MP3，购入5公游戏机，13台MP3。问总价为多少，公司实际平均给每人花费多少钱。[Work05.java]

//计算总费用和平均费用

import java.util.\*;//加载包

public class Work05{

public static int gbGame=320;//静态变量

public static int gbGameT=5;

public static int mp3=260;

public static int mp3T=13;

public static int people=18;

public static void main(String []args){

Scanner myScanner=new Scanner(System.in);//调用键盘输入

showInfo(); //调用showInfo方法

System.out.print("请输入手掌游戏机价格：");

gbGame=myScanner.nextInt();

System.out.print("请输入MP3价格：");

mp3=myScanner.nextInt();

System.out.print("请输入手掌游戏机数量：");

gbGameT=myScanner.nextInt();

System.out.print("请输入MP3数量：");

mp3T=myScanner.nextInt();

showInfo();

}

public static void showInfo(){ // 建立showInfo方法

int total=gbGame\*gbGameT+mp3\*mp3T;//总价

float average=(float)total/people;//平均价

System.out.printf("总价： %d,平均价是: %f \n",total,average);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第六题

手掌游戏机一台X元，mp3一台Y元(X,Y,N,M,Z由用户输入)

公司举行抽奖大会，一等奖游戏机N台，二等奖mp3 M台。公司共有Z人。问总价为多少，公司实际平均给每人花费了多少钱。

//计算总费用和平均费用

import java.util.\*;//加载包

public class Work06{

public static int gbGame=0;//静态变量

public static int gbGameT=0;

public static int mp3=0;

public static int mp3T=0;

public static int people=0;

public static void main(String []args){

Scanner myScanner=new Scanner(System.in);//调用键盘输入

System.out.print("请输入手掌游戏机价格：");

gbGame=myScanner.nextInt();

System.out.print("请输入MP3价格：");

mp3=myScanner.nextInt();

System.out.print("请输入手掌游戏机数量：");

gbGameT=myScanner.nextInt();

System.out.print("请输入MP3数量：");

mp3T=myScanner.nextInt();

System.out.print("请输入员工数：");

people=myScanner.nextInt();

showInfo();//调用showInfo方法

}

public static void showInfo(){ // 建立showInfo方法

int total=gbGame\*gbGameT+mp3\*mp3T;//总价

float average=(float)total/people;//平均价

System.out.printf("总价： %d,平均价是: %f \n",total,average);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第七题

九九乘法表，要求行列对齐[Work07.java]

//九九乘法表

import java.util.\*;//加载包

public class Work07{

public static void main(String []args){

Scanner myScanner=new Scanner(System.in);//调用键盘输入

System.out.print("请输入1-9中的一个数：");

int num=myScanner.nextInt();

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

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

System.out.print(i+"×"+j+"="+(i\*j)+"\t");

}

System.out.println();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第八题

请使用三个整型变量完成如下功能：

1、用户输入两个整数值；3 5

2、计算两数的和，输出此和； 8

3、求出两数的平均数，输出平均数(整数即可)； 4

4、求出平均数>第二个数的关系表达式的值，输出此值； 0

5、用第一个数减去上面关系表达式的值，得到它们的差，输出差； 3

6、存储第二个数的变量累乘上式的结果，输出此变量的当前值； 15

7、存储第一个数的变量累加第二个变量，输出第一个变量的当前值。 18

//输入两个数完成任务集合[Work08.java]

import java.util.\*;//加载包

public class Work08{

public static void main(String []args){

Scanner myScanner=new Scanner(System.in);//调用键盘输入

System.out.print("请输入一个数A：");

int a=myScanner.nextInt();

System.out.print("请输入一个数B：");

int b=myScanner.nextInt();

int e=0;//定义变量

//求和

int c=a+b;

System.out.println("和："+c);

//求平均数

int d=(a+b)/2;

System.out.println("平均数："+d);

//判断平均数是否大于第二个数

if(d>b){

e=1;

System.out.println("平均数>第二个数");

}else{

e=0;

System.out.println("小于"+b+"输出判断值"+e);

}

//第一个数减去上式的值

int f=a-e;

System.out.println("第一个数减去上式的值"+f);

//第二个数乘上式的值

int g=b\*f;

System.out.println("第二个数累乘上式的值"+g);

//第一个数累加并赋给第一个数

a+=g;

System.out.println("第一个数累加第二个数赋给第一个数的值"+a);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第九题

渔夫出海打鱼，收获若干

1、渔夫卖掉一半的鱼，然后送给海伦3条；

2、渔夫又卖掉剩下的鱼的2/3，自己吃掉1条；

3、海伦来看渔夫，问他那天打了多少鱼，渔夫数了数，还剩4条鱼，渔夫对海伦怎么说？

//打鱼--共有多少鱼[Work09.java]

public class Work09{

public static void main(String []args){

int total=0;

int rest=4;//剩余的鱼

total=((rest+1)\*3+3)\*2;//逆推

System.out.println("鱼一共有："+total);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十题

有一只猴子摘了一堆桃子，当即吃了一半，可是桃子太好吃了，它又多吃了一个，第二天它把第一天剩下的桃子吃了一半，又多吃了一个，就这样到第十天早上它只剩下一个桃子了，问它共摘了多少桃子[Work10.java]

import java.util.\*;//加载包

public class Work10{

public static void main(String []args){

Monkey mo=new Monkey();

mo.scan();

}

}

class Monkey{

//day哪天的桃子数，sday吃了多少天剩下一个

public static int peach(int day,int sday){//建一个peach方法

if(day==sday){

return 1;

}else{

return (peach(day+1,sday)+1)\*2;//算法返回值

}

}

public static void scan(){ //建立输入的方法

int a=1;

System.out.println("请输入要吃几天：");

Scanner sr=new Scanner(System.in);

int b=sr.nextInt();

if(a<b){

System.out.print(Monkey.peach(a,b));

}else{

System.out.println("不能小于等于1天");

Monkey.scan();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十一题

计算1+2+3+...+20=?思考如果让用户输入一个整数n计算累加和：1+2+3+...+n能做到吗？

[Work11.java]

import java.util.\*;//加载包

public class Work11{

public static void main(String []args){

Scanner myScanner=new Scanner(System.in);//调用键盘输入

System.out.print("请输入一个数N：");

int n=myScanner.nextInt();

int i=0;

for(int m=0;m<n;m++){

i=i+(m+1);

System.out.println("m是"+m);

}

System.out.println("1+2+3+...+n的值是"+i);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十二题

请编写一个Cat类，要求如下：

该猫可以做四则运算，也可以进行面积计算；

将四则运算器和面积运算器合二为一，作一个运算器，主菜单让用户选择是做四则运算还是面积运算，分为两个子菜单，让用户选择加减乘除或者形状。

//一个Cat类进行四则运算和面积计算[Work12.java]

//一个Cat类进行四则运算和面积计算

import java.util.\*;//加载包

public class Work12{

public static void main(String []args){

Cat cat=new Cat();

cat.scan1();

}

}

class Cat{

//输入方法

public static void scan1(){

System.out.println("题目类型如下");

System.out.println("1、四则运算");

System.out.println("2、面积运算");

System.out.println("0、退出程序");

System.out.print("请选择要做题的类型，请入0-2：");

Scanner sr=new Scanner(System.in);

int x=sr.nextInt();

switch(x){

case 0:

System.out.println("谢谢使用，再见!");//退出

break;

case 1:

Cat.szys();

break;

case 2:

Cat.mjys();

break;

default:

System.out.println("输入有误，请重新输入!");

Cat.scan1();//重复调用目录

}

}

//四则运算方法

public static void szys(){

System.out.println("四则运算");

System.out.println("1、加法运算");

System.out.println("2、减法运算");

System.out.println("3、乘法程序");

System.out.println("4、除法程序");

System.out.println("0、返回上级");

System.out.print("请选择要做题的类型，请入0-4：");

Scanner sr=new Scanner(System.in);

int y=sr.nextInt();

switch(y){

case 0:

System.out.println("返回上级");

Cat.scan1();//调用主目录

break;

case 1:

System.out.println("进入加法运算，请输入第一个数：");

Scanner jf1=new Scanner(System.in);

int jf01=jf1.nextInt();

System.out.println("进入加法运算，请输入第二个数：");

Scanner jf2=new Scanner(System.in);

int jf02=jf2.nextInt();

float jfh=(float)jf01+(float)jf02;

System.out.println(jf01+"+"+jf02+"="+jfh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner cz01=new Scanner(System.in);

int cz001=cz01.nextInt();

if(cz001==1){

Cat.szys();

}else if(cz001==2){

Cat.scan1();

}else if(cz001!=1||cz001!=2){

System.out.println("谢谢使用，再见!");

}

break;

case 2:

System.out.println("进入减法运算，请输入第一个数：");

Scanner jjf1=new Scanner(System.in);

int jjf01=jjf1.nextInt();

System.out.println("进入减法运算，请输入第二个数：");

Scanner jjf2=new Scanner(System.in);

int jjf02=jjf2.nextInt();

float jjfh=(float)jjf01-(float)jjf02;

System.out.println(jjf01+"-"+jjf02+"="+jjfh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner cz02=new Scanner(System.in);

int cz002=cz02.nextInt();

if(cz002==1){

Cat.szys();

}else if(cz002==2){

Cat.scan1();

}else if(cz002!=1||cz002!=2){

System.out.println("谢谢使用，再见!");

}

break;

case 3:

System.out.println("进入乘法运算，请输入第一个数：");

Scanner cff1=new Scanner(System.in);

int cff01=cff1.nextInt();

System.out.println("进入乘法运算，请输入第二个数：");

Scanner cff2=new Scanner(System.in);

int cff02=cff2.nextInt();

float cffh=(float)cff01\*(float)cff02;

System.out.println(cff01+"×"+cff02+"="+cffh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner cz03=new Scanner(System.in);

int cz003=cz03.nextInt();

if(cz003==1){

Cat.szys();

}else if(cz003==2){

Cat.scan1();

}else if(cz003!=1||cz003!=2){

System.out.println("谢谢使用，再见!");

}

break;

case 4:

System.out.println("进入除法运算，请输入第一个数：");

Scanner cf1=new Scanner(System.in);

int cf01=cf1.nextInt();

System.out.println("进入除法运算，请输入第二个数：");

Scanner cf2=new Scanner(System.in);

int cf02=cf2.nextInt();

float cfh=(float)cf01/(float)cf02;

System.out.println(cf01+"÷"+cf02+"="+cfh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner cf03=new Scanner(System.in);

int cf003=cf03.nextInt();

if(cf003==1){

Cat.szys();

}else if(cf003==2){

Cat.scan1();

}else if(cf003!=1||cf003!=2){

System.out.println("谢谢使用，再见!");

}

break;

default:

System.out.println("输入有误，请重新输入!");

Cat.szys();//循环调用子目录

}

}

//面积运算方法

public static void mjys(){

System.out.println("面积运算");

System.out.println("1、三角形面积计算");

System.out.println("2、圆形面积计算");

System.out.println("3、方形面积计算");

System.out.println("0、返回上级");

System.out.print("请选择要做题的类型，请入0-3：");

Scanner sr=new Scanner(System.in);

int z=sr.nextInt();

switch(z){

case 0:

System.out.println("返回上级");

Cat.scan1();//调用主目录

break;

case 1:

System.out.println("进入三角形面积计算，请输入底：");

Scanner sj1=new Scanner(System.in);

int sjx01=sj1.nextInt();

System.out.println("进入三角形面积计算，请输入高：");

Scanner sj2=new Scanner(System.in);

int sjx02=sj2.nextInt();

float sjxh=((float)sjx01\*(float)sjx02)/2;

System.out.println("底"+sjx01+" 高："+sjx02+" 面积="+sjxh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner sj03=new Scanner(System.in);

int sj003=sj03.nextInt();

if(sj003==1){

Cat.mjys();

}else if(sj003==2){

Cat.scan1();

}else if(sj003!=1||sj003!=2){

System.out.println("谢谢使用，再见!");

}

break;

case 2:

System.out.println("进入圆形面积计算，请输入半径：");

Scanner y1=new Scanner(System.in);

int bj=y1.nextInt();

float ymjh=(float)bj\*(float)bj\*3.1415926f;

System.out.println("半径:"+bj+" 面积:"+ymjh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner y01=new Scanner(System.in);

int y001=y01.nextInt();

if(y001==1){

Cat.mjys();

}else if(y001==2){

Cat.scan1();

}else if(y001!=1||y001!=2){

System.out.println("谢谢使用，再见!");

}

break;

case 3:

System.out.println("进入方形面积计算，请输入长：");

Scanner c1=new Scanner(System.in);

int c01=c1.nextInt();

System.out.println("进入方形面积计算，请输入宽：");

Scanner k2=new Scanner(System.in);

int k02=k2.nextInt();

float mjh=(float)c01\*(float)k02;

System.out.println("长:"+c01+" 宽："+k02+" 面积:"+mjh);

System.out.println("已计算完成。");

System.out.println("是否要继续使用？使用请输入1，返回请输入2，退出请输入任意数字:");

Scanner m03=new Scanner(System.in);

int m003=m03.nextInt();

if(m003==1){

Cat.mjys();

}else if(m003==2){

Cat.scan1();

}else if(m003!=1||m003!=2){

System.out.println("谢谢使用，再见!");

}

break;

default:

System.out.println("输入有误，请重新输入!");

Cat.mjys();//循环调用子目录

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十三题

请大家采用面向对象思想设计超级马里奥游戏人物。并能用键盘输入命令，来控制马里奥的位置[Work13.java]

//超级马里奥人物建造及控制台数字操作

import java.util.\*;

public class Work13 {

public static void main(String[] args) {

Scanner sr=new Scanner(System.in);//接收键盘输入

System.out.println("输入1-25的数字，你可以移动马里奥，输入其它数字退出");

int i=sr.nextInt();//将键盘输入数字赋值给i

Maria m=new Maria();//建立调用Maria构造方法

for(int a=i;a<=25;a--){

if(i<25){

m.move(i);

i=sr.nextInt();

}else{

System.out.println("退出游戏!");

break;

}

}

}

}

class Maria{

int x,y;

//构造方法，初始化场景；确定地图，Maria的位置

public Maria(){

System.out.println("I'm Maria \n"+"@");

}

public void move(int i){

for(int k=1;k<=25;k++){

if(k==i){

System.out.printf("@");

}else{

System.out.printf(".");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**代码分析：**

1、运行一下代码，将得到什么打印结果：

int i=3;

int j=0;

double k=3.2;

if(j<k)

if(i==j)

System.out.println(i);

else

System.out.println(j);//执行此语句，得到结果0

else

System.out.println(k);

-------------------------------------------------------------------------------

2、以下代码能否编译通过？假如能编译通过，运行时得到什么打印结果？

int i=4;

switch(i){

default;

System.out.println("default");//执行此语句

case 0:

System.out.println("zero");//执行此语句

break;

case 1:

System.out.println("one");

case 2:

System.out.println("two");

}

-------------------------------------------------------------------------------

3、以下那些代码是合法的？

a)

int i;

for(i=5,int j=10;i<10;j--){}//错

b)

int i,j;

for(i=0,j=10;i<10,j>0;i++,j--){}//错

c)

int i,k;

for(i=0,k=9;(i<10&&k>0);i++,j--){}//错

d)

int i,j

for(i=0;j=10;i<10;i++,j--){}//错

-------------------------------------------------------------------------------

4、运行以下代码，将得到什么打印结果？

int i=1;

switch(i){

default:

System.out.println("default");

case 0:

System.out.println("zero");

break;

case 1;

System.out.println("one");//执行此语句

case2;

System.out.println("two");//执行此语句

}

-------------------------------------------------------------------------------

5、以下代码是合法的？

a)

float x=1;//数据类型错误

switch(x){

case 1:

System.out.println("Got a 1");//错

}

b)

long y=1;//数据类型错误

switch(y){

case 1:

System.out.println("Got a 1");//错

}

c)

byte x=1;//低精转高精

switch(x){

case 1/1:

System.out.println("Got a 1");//正确

}

d)

int x=1;

int c=1;

switch(c){

case x;//需常量

System.out.println("Got a 1");//错

break;

e)

short x=1;//数据类型错误

switch(x){

case 3.2/3:

System.out.println("Got a 1");

break;

}

f)

shout x=1;//数据类型错误

switch(x){

case 1,2,3:

System.out.println("Got a 1");//错

break;

}

-------------------------------------------------------------------------------

6、以下代码能否编译通过？假如能编译通过，运行时将得到什么打印结果？

public class Myswitch{

public static void main(String args[]){

Myswitch ms=new Myswitch();

ms.amethod();

}

public void amethod(){

for(int a=0,b=0;a<2;b=++a,System.out.println("b="+b)){

System.out.println("a="+a);

}

}

}

结果：a=0

b=1

a=1

b=2

-------------------------------------------------------------------------------

7、以下代码能否编译通过？假如能编译通过，运行时将得到什么打印结果？

void looper(){

int x=0;

one:

while(x<10){

two:

System.out.println(++x);

if(x>3)

break two;

}

}//无法编译，break two;无法返回到two标签

-------------------------------------------------------------------------------

8、以下代码能否编译通过？假如能编译通过，运行时将得到什么打印结果？

public class Hope{

public static void main(String args[]){

Hope h=new Hope();

}

protected Hope(){

int i=1;

do{

System.out.println(i);

}while(++i<3);

}

}

结果：1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十四题

请编写一个猜拳的游戏

有个人Tom设计他的成员变量，成员方法，可以电脑猜拳。电脑每次都会随机生成0,1,2；0表示石头、1表示剪刀、2表示布

//与电脑猜拳[Work14\_1.java]

import java.util.\*;

public class Work14\_1 {

public static void main(String[] args) {

Kais.jingcyx();//调用开始游戏

}

}

//开始退出游戏类

class Kais{

public static void jingcyx(){

System.out.println("是否开始猜拳游戏？按1开始，按0退出");

Scanner sr=new Scanner(System.in);

int x=sr.nextInt();

switch (x){

case 0:

System.out.println("已退出游戏!");

break;

case 1:

System.out.println("游戏说明：0代表石头，1代表剪刀，2代表布");

Caiquan.caiquan();

break;

default:

System.out.println("输入错误，请重新输入!");

Kais.jingcyx();

break;

}

}

}

//猜拳游戏

class Caiquan{

public static void caiquan(){

int a1=0,b1=0,c1=0;

System.out.print("请输入想玩的盘数：");

Scanner p=new Scanner(System.in);

int ps=p.nextInt();

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

System.out.println("游戏已开始，请出拳!");

Scanner m=new Scanner(System.in);

int a=m.nextInt();//玩家出拳

//Random n=new Random(2);//随机函数

int b=(int)(0+Math.random()\*(3-1+1));//随机产生0-2的int数，电脑出拳

if(a==b){

a1++;

System.out.println("打平!");

}else if((a-b)==-1||(a-b)==2){

b1++;

System.out.println("YOU Win!");

}else{

c1++;

System.out.println("YOU LOSE");

}

}

System.out.println("平:"+ a1 +"\n赢:" + b1 +"\n输:"+c1+"\n一共玩了"+(a1+b1+c1)+"局");

Kais.jingcyx();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十五题

请编写一个类koradji(巫师的意思)

为类编写如下功能：

1、根据用户的输入某人的生日，可以计算出该人的星座；

2、根据用户的输入年月日可以计算是星期几；

3、可以显示该巫师给多少人算过星相。

1月20日- 2月18日水瓶 2月19日- 3月20日双鱼 3月21日- 4月19日白羊

4月20日- 5月20日金牛 5月21日- 6月20日双子 6月21日- 7月22日巨蟹

7月23日- 8月22日狮子 8月23日- 9月22日处女 9月23日-10月22日天秤

10月23日-11月21日天蝎 11月22日-12月21日射手 12月22日- 1月19日摩羯

//通过输入生日显示出生在周几。巫师看星座[Wrok15.java]

import java.util.\*;

public class Work15{

public static void main(String []args){ //程序入口

Start ks=new Start();//调用程序开始类Start，并启用st方法

ks.st();

}

}

class Start{//程序开始类

public void st(){

System.out.println("年月日格式：19XX XX XX\n请输入出生年月日：");

Scanner sr=new Scanner(System.in);

int year=sr.nextInt();//输入年

int month=sr.nextInt();//输入月

int date=sr.nextInt();//输入日

Rq r=new Rq(); //调用周几类

r.rq(year,month,date);//将年月日数据传给周几类的rq方法中

Xinz xz=new Xinz();//调用星座类

xz.xinz(month,date);//将月日的数据传给星座类中的xinz方法中

System.out.println("\n是否继续让巫师看星座？\n1、继续\t2、退出");

Scanner sr1=new Scanner(System.in);

int o=sr1.nextInt();

if(o==1){

Start qd=new Start();//输入1调用程序开始类

qd.st();

}else if(o!=1){ //不等于1退出

Xinz tc=new Xinz();

System.out.println("巫师为"+tc.getI()+"人看过星座");

System.out.println("退出星座查询，Goodbay!");

}

}

}

class Rq{//周几类

public void rq(int year,int month,int date){

Calendar c = Calendar.getInstance(); //调用日期Calendar抽象类

c.set(Calendar.YEAR,year);

c.set(Calendar.MONTH,month-1);

c.set(Calendar.DATE,date);

int week = c.get(Calendar.DAY\_OF\_WEEK);

switch (week){ //判断输入的日期是周几

case 1:

System.out.println("星期日");

break;

case 2:

System.out.println("星期一");

break;

case 3:

System.out.println("星期二");

break;

case 4:

System.out.println("星期三");

break;

case 5:

System.out.println("星期四");

break;

case 6:

System.out.println("星期五");

break;

case 7:

System.out.println("星期六");

break;

}

}

}

class Xinz{ //星座类

public static int i=0;//定义静态变量i，i为统计调用人数计数器

public void xinz(int month,int date){ //判断星座的方法

i++;//调用此方法计数累器

switch(month){

case 1:{

if(date>=20){

System.out.println("水瓶座");

}else{

System.out.println("摩羯座");

}

break;

}

case 2:{

if(date>=19){

System.out.println("双鱼座");

}else{

System.out.println("水瓶座");

}

break;

}

case 3:{

if(date>=21){

System.out.println("白羊座");

}else{

System.out.println("双鱼座");

}

break;

}

case 4:{

if(date>=20){

System.out.println("金牛座");

}else{

System.out.println("白羊座");

}

break;

}

case 5:{

if(date>=21){

System.out.println("双子座");

}else{

System.out.println("金牛座");

}

break;

}

case 6:{

if(date>=21){

System.out.println("巨蟹座");

}else{

System.out.println("金牛座");

}

break;

}

case 7:{

if(date>=23){

System.out.println("狮子座");

}else{

System.out.println("巨蟹座");

}

break;

}

case 8:{

if(date>=20){

System.out.println("处女座");

}else{

System.out.println("巨蟹座");

}

break;

}

case 9:{

if(date>=23){

System.out.println("天平座");

}else{

System.out.println("处女座");

}

break;

}

case 10:{

if(date>=23){

System.out.println("天蝎座");

}else{

System.out.println("天平座");

}

break;

}

case 11:{

if(date>=22){

System.out.println("射手座");

}else{

System.out.println("天蝎座");

}

break;

}

case 12:{

if(date>=22){

System.out.println("摩羯座");

}else{

System.out.println("射手座");

}

break;

}

}

}

public int getI(){ //返回统计调用次数

return i;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第十六题

10个人投票选班长，有三个侯选人(张三，李四，王五)，通过编号投票，要求最后显示班长姓名。[Work16.java]

import java.util.\*;

public class Work16 {

public static void main(String[] args) {

//设定投票人数

System.out.print("设定投票人数：");

Scanner sr=new Scanner(System.in);

int a=sr.nextInt();

System.out.print("请输入第1位侯选人的名字：\n");

Scanner sr1=new Scanner(System.in);

String i1=sr1.nextLine();//键盘接收字符串

System.out.print("请输入第2位侯选人的名字：\n");

Scanner sr2=new Scanner(System.in);

String i2=sr2.nextLine();

System.out.print("请输入第3位侯选人的名字：\n");

Scanner sr3=new Scanner(System.in);

String i3=sr3.nextLine();

System.out.println("本次选举将有"+ a +"人参与!");

System.out.println("1号:"+i1+"\t2号:"+i2+"\t3号:"+i3);

Xuanju tp=new Xuanju();

tp.tp(a,i1,i2,i3);

}

}

class Xuanju{

public void tp(int a,String i1,String i2,String i3){

int j1=0,j2=0,j3=0,qq=0;//计票器

for (int i=1;i<=a;i++){ //统计投票

System.out.println("请投票，投票请按1,2,3：");

Scanner tp=new Scanner(System.in);

int b=tp.nextInt();

System.out.println("已有"+ i +"人投票");

switch (b){

case 1:

j1++;

break;

case 2:

j2++;

break;

case 3:

j3++;

break;

default:

qq++;

break;

}

}

if(j1>j2&&j1>j3){

System.out.println(i1 +"得"+ j1 +"票\n"+ i2 +"得"+ j2 +"票\n"+ i3 +"得"+ j3 +"票\n"+"弃权票数"+qq);

System.out.println("恭喜" + i1 + "当选班长!");

}else if(j1<j2&&j2>j3){

System.out.println(i1 +"得"+ j1 +"票\n"+ i2 +"得"+ j2 +"票\n"+ i3 +"得"+ j3 +"票\n"+"弃权票数"+qq);

System.out.println("恭喜" + i2 + "当选班长!");

}else if(j3>j1&&j2<j3){

System.out.println(i1 +"得"+ j1 +"票\n"+ i2 +"得"+ j2 +"票\n"+ i3 +"得"+ j3 +"票\n"+"弃权票数"+qq);

System.out.println("恭喜" + i3 + "当选班长!");

}else {

System.out.println(i1 +"得"+ j1 +"票\n"+ i2 +"得"+ j2 +"票\n"+ i3 +"得"+ j3 +"票\n"+"弃权票数"+qq);

System.out.println("有同等票数，投票无效重新投票!");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**一维数组**

什么是数组？

数组可以存放多个同一类型数据。

数组

一、关于数组的用法，有三种方式：1、**程序员用法**

**1、数组的定义**

数据类型 数组名[]=new 数据类型[数组大小];

int a[]=new int[5];//定义一个数组名为a的整型数组，并可以放入5个整型数。

说明：这是定义数组的一种方法。

a｜a[0]｜a[1]｜a[2]｜a[3]｜a[4]｜电脑中的数组

3楼｜301房｜302房｜303房｜304房｜305房｜现实中的楼房

楼是存在地球上的，那么数组是存在哪里的呢？-->电脑内存中

**2、数组的引用(使用)**

数组名[下标]

比如：你要使用a数组的第三个数a[2]

二、关于数组的用法，有几种方式：2、**没事找事用法**

第一步：**先声明数组**

语法：数据类型 数组名[]; 也可以 数据类型[] 数组名;

例：int a[];或者int[] a;

第二步：**创建数组**

语法：数组名=new 数据类型[数组大小];

例：a=new int[10];

第三步：**数组的引用(使用)**

语法：数组名[下标]

例：引用a数组的第8个元素 a[7]

要想知道数组的大小可以使用数组的length方法

**语法：数组名.length**

三、关于数组的用法，有几种方式：**3、古板用法** (当已知元素值的时候可以使用此法)

1、**初始化数组**

语法：数据类型 数组名[]={元素值,元素值...};

例：int a[]={2,5,6,7,8,89,90,34,56}

上面的用法相当于：

int a[]=new int[9]

int a[0]=2;int a[1]=5;int a[2]=6;...a[8]=56;

2、**数组的引用(使用)**

语法：数组名[下标]

例：a数组的第8个元素a[7]

一个问题？

一个养鸡场有6只鸡，它们的体重分别是3kg、5kg、1kg，3.4kg、2kg、50kg。请问这六只鸡的总体重是多少？平均体重是多少？请你编写一个程序。

//数组的必要性[Demo129.java]

public class Demo129 {

public static void main(String[] args) {

//定义一个可以存放六个float类型的数组

float arr[]=new float[6];

//使用for循环赋值

//给数组的各个元素赋值

arr[0]=3;

arr[1]=5;

arr[2]=1;

arr[3]=3.4f;

arr[4]=2;

arr[5]=50;

//计算总体重[遍历数组]

float all=0;

for(int i=0;i<6;i++){

all+=arr[i];

}

System.out.println("总体重是："+all);

}

}

-------------------------------------------------------------------------------

数组--一维数组

在运动会上，五个小孩比赛滑轮，他们划完100米用时，分别用了10s/12s/5.7s/9s/14s，请编写一个程序，计算他们的平均时间？

public class Demo130{

public static void main(String []args){

//使用古板法定义数组并给数组赋值

float time[]={10,12,5.7f,9,14};

float zs=0;

for(int i=0;i<time.length;i++){

zs+=time[i];

}

System.out.println("百米平均用时："+(zs/time.length)+"s");

//如何知道数组的大小？使用数组的length方法便可知道数组大小

System.out.println(time.length);

}

}

-------------------------------------------------------------------------------

数组

请思考，既然int,float,double..等都可以有数组，那么可不可以有对象数组呢？

对前面的养鸡场的题，进行升级，题要求如下：

一个养狗场有4只狗，分别是：

名字 体重

花花 4.5kg

白白 5.6kg

黑黑 78kg

红红 9.0kg

请编写一个程序，可以计算他们的平均体重，可以找出体重最大和最小的狗的名字，可以通过输入狗的名字，查找它的体重。

//对象数组的使用[Demo131.java]

//import java.io.\*;

import java.util.\*;

public class Demo131 {

public static void main(String[] args) **throws Exception** {//throws Exception将输入错误踢除程序块

//定义一个对象数组可以存放四只狗的对象数组

Dog dogs[]=new Dog[4];

//给各个狗赋初值

/\* dogs[0]=new Dog();

dogs[0].setName("花花");

dogs[0].setWeight(4.5f);

dogs[]\*/

//从控制台输入各个狗的信息

// InputStreamReader isr=new InputStreamReader(System.in);

// BufferedReader br=new BufferedReader(isr);

Scanner sr=new Scanner(System.in);

for(int i=0;i<4;i++){

dogs[i]=new Dog();//必需要使用new方法将数组指向Dog类

System.out.println("请输入第"+(i+1)+"狗名");

//从控制台读取狗名

String name=sr.nextLine();

//将名字赋给对象

dogs[i].setName(name);//将狗名使用set方法传入Dog类中

System.out.println("请输入"+(i+1)+"狗的体重");

String s\_weight=sr.nextLine();

float weight=Float.parseFloat(s\_weight);

//将名字赋给对象

dogs[i].setWeight(weight);//将狗体重使用set方法传入Dog类中

}

//计算总体重

float allWeight=0;

for(int i=0;i<4;i++){

allWeight+=dogs[i].getWeight();//将dogs数组中的狗体重从Dog类中取出并累加赋值给总体重

}

//计算平均体重

float avgWeight=allWeight/dogs.length;

System.out.println("总体重="+allWeight+"\t平均体重="+avgWeight);

//找出体重最大的狗

//假设第一狗体重最大

float maxWeight=dogs[0].getWeight();

int maxIndex=0;//定义用于比较体重的下标

//依次和第一只狗比较体重

for(int i=1;i<dogs.length;i++){

if(maxWeight<dogs[i].getWeight()){

//如何比较的狗体重大于第一只狗的体重则进行修改

maxWeight=dogs[i].getWeight();

maxIndex=i;

}

}

//找出体重最小的狗

float minWeight=dogs[0].getWeight();

int minIndex=0;

for(int j=1;j<dogs.length;j++){

if(minWeight>dogs[j].getWeight()){

//如何比较的狗体重小于第一只狗的体重则进行修改

minWeight=dogs[j].getWeight();

minIndex=j;

}

}

System.out.println("体重大的狗是第"+(maxIndex+1)+"狗，名字叫："+dogs[maxIndex].getName()+"\t体重是"+maxWeight);

System.out.println("体重小的狗是第"+(minIndex+1)+"狗，名字叫："+dogs[minIndex].getName()+"\t体重是"+minWeight);

//输入狗的名字查狗的体重

System.out.println("请输入你要找的狗的名字：");

String cname=sr.nextLine();

int cIndex=0;

for(int k=0;k<dogs.length;k++){

if(cname.equals(dogs[k].getName())){//对比狗名。变量名.equals()方法用于字符串比较内容是否一致。

System.out.println("你要找狗名"+ dogs[cIndex].getName()+"\t体重是"+dogs[cIndex].getWeight());

}

}

}

}

//定义一个狗类

class Dog{

private String name;

private float weight;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getWeight() {

return weight;

}

public void setWeight(float weight) {

this.weight = weight;

}

}

-------------------------------------------------------------------------------

**一维数组--小结**

1、数组可存放同一类型数据；

2、简单数据类型(int,float)数组，可直接赋值；

3、对象数组在定义后，赋值时需要再次为每个对象分配空间[即：new 对象]；

4、数组大小必须事先指定；

5、数组名可以理解为指向数组首地址的引用；

6、数组的下标是从0开始编号的。

===============================================================================

**java基本语法--排序**

排序的介绍

排序是将一群数据，依指定的顺序进行排列的过程。

排序分类：

1、内部排序法：指将需要处理的所有数据都加载到内部存储器中进行排序。包括(交换式排序法、选择式排序法和插入式排序法)；

2、外部排序法：数据量过大，无法全部加载到内存中，需要借助外部存储进行排序。包括(合并排序法和直接合并排序法)。

排序(Sorting)是数据处理中一种很重要的运算，同时也是很常用的运算，一般数据处理工作25%的时间都在进行排序。

简单地说，排序就是把一组记录(元素)按照某个域的值的递增(即由小到大)或递减(即由大到小)的次序重新排列的过程。

内部排序法--交换式排序法

交换式排序属于内部排序法，是运用数据值比较后，依判断规则对数据位置进行交换，以达到排序的目的。

交换式排序法又可分为两种：

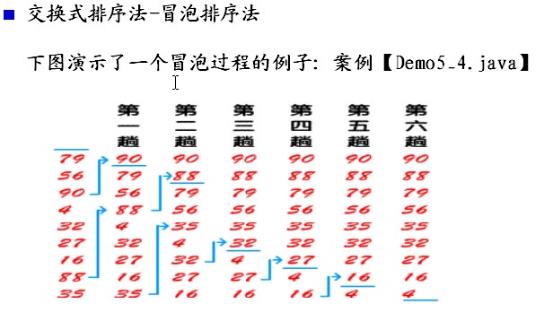
1、冒泡排序法(Bubble Sorting)

2、快速排序法(Quick Sorting)

交换式排序法--冒泡排序法

冒泡排序(Bubble Sorting)的基本思想是：通过对待排序序列从后向前(从下标较大的元素开始)，依次比较相邻元素的排序码，若发现逆序则交换，使排序码较小的元素逐渐从后部移向前部(从下标较大的单元移向下标较小的单元)，就象水底下的气泡一样逐渐向上冒。

因为排序的过程中，各元素不断接近自己的位置，如果一趟比较下来没有进行过交换，就说明序列有序，因此要在排序过程中设置一个标志flag判断元素是否进行过交换。从而减少不必要的比较。



**冒泡法程序演示[Demo132]排序10个数用时15秒**

//演示冒泡排序法

public class Demo132 {

public static void main(String[] args) {

int arr[]={1,6,0,-1,9,-100,90};

int temp=0;

//排序

//外层循环，可以决定一共走趟

for(int i=0;i<arr.length-1;i++){

//内层循环，开始逐个比较，如果发现前一个数比后一个数大则交换

for(int j=0;j<arr.length-1-i;j++){

if(arr[j]>arr[j+1]){

//换位

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

//输出最后结果

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+"\t");

}

}

}

-------------------------------------------------------------------------------

**选择式排序法--选择排序法**

选择式排序也属于内部排序法，是从欲排序的数据中，按指定的规则选出某一元素，经过和其他元素重整，再依原则交换位置后达到排序的目的。

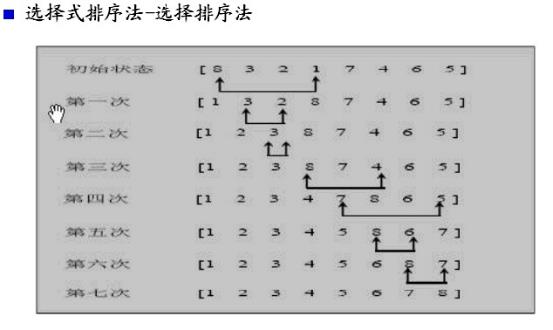
选择式排序又可分为两种：

1、选择排序法(Selection Sorting)

2、堆排序法(Heap Sorting)

选择排序(Select Sorting)也是一种简单的排序方法。它的基本思想是：第一次从R[0]-R[n-1]中选取最小值，与R[0]交换，第二次从R[1]-R[n-1]中选取最小值，与R[1]交换，第三次从R[2]-R[n-1]中选取最小值，与R[2]交换，...，第i次从R[i-1]-R[n-1]中选取最小值，与R[i-1]交换，...，第n-1次从R[n-2]-R[n-1]中选取最小值，与R[n-2]交换，总共通过n-1次，得到一个按排序码从小到大排列的有序序列。

例如，给定n=8，数组R中的8个元素的排序码为：(8,3,2,1,7,4,6,5)，选择排序过程。



//选择排序法[Demo133.java]排序10万个数用时7秒

public class Demo133{

public static void main(String []args){

int arr[]={8,3,2,1,7,4,6,5};

int temp=0;

for(int j=0;j<arr.length-1;j++){

//认为第一个数就是最小数

int min=arr[j];

//记录最小数的下标

int minIndex=j;

for(int k=j+1;k<arr.length;k++){

if(min>arr[k]){

//修改最小值

min=arr[k];

minIndex=k;

}

}

//当退出for循环时就找到这次的最小值

temp=arr[j];

arr[j]=arr[minIndex];

arr[minIndex]=temp;

}

//输出最后结果

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+"\t");

}

}

}

-------------------------------------------------------------------------------

java基本语法--排序

**插入式排序法--插入排序法**

插入式排序属于内部排序法，是对于欲排序的元素以插入的方式找寻该元素的适当位置，以达到排序的目的。

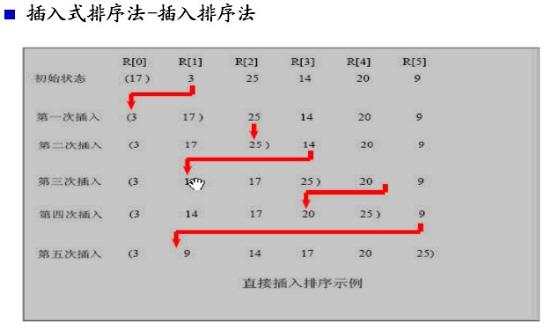
插入式排序法又可分为**3种**：

1、插入排序法(Insertion Sorting)

2、谢耳排序法(Shell Sorting)(欧洲人员喜欢使用)

3、二叉树排序法(Binary-tree Sorting)

插入排序(Insertion Sorting)的基本思想是：把n个待排序的元素看成为一个有序表和一个无序表，开始有序表只包含一个元素，无序表中包含有n-1个元素，排序过程中每次从无序表中取出第一个元素，把它的排序码依次与有序表元素的排序码进行比较，将它插入到有序表中的适当位置，使之成为新的有序表。



//插入式排序法[Demo134.java]排序10万数据用时2秒

public class Demo134{

public static void main(String []args){

int arr[]={23,15,-13,62,5,-23,0,17};

for(int i=1;i<arr.length;i++){

int insertVal=arr[i];

//insertVal准备和前一个数比较

int index=i-1;

while(index>=0&&insertVal<arr[index]){

//将把arr[index]向后移动一位

arr[index+1]=arr[index];

//让index向前移动一位

index--;

}

//将insertVal插入到适当位置

arr[index+1]=insertVal;

}

//输出最后结果

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+"\t");

}

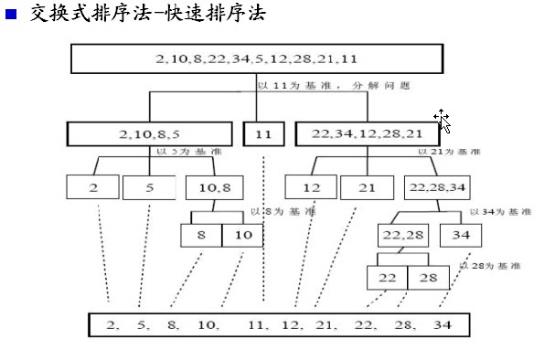
}

}

-------------------------------------------------------------------------------

**交换式排序法--快速排序法**

快速排序(QuickSorting)是对冒泡排序的一种改进，由C.A.R.Hoare在1962年提出。它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。



//快速排序法[Demo135.java]排序1亿数据用时14秒

public class Demo135{

public static void main(String []args){

int arr[]={-1,-5,6,2,0,9,-3,-8,12,7};

QuickSort qs=new QuickSort();

qs.sort(0, arr.length-1, arr);

//输出最后结果

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+"\t");

}

}

}

class QuickSort{

public void sort(int left,int right,int [] arr){

int l=left;

int r=right;

int pivot=arr[(left+right)/2];//找中间值

int temp=0;

while(l<r){

while(arr[l]<pivot) l++;

while(arr[r]>pivot) r--;

if(l>=r) break;

temp=arr[l];

arr[l]=arr[r];

arr[r]=temp;

if(arr[l]==pivot) --r;

if(arr[r]==pivot) ++l;

}

if(l==r){

l++;

r--;

}

if(left<r) sort(left,r,arr);

if(right>l) sort(l,right,arr);

}

}

-------------------------------------------------------------------------------

其它排序法--**选堆排序法**(主考高级程序员，工作中基本上用不到，不再详解)

将排序码k1,k2,k3,...,kn表示成一棵完全二叉树，然后从第n/2个排序码开妈筛选，使由该结点组成的子二叉树符合堆的定义，然后从第n/2-1个排序码重复刚才操作，直到第一个排序码止，这时候，该二叉树符合堆的定义，初始堆已经建立。

接着，可以按如下方法进行堆排序：将堆中第一个结点(二叉树根结点)和最后一个结点的数据进行交换(k1与kn)，再将k1--kn-1重新建堆，然后k1和kn-1交换，再将k1--kn-2重新建堆，然后k1和kn-2交换，如此重复下去，每次重新建堆的元素个数不断减1，直到重新建堆的元素个数仅剩一个为止。这时堆排序已经完成，则排序码k1,k2,k3,...kn已排成一个有序序列。

若排序是从小到大排列，则可以建立大根堆实现堆排序，若排序是从大到小排列，则可以用建立小根堆实现堆排序。

其它排序法--**希尔排序法**(知道有这个排序法即可)

希尔排序(Shell Sorting)又称为“缩小增量排序”。是1959年由D.L.Shell提出来的。该方法的基本思想是：先将整个待排元素序列分割成若干个子序列(由相隔某个“增量”的元素组成的)分别进行直接插入排序，待整个序列中的元素基本有序(增量足够小)时，再对全体元素进行一次直接插入排序。

因为直接插入排序在元素基本有序的情况下(接近最好情况)，效率是很高的，因此希尔排序在时间效率上比前两种方法有较大提高。

其它排序法--**二叉树排序法**

二分插入排序(Binary Insert Sorting)的基本思想是：在有序表中采用二分查找的方法查找待排元素的插入位置。

其处理过程：先将第一个元素作为有序序列，进行n-1次插入，用二分查找的方法查找待排元素的插入位置，将待排元素插入。

-------------------------------------------------------------------------------

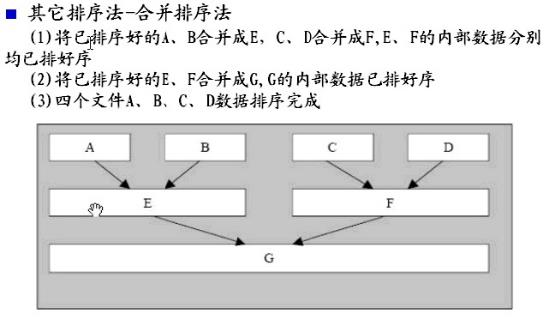
外部排序法

其它排序法--**合并排序法**(最为常用的排序方法)

合并排序法(Merge Sorting)是外部排序最常使用的排序方法。若数据量太大无法一次完全加载内存，可使用外部辅助内存来处理排序数据，主要应用在文件排序。

排序方法：

将欲排序的数据分别存在数个文件大小可加载内存的文件中，再针对各个文件分别使用“内部排序法”将文件中的数据排序好写回文件。再对所有已排序好的文件两两合并，直到所有文件合并成一个文件后，则数据排序完成。



1、将已排序好的A、B合并成E，C、D合并成F，E、F的内部数据分别均已排好序

2、将已排序好的E、F合并成G，G的内部数据已排好序

3、四个文件A、B、C、D数据排序完成

//合并排序法[Demo137.java]

public class Demo137{

public static void main(String[] args)

{

Merge m=new Merge();

int a[]={5,4,10,8,7,9};

m.merge\_sort(a,0,a.length-1);

}

}

class Merge{

//递归分成小部分

public void merge\_sort(int[] arrays,int start,int end){

if(start<end){

int m=(start+end)/2;

merge\_sort(arrays,start,m);

merge\_sort(arrays,m+1,end);

combin\_arrays(arrays,start,m,end);

}

}

//合并数组

public void combin\_arrays(int[] arrays,int start,int m,int end){

int length=end-start+1;

int temp[]=new int[length];//用来存放比较的数组，用完复制回到原来的数组

int i=start;

int j=m+1;

int c=0;

while(i<=m &&j<=end){

if(arrays[i]<arrays[j]){

temp[c]=arrays[i];

i++;

c++;

}else{

temp[c]=arrays[j];

j++;

c++;

}

}

while(i<=m){

temp[c]=arrays[i];

i++;

}

while(j<=end){

temp[c]=arrays[j];

j++;

}

c=0;

for(int t=start;t<=end;t++,c++){

arrays[t]=temp[c];

}

snp(arrays);

}

//打印数组

public void snp(int[] arrays){

for(int i=0;i<arrays.length;i++){

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

}

System.out.println();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**查找**

在java中，常用的查找方式有两种：

1、顺序查找(最简单，效率最低)

2、二分查找

//功能：二分查找[Demo136.java]

import java.util.\*;

public class Demo136 {

public static void main(String[] args) {

int arr[]={2,5,7,12,25};//定义arr数组并赋值

System.out.print("请输入你需要查找的数：");

Scanner sr=new Scanner(System.in);

int a=sr.nextInt();

BinaryFind bf=new BinaryFind();//创建BinaryFind对象

bf.find(0,arr.length-1,a,arr);//调用find方法，并将数据传给方法

}

}

//二分法

class BinaryFind{

public void find(int leftIndex,int rightIndex,int val,int arr[]){

//首先找到中间的数

int midIndex=((rightIndex+leftIndex)/2);

int midVal=arr[midIndex];

if(rightIndex>=leftIndex){

//如果要找的数比midVal大

if(midVal>val){

//在arr数组左边数列中找

find(leftIndex,midIndex-1,val,arr);

}else if(midVal<val){

//在arr数组右边数列中找

find(midIndex+1,rightIndex,val,arr);

}else if(midVal==val){

System.out.println("数组arr["+midIndex+"]中的数字是"+arr[midIndex]);

}

}else{

System.out.println("没有找到你要找的数!");

}

}

}

-------------------------------------------------------------------------------

**递归**

什么是递归？

**递归算法：是一种直接或者间接地调用自身的算法**，就我个人的理解而言，不论是直接还是间接，其算法的流程走向必须形成封闭（即本身属于广义上的循环过程），否则递归将不能形成。在计算机编写程序中，递归算法对解决很多类问题是十分有效，它使算法的描述简洁而且易于理解。

**递归算法的特点：**

递归过程一般通过函数或子过程来实现。

**递归算法：在函数或子过程的内部，直接或者间接地调用自己的算法。**

递归算法的实质：是把问题转化为规模缩小了的同类问题的子问题。然后递归调用函数(或过程)来表示问题的解。

递归算法解决问题的特点：

(1)递归就是在过程或函数里调用自身。

(2)在使用递增归策略时，必须有一个明确的递归结束条件，称为递归出口。

(3)递归算法解题通常显得很简洁，但递归算法解题的**运行效率较低**，即**占用内存很大**，有时这种情况**用栈来代替解决**。所以一般**不提倡用递归算法设计程序**。

(4)在递归调用的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成栈溢出等。所以一般不提倡用递归算法设计程序。

递归算法所体现的“重复”一般有三个要求：

一是每次调用在规模上都有所缩小(通常是减半)；

二是相邻两次重复之间有紧密的联系，前一次要为后一次做准备(通常前一次的输出就作为后一次的输入)；

三是在问题的规模极小时必须用直接给出解答而不再进行递归调用，因而每次递归调用都是有条件的(以规模未达到直接解答的大小为条件)，无条件递归调用将会成为死循环而不能正常结束。

最后我再补充一点:递归只是用在简单的循环场合中,这种场合就是递归函数体只含有一个IF-ELAE语言的情况:

权限修饰符+(static)+函数返回类型+函数名(参数列表){

if(递归结束条件){递归结束终值赋值语句;return 返回值;}

else{累计计算+循环调用函数语句;

return 返回值;}

}

其余的复杂循环(甚至嵌套)情况建议不用递归,因为递归过程中所用到的变量都在递归结束前的过程中一直占用着内存,如果循环过程又趋于复杂则内存耗用量将显著提升,运行速度也将下降.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**java基本语法--多维数组**

多维数组--二维数组

多维数组我们只介绍二维数组

1、定义

**语法：类型 数组名[][]=new 类型[大小][大小];**

比如：int a[][]=new int[2][3];

2、分析

二维数组在内存中存在的形式

3、案例，请用二维数组输出如下图形

0 0 0 0 0 0

0 0 1 0 0 0

0 2 0 3 0 0

0 0 0 0 0 0

//二维数组演示[Demo138.java]

public class Demo138 {

public static void main(String[] args) {

int a[][]=new int[4][6];//定义二维数组a4行6列

a[1][2]=1;

a[2][1]=2;

a[2][3]=3;

//把图形输出

for(int i=0;i<4;i++){//控制行

for(int j=0;j<6;j++){//控制列

System.out.print(a[i][j]+"\t");//输出数组

}

System.out.println();//换行

}

}

}

-------------------------------------------------------------------------------

二进制、位运算

思考题

1、请看下面的代码段，回答a,b,c,d,e结果是多少？

public static void main(String []args){

int a=1>>2;

int b=-1>>2;

int c=1<<2;

int d=-1<<2;

int e=3>>>2;

//a,b,c,d,e结果是多少

System.out.println("a="+a);//a=0

System.out.println("b="+b);//b=-1

System.out.println("c="+c);//c=4

System.out.println("d="+d);//d=-4

System.out.println("e="+e);//e=0

}

注：">>"代表算术右移，"<<"代表算术左移，">>>"代表逻辑右移

2、请回答在java中，下面的表达式运算的结果是：

~2=? //-3

2&3=? //2

2|3=? //3

~-5=? //4

13&7=? //5

5|4=? //5

-3^3=? //-2

注："~"代表位取反，"&"代表位与，"|"代表位或，"^"代表位异或

**二进制--基本概念**

二进制是逢2进位的进位制，0、1是基本算符。

现代的电子计算机技术全部采用的是二进制，因为它只使用0、1两个数字符号，非常简单方便，易于用电子方式实现。计算机内部处理的信息，都是采用二进制数来表示的。二进制(Binary)数用0和1两个数字及其组合来表示任何数。进位规则是“逢2进1”，数字1在不同的位上代表不同的值，按从右至左的次序，这个值以二倍递增。

**注：1个字节=8位bit,**

bit最高位是符号位如：■□□□□□□□黑色方框为符号位。

符号位0代表正数，1代表负数

**二进制--原码、反码、补码**

对于有符号的而言：

1、二进制的最高位是符号位：0表示正数，1表示负数

2、正数的原码、反码、补码都一样

3、负数的反码=它的原码符号位不变，其它位取反

4、负数的补码=它的反码+1

5、0的反码，补码都是0

6、java没有无符号数，换言之，java中的数都是有符号的

7、在计算机运算的时候，都是以补码的方式来运算的。

**位运算符和移位运算**

java中有4个位运算，分别是**“按位与&、按位或|、按位异或^，按位取反~”**，它们的运算规则是：

按位与&：两位全为1，结果为1

按位或|：两位有一个为1，结果为1

按位异或^：两位一个为0，一个为1，结果为1

按位取反：0->1，1->0

java中有3个移位运算符：

**>>、<<算术右移和算术左移，运算规则：**

**算术右移：**低位溢出，符号位不变，并用符号位补溢出的高位

**算术左移：**符号位不变，低位补0

**>>>逻辑右移，运算规则是：**低们溢出，高位补0

-------------------------------------------------------------------------------

集合框架

一个问题？

前面我们学习了数组，充分体会到数组的优越性，就是可以存储同一类型的数据，但是我们假设有这样的需求，大家看看如何解决？[Demo140.java]

请做一个公司职员薪水管理系统，要求完成如下功能：

1、当有新员工时，将该员工加入到管理系统

2、可以根据员工号，显示该员工的信息

3、可以显示所有员工信息

4、可以修改员工的薪水

5、当员工离职时，将该员工从管理系统中删除

6、可以按照薪水从低到高顺序排序[思考题]

7、可以统计员工的平均工资和最低和最高工资

//ArrayList集合类的使用

//公司职员薪水管理系统

import java.util.\*;

public class Demo140 {

public static void main(String[] args) {

//创建EmpManage对象

EmpManage em=new EmpManage();

Scanner sr=new Scanner(System.in);

//作出一个菜单

while(true){

System.out.println("公司职员薪水管理系统");

System.out.println("1、录入新员工");

System.out.println("2、根据工号查询信息");

System.out.println("3、查询所有员工信息");

System.out.println("4、通过工号修改员工薪水");

System.out.println("5、删除员工信息");

System.out.println("6、按薪水高低排序");

System.out.println("7、计算平均工资及最高(低)工资");

System.out.println("0、退出系统");

System.out.print("请输入对应的数字进行操作：");

int sel=sr.nextInt();

if(sel==1){

System.out.println("请录入新员工的信息");

System.out.print("工号:");

String empNo=sr.next();

System.out.print("姓名:");

String name=sr.next();

System.out.print("工资:");

float sal=sr.nextFloat();

//构建emp对象

Emp emp=new Emp(empNo,name,sal);

//将empNo,name,sal的值传给构造函数Emp

em.addEmp(emp);

System.out.println("创建新员工"+name+"成功!");

}else if(sel==2){

System.out.println("请录入员工工号：");

String empNo=sr.next();

em.showInfo(empNo);

}

else if(sel==3){

System.out.println("公司所有员工信息如下：");

em.AllInfo();

}

else if(sel==4){

System.out.println("请输入工号：");

String empNo=sr.next();

System.out.println("将工资修改为：");

float newSal=sr.nextFloat();

em.updateSal(empNo, newSal);

}

else if(sel==5){

System.out.println("请输入要删除人员的工号：");

String empNo=sr.next();

em.delEmp(empNo);

}

else if(sel==6){

System.out.println("已按薪资高低进行排序如下：");

em.SortSal();

}

else if(sel==7){

System.out.println("显示平均工资及最高、最低工资人员信息如下：");

em.Average();

}

else if(sel==0){

System.out.println("已正常退出!");

System.exit(0);

}else{

System.out.println("输入错误，请重新输入!");

}

}

}

}

//创建员工管理类

class EmpManage{

private ArrayList<Emp> al=null;

//创建构造函数，初始化成员变量

public EmpManage(){

al=new ArrayList<Emp>();

}

//加入员工

public void addEmp(Emp emp){//传入员工信息

al.add(emp);

}

//根据员工工号显示员工的相关信息

public void showInfo(String empNo){//将工号传入showInfo方法中

//遍历整个ArrayList

for(int i=0;i<al.size();i++){

//取出Emp对象

Emp emp=(Emp)al.get(i);

//比较编号

if(emp.getEmpNo().equals(empNo)){//由于empNo类型为String，所以要使用equals进行内容比较，不可以使用==地址比较

System.out.println("找到该员工，他的信息是：");

System.out.println("工号:"+empNo+"\t姓号:"+emp.getName()+"\t薪水:"+emp.getSal());

}else{

System.out.println("工号不存在或无此人!");

}

}

}

//显示所有员工信息

public void AllInfo(){

for(int i=0;i<al.size()-1;i++){//ArrayList集合类的al大小，控制循环

for(int j=1;j<al.size()-i;j++){//将al中的值进行循环比较

Emp emp1=(Emp)al.get(j-1);

Emp emp2=(Emp)al.get(j);

//使用compareTo方法进行Sting类型值比较

if(emp1.getEmpNo().compareTo(emp2.getEmpNo())>0){

al.set(j,emp1);//交换值并重写入al中

al.set(j-1,emp2);//交换值并重写入al中

}

}

}

for(Emp emp:al){

System.out.println("工号："+emp.getEmpNo()+"\t姓名："+emp.getName()+"\t工资："+emp.getSal());

}

}

//修改员工的薪水

public void updateSal(String empNo,float newSal){

//遍历整个ArrayList

for(int i=0;i<al.size();i++){

//取出Emp对象

Emp emp=(Emp)al.get(i);

if(emp.getEmpNo().equals(empNo)){

//修改新水

emp.setSal(newSal);//setSal会将修改的薪水传和ArrayList中

System.out.println("已将"+emp.getName()+"调整为:"+newSal);

}

}

System.out.println("工号不存在或无此人，无法进行相应操作!");

}

//员工离职删除指定员工

public void delEmp(String empNo){

//遍历整个ArrayList

for(int i=0;i<al.size();i++){

//取出Emp对象

Emp emp=(Emp)al.get(i);

if(emp.getEmpNo().equals(empNo)){

//按工号删除

al.remove(i);//也可以使用al.remove(emp);

System.out.println("已将"+emp.getName()+"信息清除!");

}

}

}

//按薪水高低排序

public void SortSal(){

for(int i=0;i<al.size()-1;i++){//ArrayList集合类的al大小，控制循环

for(int j=1;j<al.size()-i;j++){//将al中的值进行循环比较

Emp emp1=(Emp)al.get(j-1);

Emp emp2=(Emp)al.get(j);

if(emp1.getSal()<emp2.getSal()){//比较sal的值大小

al.set(j,emp1);//交换值并重写入al中

al.set(j-1,emp2);//交换值并重写入al中

}

}

}

for(Emp emp:al){

System.out.println("工号："+emp.getEmpNo()+"\t姓名："+emp.getName()+"\t工资："+emp.getSal());

}

}

//计算平均工资并找出最高工资和最低工资的员工

public void Average(){

float sum=0f,ave=0f;

//遍历整个ArrayList

for(int k=0;k<al.size();k++){

//取出Emp对象

Emp emp=(Emp)al.get(k);

sum=emp.getSal()+sum;

}

ave=sum/al.size();

System.out.println("共有员工"+al.size()+"人\t总工资为："+sum+"\t平均工资为："+ave);

//找出最高工资与最低式资

for(int i=0;i<al.size()-1;i++){//ArrayList集合类的al大小，控制循环

for(int j=1;j<al.size()-i;j++){//将al中的值进行循环比较

Emp emp1=(Emp)al.get(j-1);

Emp emp2=(Emp)al.get(j);

if(emp1.getSal()<emp2.getSal()){//比较sal的值大小

al.set(j,emp1);//交换值并重写入al中

al.set(j-1,emp2);//交换值并重写入al中

}

}

}

for(int i=0;i<al.size();i++){

if(i==0){

Emp emp=(Emp)al.get(i);

System.out.println("工资最高的人员是："+emp.getName()+"\t薪水是："+emp.getSal());

}else if(i==al.size()-1){

Emp emp=(Emp)al.get(i);

System.out.println("工资最低的人员是："+emp.getName()+"\t薪水是："+emp.getSal());

}

}

}

}

//创建员工类

class Emp{

//定义成员变量工号、姓名、薪水

private String empNo;

private String name;

private float sal;

//创建构造函数，初始化成员变量

public Emp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

//使用set、get方法进行数据传递

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

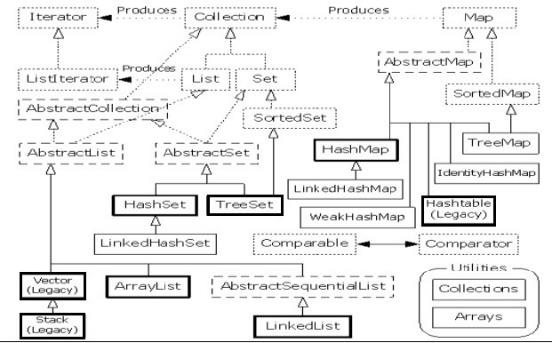
this.sal = sal;

}

}

-------------------------------------------------------------------------------

# 集合框架--使用



从上面的图可以看出java集合类主要有以下几种：

## List结构的集合类

ArrayList类，LinkedList类，Vector类，Stack类

集合框架List结构集合类--**ArrayList类的使用(无同步性，线程不安全)[Demo139.java]**

//java集合类用法--List结构--ArrayList类

import java.util.\*;//集合类基本上在util包中

public class Demo139 {

public static void main(String[] args) {

//定义ArrayList对象

ArrayList al=new ArrayList();

//显示大小

System.out.println("al大小:"+al.size());

//向all中加入数据(类型是Object)

//创建一个职员

Clerk clerk1=new Clerk("宋江",50,1000);

Clerk clerk2=new Clerk("吴用",45,1200);

Clerk clerk3=new Clerk("林冲",35,1300);

//将clerk1加入到al中

al.add(clerk1);

al.add(clerk2);

al.add(clerk3);

//可不可以放入同样的对象？

al.add(clerk1);

//显示大小

System.out.println("al大小:"+al.size());

//如何访问al中的对象(数据)

//访问第一个对象

//Clerk temp=(Clerk)al.get(0);

//System.out.println("第一个人的名字是："+temp.getName());

//遍历al所有的对象(数据)

for(int i=0;i<al.size();i++){

Clerk temp=(Clerk)al.get(i);

System.out.println("名字："+temp.getName());

}

//如何从al中删除一个对象

al.remove(1);

System.out.println("===删除吴用===");

//遍历al所有的对象(数据)

for(int i=0;i<al.size();i++){

Clerk temp=(Clerk)al.get(i);

System.out.println("名字："+temp.getName());

}

}

}

//定义一个员工类

class Clerk{

private String name;

private int age;

private float sal;

public Clerk(String name,int age,float sal){

this.name=name;

this.age=age;

this.sal=sal;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

## LinkedList集合类的使用方法[Demo141.java]

//LinkedList集合类的使用

import java.util.\*;

public class Demo141 {

public static void main(String[] args) {

LinkedList ll=new LinkedList();

Empp emp1=new Empp("sa01","aa",1.2f);

Empp emp2=new Empp("sa02","bb",1.2f);

Empp emp3=new Empp("sa03","cc",1.2f);

//addFirst表示把emp1加载(链表)队列的最前面

ll.addFirst(emp1);//addFirst方法是可以插入在数组之前

ll.addFirst(emp2);//也可以理解为addFirst方法是后进先出的方法

//addLast表示把emp3加载(链表)队列的后面

ll.addLast(emp3);

System.out.println("测试LinkedList集合类中的addFist及addLast方法");

for(int i=0;i<ll.size();i++){

System.out.println(((Empp)ll.get(i)).getName());

}

//remove表示将某一条数据进行删除

ll.remove(emp1);//将ll中的emp1数据删除

System.out.println("测试LinkedList集合类中的remove方法");

for(int i=0;i<ll.size();i++){

System.out.println(((Empp)ll.get(i)).getName());

}

ll.removeAll(ll);//清除整个链表

System.out.println("测试LinkedList集合类中的remmoveall方法");

for(int i=0;i<ll.size();i++){

System.out.println(((Empp)ll.get(i)).getName());

}

}

}

//创建员工类

class Empp{//在同一个包中的类，可以同包中的其它class文件直接访问或调用

//定义成员变量工号、姓名、薪水

private String empNo;

private String name;

private float sal;

//创建构造函数，初始化成员变量

public Empp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

//使用set、get方法进行数据传递

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

===============================================================================

## Vector集合类的使用(线程安全具有同步性) [Demo142.java]

//Vector集合类(向量)的使用方法

import java.util.\*;

public class Demo142 {

public static void main(String[] args) {

//Vector的用法

Vector vv=new Vector();

AEmp emp1=new AEmp("1","aa",1.2f);

AEmp emp2=new AEmp("2","bb",1.2f);

AEmp emp3=new AEmp("3","cc",1.2f);

vv.add(emp1);

vv.add(emp2);

vv.add(emp3);

//遍历

for(int i=0;i<vv.size();i++){

AEmp emp=(AEmp)vv.get(i);

System.out.println(emp.getName());

}

}

}

//创建员工类

class AEmp{

//定义成员变量工号、姓名、薪水

private String empNo;

private String name;

private float sal;

//创建构造函数，初始化成员变量

public AEmp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

//使用set、get方法进行数据传递

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

===============================================================================

## Stack集合类(栈)的使用[Demo143.java]

//Stack集合类(栈)的使用方法

package com.haiding.set;

import java.util.\*;

public class Demo143 {

public static void main(String[] args) {

//Stack的用法

Stack stack=new Stack();

AEmp emp1=new AEmp("s1","aa",1.2f);

AEmp emp2=new AEmp("s2","bb",1.2f);

stack.add(emp1);

stack.add(emp2);

for(int i=0;i<stack.size();i++){

System.out.println(((AEmp)stack.get(i)).getName());

}

}

}

//创建员工类

class AEmp{

//定义成员变量工号、姓名、薪水

private String empNo;

private String name;

private float sal;

//创建构造函数，初始化成员变量

public AEmp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

//使用set、get方法进行数据传递

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

-------------------------------------------------------------------------------

集合框架--深入讨论

## ArrayList和Vector的区别

ArrayList与Vector都是java的集合类，都可以用来存放java对象，这是他们的相同点，但是他们也有区别：

1、同步性

Vector是线程同步的。这个类中的一些方法保证了Vector中的对象是线程安全的。而ArrayList则是线程异步的，因此ArrayList中的对象并不是线程安全的。因为**同步的要求会影响执行的效率**，所以如果你不需要线程安全的集合那么使用ArrayList是一个很好的选择，这样可以避免由于同步带来的不必要的性能开销。

2、数据增长

从内部实现机制来讲ArrayList和Vector都是使用数组(Array)来控制集合中的对象。当你向这两种类型中增加元素的时候，如果元素的数目超出了内部数组目前的长度它们都需要扩展内部数组的长度，Vector缺省情况下自动增长原来一倍的数组长度，ArrayList是原来的50%，所以最后你获得的这个集合所占的空间总是比你实际需要的要大。所以如果你要在集合中保存大量的数据那么使用Vector有一些优势，因为你可以通过设置集合的初始化大小来避免不必要的资源开销。

===============================================================================

## Map结构的集合类

HashMap类，Hashtable类

HashMap集合类的使用[Demo143.java]

//HashMap集合类的使用

import java.util.\*;

public class Demo143 {

public static void main(String[] args) {

//创建HashMap对象

HashMap hm=new HashMap();

Emp emp1=new Emp("s001","aa",3.4f);

Emp emp2=new Emp("s002","bb",5.6f);

Emp emp3=new Emp("s003","cc",1.2f);

//将emp放入到hm中

//hm.put(null,null);//可以放空值

hm.put("s001", emp1);

hm.put("s002", emp2);

hm.put("s002", emp3);//不允许key重复,所以emp3会覆盖emp2

//如果你要查找编号是s002

if(hm.containsKey("s002")){//取键值containsKey

System.out.println("有该员工");

//如何取出，键<key>值

Emp emp=(Emp)hm.get("s002");

System.out.println("名字"+emp.getName());

}else{

System.out.println("没该员工");

}

//遍历HashMap中所有的key和value值

//Iterator迭代

Iterator it=hm.keySet().iterator();

//hasNext返回一个boolean值

while(it.hasNext()){

//如果有下一个取出key值

String key=it.next().toString();

//通过key取出value

Emp emp=(Emp)hm.get(key);

System.out.println("名字："+emp.getName());

System.out.println("工资："+emp.getSal());

}

}

}

//创建员工类

class Emp{

//定义成员变量工号、姓名、薪水

private String empNo;

private String name;

private float sal;

//创建构造函数，初始化成员变量

public Emp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

//使用set、get方法进行数据传递

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

-------------------------------------------------------------------------------

## Hashtable集合类的使用(Hashtable具有同步性，线程安全)

import java.util.\*;

public class Demo144{

public static void main(String []args){

Hashtable ht=new Hashtable();//Hashtable与HsahMap在用法上一致

Emp emp4=new Emp("s101","a1",2.2f);

Emp emp5=new Emp("s102","a2",1.2f);

Emp emp6=new Emp("s103","a3",4.2f);

ht.put("s101", emp4);

ht.put("s102", emp5);

ht.put("s103", emp6);

//遍历

for(Iterator it=ht.keySet().iterator();it.hasNext();){

String key=it.next().toString();

Emp emp=(Emp)ht.get(key);

System.out.println("名字："+emp.getName()+"\t工资："+emp.getSal());

}

}

}

//创建员工类

class Emp{

//定义成员变量工号、姓名、薪水

private String empNo;

private String name;

private float sal;

//创建构造函数，初始化成员变量

public Emp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

//使用set、get方法进行数据传递

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

-------------------------------------------------------------------------------

## HashMap和Hashtable集合类的区别

HashMap与Hashtable都是java的集合类，都可以用来存放java对象，这是他们的相同点，但是他们也有区别。

1、历史原因

Hashtable是基于陈旧的Dictionary类的，HashMap是java 1.2引进的Map接口的一个实现。

2、同步性

Hashtable是线程同步的。这个类中的一些方法保证了Hashtable中的对象是线程安全的。而HashMap则是线程异步的，因此HashMap中的对象并不是线程安全的。因为同步的要求会影响执行的效率，所以如果你不需要线程安全的集合那么使用HashMap是一个很好的选择，这样可以避免由于同步带来的不必要的性能开销，从而提高效率。

3、值

HashMap可以让你将空值作为一个表的条目的key或value但是Hashtable是不能放入空值的(null)

集合框架--深入讨论

进一步理解集合框架

java的设计者给我们提供了这些集合类，在后面编程中是相当有用的，具体什么时候用什么集合，要根据我们刚才分析的集合异同来选取。

**如何选用集合类？**

1、要求线程安全，使用Vector、Hashtable

2、不要求线程安全，使用ArrayList,LinkedList,HashMap

3、要求key和value键值，则使用HashMap,Hashtable

4、数据量很大，又要线程安全，则使用Vector

===============================================================================

## Set结构的集合类

**HashSet类，TreeSet类**

HashSet是基于HashMap实现的，HashSet底层采用HashMap来保存所有元素。

hashCode和equal()是HashMap用的，因为无需排序所以只需要关注定位和唯一性即可

hashCode是用来计算hash值的，hash值是用来确定hash表索引的

hash表中的一个索引存放的是一张链表，所以还要通过equal方法循环比较链上的每一个对象才可以真正定位到键值对应的Entry

put时，如果hash表中没定定位到，就在链表前加一个Entry，如果定位到了，则更换Entry中的value(值)并返回旧value(值)

覆写key的hashCode()和equal()时一定要注意，不要把它们和可变属性关联上，否则属性变了之后hashCode会变，equal也会为false，这样在Map中就找不到它了而且这样的对象因为找不到它所以得不到释放，这样就变成了一个无效引用(相当于内存泄漏)

//HashSet的使用方法[Demo145.java]

import java.util.\*;

public class Demo145{

public static void main(String []args){

HashSet<Emp> hs=new HashSet<Emp>();

Emp emp1=new Emp("s001","aa",1.2f);

Emp emp2=new Emp("s002","bb",1.6f);

Emp emp3=new Emp("s003","cc",1.8f);

Emp emp4=new Emp("s001","aa",1.2f);

hs.add(emp1);

hs.add(emp2);

hs.add(emp3);

hs.add(emp4);

hs.add(emp1);//重复的emp1,HashSet会自动去除

System.out.println("HashSet\_size="+hs.size());

System.out.println();

ArrayList<Emp> al=new ArrayList<Emp>();

Emp emp5=new Emp("s004","dd",1.0f);

Emp emp6=new Emp("s005","ee",2.5f);

al.add(emp5);

al.add(emp6);

//al.add(emp1);

hs.addAll(al);//将al中的值加到hs中，并去除重复的emp1

System.out.println("HashSet\_ArrayList\_size="+hs.size());

System.out.println();

//转换数组o[]，遍历并输出HashSet中的无素

Object o[]=hs.toArray();

for(int i=0;i<o.length;i++){

System.out.println("工号:"+((Emp)o[i]).getEmpNo()+"\t姓名:"+((Emp)o[i]).getName()+"\t工资:"+((Emp)o[i]).getSal());

}

}

}

class Emp{

private String empNo;

private String name;

private float sal;

public Emp(String empNo,String name,float sal){

this.empNo=empNo;

this.name=name;

this.sal=sal;

}

public String getEmpNo() {

return empNo;

}

public void setEmpNo(String empNo) {

this.empNo = empNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getSal() {

return sal;

}

public void setSal(float sal) {

this.sal = sal;

}

}

-------------------------------------------------------------------------------

**TreeSet集合类是一个有序集合**，它的元素按照升序排序，默认是自然顺序排列，也就是说

TreeSet中的对象元素需要实现Comparable接口。TreeSet与HashSet类一样没有get()方法来获取列表中的元素，所以也只能通过迭代器方法来获取。

由于TreeMap需要排序，**所以需要一个Comparator为键值进行大小比较**，当然也是用Comparator定位的Comparator可以在创建TreeMap时指定，这时排序时使用Comparator.compare

如果创建时没有指定Comparator那么就会使用key.compareTo()方法，这就**要求key必须实现Comparable接口**

**TreeMap是使用Tree数据结构实现的**，所以使用compare接口就可以完成定位了。

**TreeSet是依靠TreeMap来实现的**，TreeSet是一个有序集合，它的元素按照升序排列，默认是按照自然顺序排列，也就是说TreeSet中的对象元素需要实现Comparable接口。

**TreeSet类中跟HashSet类一要也没有get()方法来获取列表中的元素，所以也只能通过迭代器的方法来获取 。**

//TreeSet集合类的使用[Demo146.java]

import java.util.\*;

public class Demo146{

public static void main(String[] args){

//传递一个比较器来实现你自己的排序方式

TreeSet tr =new TreeSet(new Student.StudentComparator());

tr.add(new Student(301,"张三"));//将学生数据写入TreeSet集合类的tr中

tr.add(new Student(201,"李二"));

tr.add(new Student(101,"王五"));

tr.add(new Student(101,"穷一"));

Iterator it=tr.iterator();//迭代器，遍历

while(it.hasNext()){//判断是否有下一个元素

System.out.println(it.next());

}

}

}

//创建Strudent学生类并实现Comparable与Comparator接口

class Student implements Comparable,Comparator{

private int num;//定义学号

private String name;//定义名字

public Student(int num,String name){

this.num=num;

this.name=name;

}

public int compareTo(Object o){

Student st=(Student)o;

int result;

result=num>st.num?1:(num==st.num?0:-1);//判断学号是否相同并返回result的值

//如果学号相等，就按姓名排列

/\* if(result==0){

return name.compareTo(st.name);

}\*/

return result;

}

//实现Comparator接口并实现它的抽象方法

public int compare(Object o1,Object o2){

Student st1 =(Student)o1;

Student st2 =(Student)o2;

return st1.name.compareTo(st2.name);//比较姓名是否相同

}

//重写toString()方法，因为如果不重写，打印出来的是16进制代码

public String toString(){

return "num="+num+"; name="+name;

}

public static class StudentComparator implements Comparator{//定义一个静态StudentComparator类并实现Comparator接口

public int compare(Object o1,Object o2){

Student st1 =(Student)o1;

Student st2 =(Student)o2;

int result;

result=st1.num>st2.num?1:(st1.num==st2.num?0:-1);//判断学号是否相同进行排序

if(result==0){//如果学号相等 就进行名字排序

result=st1.name.compareTo(st2.name);

}

return result;

}

}

}

-------------------------------------------------------------------------------

**HashSet与TreeSet集合类的区别：**

HashSet是基于hash算法实现的，性能优于TreeSet。通常使用HashSet，在我们需要对其中元素排序的时候才使用TreeSet。

===============================================================================

## Queue结构的集合

Queue接口

Queue接口与List、Set同一级别，都是继承了Collection接口。LinkedList实现了Queue接口。Queue接口窄化了对LinkedList的方法的访问权限（即在方法中的参数类型如果是Queue时，就完全只能访问Queue接口所定义的方法了，而不能直接访问 LinkedList的非Queue的方法），以使得只有恰当的方法才可以使用。BlockingQueue继承了Queue接口。

队列是一种数据结构。它有两个基本操作：在队列尾部加人一个元素，和从队列头部移除一个元素。就是说，队列以一种先进先出的方式管理数据，如果你试图向一个已经满了的阻塞队列中添加一个元素或者是从一个空的阻塞队列中移除一个元索，将导致线程阻塞。

在多线程进行合作时，阻塞队列是很有用的工具。工作者线程可以定期地把中间结果存到阻塞队列中而其他工作者线线程把中间结果取出并在将来修改它们。队列会自动平衡负载。如果第一个线程集运行得比第二个慢，则第二个线程集在等待结果时就会阻塞。如果第一个线程集运行得快，那么它将等待第二个线程集赶上来。

|  |  |  |
| --- | --- | --- |
| add | 增加一个元索 | 如果队列已满，则抛出一个IIIegaISlabEepeplian异常 |
| remove | 移除并返回队列头部的元素 | 如果队列为空，则抛出一个NoSuchElementException异常 |
| element | 返回队列头部的元素 | 如果队列为空，则抛出一个NoSuchElementException异常 |
| offer | 添加一个元素并返回true | 如果队列已满，则返回false |
| poll | 移除并返问队列头部的元素 | 如果队列为空，则返回null |
| peek | 返回队列头部的元素 | 如果队列为空，则返回null |
| put | 添加一个元素 | 如果队列满，则阻塞 |
| take | 移除并返回队列头部的元素 | 如果队列为空，则阻塞 |
| **remove、element、offer 、poll、peek 其实是属于Queue接口。** | | |

阻塞队列的操作可以根据它们的响应方式分为以下三类：add、remove和element操作在你试图为一个已满的队列增加元素或从空队列取得元素时 抛出异常。当然，在多线程程序中，队列在任何时间都可能变成满的或空的，所以你可能想使用offer、poll、peek方法。这些方法在无法完成任务时 只是给出一个出错示而不会抛出异常。

**注意：**poll和peek方法出错进返回null。因此，向队列中插入null值是不合法的。

**offer，add区别：**

一些队列有大小限制，因此如果想在一个满的队列中加入一个新项，多出的项就会被拒绝。这时新的 offer 方法就可以起作用了。它不是对调用 add() 方法抛出一个 unchecked 异常，而只是得到由 offer() 返回的 false。

**poll，remove区别：**

remove()和poll()方法都是从队列中删除第一个元素（head）。remove()的行为与Collection 接口的版本相似，但是新的 poll()方法在用空集合调用时不是抛出异常，只是返回null。因此新的方法更适合容易出现异常条件的情况。

**peek，element区别：**

element()和peek()用于在队列的头部查询元素。与 remove()方法类似，在队列为空时，element()抛出一个异常，而peek()返回null。

**五个队列所提供的各有不同：**

\* ArrayBlockingQueue ：

一个由数组支持的有界队列。基于数组的阻塞循环队列，先进先出原则对元素进行排序。

\* LinkedBlockingQueue：

一个由链接节点支持的可选有界队列。基于链表的队列，先进先出排序元素。

\* PriorityBlockingQueue：

一个由优先级堆支持的无界优先级队列。PriorityBlockingQueue是对PriorityQueue的再次包装，是基于堆数据结构的，而PriorityQueue是没有容量限制的，与ArrayList一样，所以在优先阻塞队列上put时是不会受阻的。但是由于资源被耗尽，所以试图执行添加操作可能会导致OutOfMemoryError），但是如果队列为空，那么取元素的操作take就会阻塞，所以它的检索操作take是受阻的。另外，往入该队列中的元素要具有比较能力。

\* DelayQueue：

一个由优先级堆支持的、基于时间的调度队列。（基于PriorityQueue来实现的）是一个存放Delayed 元素的无界阻塞队列，只有在延迟期满时才能从中提取元素。该队列的头部是延迟期满后保存时间最长的 Delayed 元素。如果延迟都还没有期满，则队列没有头部，并且poll将返回null。当一个元素的 getDelay(TimeUnit.NANOSECONDS) 方法返回一个小于或等于零的值时，则出现期满，poll就以移除这个元素了。此队列不允许使用null元素。

\* SynchronousQueue：

一个利用 BlockingQueue接口的简单聚集（rendezvous）机制。

SynchronousQueue 类是最简单的。它没有内部容量。它就像线程之间的手递手机制。在队列中加入一个元素的生产者会等待另一个线程的消费者。当这个消费者出现时，这个元素就直接在消费者和生产者之间传递，永远不会加入到阻塞队列中。

**注意：Queue队列是不能直接实例化的。**

原先在java编程中，Queue的实现都是用LinkedList

如：Queue qe=new LinkedList();

**注意：**此实现不是同步的。如果多个线程同时访问一个链接列表，而其中至少一个线程结构上修改了该列表，则它必须保质外部同步。(结构修改指添加或删除一个或多个元素的任何操作；仅设置元素的值不是结构修改。)这一般通过对自然封装该列表的对象进行同步操作来完成。

JDK1.6中：

接口Queue<E>

类型参数：E - collection 中所保存元素的类型。

所有超级接口：Collection<E>, Iterable<E>

所有已知子接口：BlockingDeque<E>, BlockingQueue<E>, Deque<E>

所有已知实现类：AbstractQueue, ArrayBlockingQueue, ArrayDeque,

ConcurrentLinkedQueue, DelayQueue, LinkedBlockingDeque, LinkedBlockingQueue, LinkedList, PriorityBlockingQueue, PriorityQueue, SynchronousQueue

JDK1.7中：

接口Queue<E>

类型参数：E - collection 中所保存元素的类型。

所有超级接口：Collection<E>, Iterable<E>

所有已知子接口：BlockingDeque<E>, BlockingQueue<E>, Deque<E>, TransferQueue<E>

所有已知实现类：AbstractQueue, ArrayBlockingQueue, ArrayDeque,

ConcurrentLinkedQueue, ConcurrentLinkedQueue, DelayQueue, LinkedBlockingDeque,

LinkedBlockingQueue, LinkedList, LinkedTransferQueue, PriorityBlockingQueue,

PriorityQueue, SynchronousQueue

===============================================================================

**Java中的List/Set和Map的区别**

List按对象进入的顺序保存对象，不做排序和编辑操作。

Set对每个对象只接受一次，并使用自己内部的排序方法(通常，你只关心某个元素是否属于Set而不关心它的顺序--否则使用List)。

Map同样对每个元素保存一份，但这是基于"键"(key)的，Map也有内置的排序，因而不关心元素添加的顺序。

如果添加元素的顺序对程序设计很重要，应该使用LinkedHashSet或者LinkedHashMap。

**List的功能方法**

实际上有两种List：一种是基本的ArrayList其优点在于随机访问元素，另一种是更强大的LinkedList它并不是为快速随机访问设计的，而是具有一套更通用的方法。

List：次序是List最重要的特点：它保证维护元素特定的顺序。List为Collection添加了许多方法，使得能够向List中间插入与移除元素(这只推荐LinkedList使用)一个List可以生成Listlterator，使用它可以从两个方向遍历List，也可以从List中间插入和移除元素。

ArrayList：由数组实现的List。允许对元素进行快速随机访问，但是向List中间插入与移除元素的速率很慢。Listlterator只应该用来由后向前遍历ArrayList。而不是用来插入和移除元素。因为那比LinkedList开销要大很多。

LinkedList：对顺序访问进行了优化，向List中间插入与删除的开销并不大。随机访问则相对较慢。(使用ArrayList代替)还具有下列方法：addFirst()，addLast()，getFirst()，getLast()，removeFirst()和removeLast()这些方法(没有在任何接口或基类中定义过)使得LinkedList可以当作堆栈、队列和双向队列使用。

**Set的功能方法**

Set具有与Collection完全一样的接口，因此没有任何额外的功能，不象前面有两个不同的List。实际上Set就是Collection，只是行为不同。(这是继承与多态思想的典型应用：表现不同的行为。)Set不保存重复的元素(至于如何判断元素相同则较为负责)

Set：存入Set的每个元素都必须是唯一的，因为Set不保存重复元素。加入Set的元素必需定义equals()方法以确保对象的唯一性。Set与Collection有完全一样的接口。Set接口不保证维护元素的次序。

HashSet：为快速查找设计的Set。存入HashSet的对象必须定义hashCode()。

TreeSet：保存次序的Set，底层为树结构。使用它可以从Set中提取有序的序列。

LinkedHashSet：具有HashSet的查询速度，且内部使用链表维护元素的顺序(插入的次序)。于是在使用迭代器遍历Set时，结果会按元素插入的次序显示。

**Map的功能方法**

方法put(Object key,Object value)添加一个"值"(想要得东西)和与"值"相关的"键"(key)(使用它来查找)。方法get(Object key)返回与给定"键"相关联的"值"。可以用containsKey()和containsValue()测试Map中是否包含某个"键"或"值"。标准的java类库中包含了几种不同的Map：HashMap，TreeMap，LinkedHashMap，WeakHashMap，ldentityHashMap。它们都有同样的基本接口Map，但是行为、效率、排序策略、保存对象的生命周期和判定"键"等价的策略等各不相同。

执行效率是Map的一个大问题。看看get()要做哪些事，就会明白为什么在ArrayList中搜索"键"是相当慢的。这正是HashMap提高速度的地方。HashMap使用了特殊的值，称为"散列码"(hash code)，来取代对键的缓慢搜索。"散列码"是"相对唯一"用以代表对象的int值，它是通过将该对象的某些信息进行转换而生成的。所有java对象都能产生散列码，因为hashCode()是定义在基类Object中的方法。

HashMap就是使用对象的hashCode()进行快速查询的。此方法能够显著提高性能。

Map：维护"键值对"的关联性，使你可通过"键"查找"值"

HashMap：Map基于散列表的实现。插入和查询"键值对"的开销是固定的。可以通过构造器设置容量capacity和负载因子load factor，以调整容器的性能。

LinkedHashMap：类似于HashMap，但是迭代遍历它时，取得"键值对"的顺序是其插入次序，或者是最近最少使(LRU)的次序。只能HashMap慢一点。而在迭代访问时发而更快，因为它使用键表维护内部次序。

TreeMap：基于红黑树数据结果的实现。查看"键"或"键值对"时，它们会被排序(次序由Comparabel或Comparator决定)。TreeMap的特点在于，你得到的结果是经过排序的。TreeMap是唯一的带有subMap()方法的Map，它可以返回一个子树。

WeakHashMap:旨键(weak key)Map，Map中使用的对象也被允许释放：这是为解决特殊问题设计的。如果没有map之外的引用指向某个"键"，则此"键"可以被垃圾收集器回收。

ldentifyHashMap：使用==代替equals()对"键"作比较的hash map。专为解决特殊问题而设计。

-------------------------------------------------------------------------------

# Java中的Iterator(迭代器)的用法

java.util包中包含了一系列重要的集合类，集合类的根接口Collection。

Collection接口是所有集合类的根类型。它的一个主要的接口方法是：

boolean add(Object c)添加数据

add()方法将添加一个新元素。注意这个方法会返回一个boolean，但是返回值不是表示添加成功与否。Collection规定：如果一个集合拒绝添加这个元素，无论任何原因，都必须抛出异常。这个返回值表示的意义是add()方法执行后，集合的内容是否改变了(就是元素有无数量，位置等变化)，这是由具体类实现的。即：如果方法出错，总会抛出异常；返回值仅仅表示该方法执行后这个Collection的内容有无变化。

类似还有：

boolean addall(Collection c);添加所有数据

boolean remove(Object o);删除数据

boolean removeall(Collection c);删除所有数据

boolean remainall(Collection c);保持所有数据

Object[]toArray()方法很简单，把集合转换成数组返回。Object[]toArray(Object[] a)方法就有点复杂了，首先，返回的Object[]仍然是把集合的所有元素变成数组，但是类型和参数a的类型是相同的。

如：String[] o=(String)c.toArray(new String[0]);

得到的o实际类型是String[]数组。

其次，如果参数a的大小装不下集合的所有元素，返回的将是一个新的数组。如果参数a的大小能装下集合的所有元素，则返回的还是a，但a的内容用集合的元素来填充。尤其要注意的是，如果a的大小比集合元素的个数还多，a后面的部分全部被置为null(空)。

最后一个最重要的方法是Iterator()，返回一个Iterator(迭代子)，用于遍历集合的所有元素。

## 用Iterator模式实现遍历集合

Iterator模式 是用于遍历集合类的标准访问方法。它可以把访问逻辑从不同类型的集合类中抽象出来，从而避免向客户端暴露集合的内部结构。

例如，如果没有使用Iterator，遍历一个数组的方法是使用索引：

for(int i=0;i<array.size();i++){..get(i)...}

而访问一个链表(LinkedList)又必须使用while循环：

while((e=e.next())!=null){...e.data()...}

以上两种方法客户端都必须事先知道集合的内部结构，访问代码和集合本身是紧耦合，无法将访问逻辑从集合类和客户端代码中分离出来，每一种集合对应一种遍历方法，客户端代码无法复用。

更恐怖的是，如果以后需要把ArrayList更换为LinkedList，则原来的客户端代码必须全部重写。

为解决以上问题，Iterator模式总是用同一种逻辑来遍历集合：

for(Iterator it=c.iterater();it.hasNext();){ ...}

奥秘在于客户端自身不维护遍历集合的"指针"，所有的内部状态（如当前元素位置，是否有下一个元素）都由Iterator来维护，而这个Iterator由集合类通过工厂方法生成，因此，它知道如何遍历整个集合。

客户端从不直接和集合类打交道，它总是控制Iterator，向它发送"向前"，"向后"，"取当前元素"的命令，就可以间接遍历整个集合。

首先看看java.util.Iterator接口的定义：

public interface Iterator {

　 boolean hasNext();

　 Object next();

　 void remove();

}

依赖前两个方法就能完成遍历，典型的代码如下：

for(Iterator it=c.iterator();it.hasNext();){

　 Object o=it.next();

　 // 对o的操作...

}

在JDK1.5中，还对上面的代码在语法上作了简化：

// Type是具体的类型，如String。

for(Type t:c){

// 对t的操作...

}

每一种集合类返回的Iterator具体类型可能不同，Array可能返回ArrayIterator，Set可能返回SetIterator，Tree 可能返回TreeIterator，但是它们都实现了Iterator接口，因此，客户端不关心到底是哪种Iterator，它只需要获得这个 Iterator接口即可，这就是面向对象的威力。

要确保遍历过程顺利完成，必须保证遍历过程中不更改集合的内容（Iterator的remove()方法除外），因此，确保遍历可靠的原则是只在一个线程中使用这个集合，或者在多线程中对遍历代码进行同步。

Iterator示例:

Collection c=new ArrayList();

c.add("abc");

c.add("xyz");

for(Iterator it=c.iterator();it.hasNext();){

String s=(String)it.next();

System.out.println(s);

}

如果你把第一行代码的ArrayList换成LinkedList或Vector，剩下的代码不用改动一行就能编译，而且功能不变，这就是针对抽象编程的原则：对具体类的依赖性最小。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 泛型--基本概念

泛型的基本概念

泛型是java se1.5的新特性，泛型的本质是参数化类型，也就是说所操作的数据类型被指定为一个参数。这种参数类型可以用在类、接口和方法的创建中，分别称为泛型类、泛型接口、泛型方法。

java语言引入泛型的好处是安全简单。

在java se1.5之前，没有泛型的情况下，通过对类型Object的引用来实现参数的“任意化”，“任意化”带来的缺点是要做显式的强制类型转换，而这种转换是要求开发者对实际参数类型可以预知的情况下进行的，对于强制类型转换错误的情况，编译器可能不提示错误，在运行的时候才出现异常，这是一个安全隐患。

泛型的好处是在编译的时候**检查类型安全**，并且所有的**强制转换都是自动和隐式**的，**提高代码的重用率**。

//泛型的必要性:示例：

import java.util.\*;

public class Generics {

public static void main(String[] args) {

ArrayList<Dog> al=new ArrayList<Dog>();//<Dog>即泛型的指定参数，提高安全性

ArrayList bl=new ArrayList();

//创建一只狗

Dog dog1=new Dog();

//放入到集合中

al.add(dog1);

//取出

Dog temp=al.get(0);//引用泛型后即可不用强转，Dog temp=(Dog)al.get(0);

Cat temp1=(Cat)bl.get(0);

}

}

class Cat{

private String color;

private int age;

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

class Dog{

private String name;

private int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

-------------------------------------------------------------------------------

# Java-->反射机制[Demo147.java]

//泛型的必要性

import java.util.\*;

import java.lang.reflect.Method;//引入Java反射方法类

public class Demo147 {

public static void main(String[] args) {

Gen<String> gen1=new Gen<String>("aa");//<>可以放任意类型

Gen<Bird> gen2=new Gen<Bird>(new Bird());//<>也可以放入定义好的类

gen1.showTypeName();

gen2.showTypeName();

}

}

//定义一个Bird

class Bird{

public void test1(){

System.out.println("aa");

}

public void count(int a,int b){

System.out.println(a+b);

}

}

//定义一个类

class Gen<T>{//T传入什么类型，Gen类就是什么什么类型

private T o;

//构造函数

public Gen(T a){

o=a;

}

//得到T的类型名称

public void showTypeName(){

System.out.println("类型是："+o.getClass().getName());

//通过反射机制，我们可以得到T这个类型的很多信息

//得到成员函数名

Method [] m=o.getClass().getDeclaredMethods();

//打印

for(int i=0;i<m.length;i++){

System.out.println(m[i].getName());//打印函数名列表

}

}

}

-------------------------------------------------------------------------------

## 泛型的优点

使用**泛型有下面几个优点：**

1、类型安全

2、向后兼容

3、层次清晰

4、性能较高，用GJ(泛型JAVA)编写的代码可以为java编译器和虚拟机带来更多的类型信息，这些信息对java程序做进一步优化提供条件。

===============================================================================

# 异常处理--基本概念

当出现程序无法控制的外部环境问题(用户提供的文件不存在，文件内容损坏，网络不可用...)时，JAVA就会用异常对象来描述。

**java中用2种方法处理异常**：

1、在发生异常的地方直接处理；

2、将异常抛给调用者，让调用者处理。

## 异常分类

1、检查性异常：java.lang.Exception

2、运行期异常：java.lang.RuntimeException

3、错误：java.lang.Error

顶层是java.lang.Throwable类，检查性异常、运行期异常、错误都是这个类的子孙类，java.lang.Exception和java.lang.Error继承自java.lang.Throwable，而java.lang.RuntimeException继承自java.lang.Exception

**1、检查性异常：**

程序正确，但因为外在的环境条件不满足引发。例如：用户错误及I/O问题--程序试图打开一个并不存在的远程Socket端口，或者是打开不存在的文件时。这不是程序本身的逻辑错误，而很可能是远程机器名字错误(用户拼写错误)，对商用软件系统，程序开发者必须考虑并处理这个问题。java编译器强制要求处理这类异常，如果不捕获这类异常，程序将不能被编译。

**2、运行期异常：**

这意味着程序存在bug，如数组越界、0被除、入参不满足规范...这类异常需要更改程序来避免，java编译器强制要求处理这类异常。

**3、错误：**

一般很少见，也很难通过程序解决，它可能源于程序的bug，但一般更可能源于环境问题，如内存耗尽。错误在程序中无需处理，而由运行环境处理。

//异常示例

import java.io.\*;

import java.net.\*;

public class Demo148 {

public static void main(String[] args) {

//检查异常

//1、打开不存在的文件

//FileReader fr=new FileReader("d:\\aa.txt");

//2、连接一个192.168.12.12 ip的端口号4567

//Socket s=new Socket("192.168.1.1",80);

//运行异常

//除0导致异常

//int a=4/0;

//数组越界异常

//int arr[]={1,2,3};

//System.out.println(arr[1234]);

}

}

-------------------------------------------------------------------------------

**异常处理**

**1、try...catch**

程序运行产生异常时，将从异常发生点中断程序并向外抛出异常信息。

int x=(int)(Math.random()\*5);

int y=(int)(Math.random()\*10);

int[] z=new int[5];

try{

System.out.println("y/x="+(y/x));

System.out.println("y="+y+"z[y]="+z[y]);

}

catch(ArithmeticException exc1){//分步捕获算术运算异常信息

System.out.println("算术运算异常:"+exc1.getMessage());

}

catch(ArrayIndexOutOfBoundsException exc2){//分步捕获数据越界异常信息

System.out.println("数据越界异常:"+exc2.getMessage());

}

------------------------------------------------------------------------------

**异常处理**

**2、finally**

如果把finally块置try...catch...语句后，**finally块一般都会得到执行**，它相当于一个万能的保险，即使前面的try块发生异常，而又没有对应异常的catch块，finally块将马上执行。

以下情形，**finally块将不会被执行：**

1、finally块中发生了异常；

2、程序所在的线程死亡；

3、在前面的代码中用了System.exit()；

4、关闭CPU

//异常示例[Demo148.java]

import java.io.\*;

import java.net.\*;

public class Demo148 {

public static void main(String[] args) {

FileReader fr=null;

//检查异常

//1、打开不存在的文件

//FileReader fr=new FileReader("d:\\aa.txt");

try {//使用try{}catch(Exception e){}将可能出错的程序放入到里面，当出错时会有相应提示，便于解决bug

//在出现异常的地方就终止执行代码，然后直接进入到catch语句

//如里有多个catch语句，则进入匹配异常的catch语句输入出信息

fr=new FileReader("d:\\aa.txt");

//System.exit(-1);//使用System.exit()后finally语句块不再执行

Socket s=new Socket("192.168.1.1",21);

} catch (FileNotFoundException e) {//catch(Exception e)捕获所有错误信息，为了方便一般使用此方法来捕获所有错误信息

// 把异常的信息输出，利于排除bug

//e.getMessage();

System.out.println("文件不存在："+e.getMessage());//.getMessage()不如.printStackTrace()

//e.printStackTrace();//输出bug信息

//处理

} catch (IOException e2){//UnknownHostException

e2.printStackTrace();

} finally {

//try..catch..语句块中不管出没出现异常，一般都会执行finally语句块

//一般说，把需要关闭的资源。如[文件]、[链接]、[内存]...

System.out.println("测试进入finally语句块");

if(fr!=null){

try {

fr.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

System.out.println("OK1");

//2、连接一个192.168.12.12 ip的端口号4567

//Socket s=new Socket("192.168.1.1",80);

//运行异常

//除0导致异常

//int a=4/0;

//数组越界异常

//int arr[]={1,2,3};

//System.out.println(arr[1234]);

}

}

-----------------------------------------------------------------------------

**异常处理**

将异常抛给调用者，让调用者处理异常**thorws**

//抛出异常[Demo149.java]

import java.io.\*;

public class Demo149 {

public static void main(String[] args) {

Father father=new Father();

father.test1();

}

}

class Father{

private Son son=null;

public Father(){

son=new Son();

}

public void test1(){

System.out.println("1");

try {

son.test2();

} catch (Exception e) {

System.out.println("Father在处理Son的异常");

e.printStackTrace();

}

}

}

class Son{

public void test2() **throws Exception**{//throws Exception抛出程序块的异常，由调用者解决异常

FileReader fr=null;

fr=new FileReader("d:\\aaa.txt");

}

}

-------------------------------------------------------------------------------

**异常处理**

3、多个异常的处理规则

定义多个catch可精确地定位异常。如果为子类的异常定义了特殊的catch块，而父类的异常则放在另外一个catch块中，此时，必需满足以下规则：

子类异常的处理块必需在父类异常处理块的前面，否则会发生编译错误。所以越特殊的异常越在前面处理，越普通的异常越在后面处理。这类似于制订防火墙的规则次序：**较特殊的规则在前，较普通的规则在后。**

自己也可以定义并抛出异常，方法是2步：

创建异常，抛出异常(首先实例化一个异常对象，然后用throw抛出)合在一起就是---throw new IOException("异常说明信息")，将创建异常，抛出异常合在一起的好处是：创建异常时，会包含异常创建处的行信息，异常被捕获时可以通过堆栈迹(Stack Trace)的形式报告这些信息。如果在同一行代码创建和抛出异常时，对于程序的调试将非常有用。所以，throw new XXX()已经成为一个标准的异常抛出范式。在定义一个方法时，方法块中调用的方法可能会抛出异常，可用上面的throw new XXX()处理，如果不处理，那么必需在方法定义时，用throws声明这个方法全抛出的异常。

**对异常的处理，有一条行之有效的默认规则：向上抛出**----被调用类在运行过程中对遇到的异常一概不作处理，而是直接向上抛出，一直到最上层的调用类，调用类根据应用系统的需求和特定的异常处理规则进行处理，如向控制台输出异常堆栈信息，打印在日志文件中。用一句形象的话来说，就是谁使用，谁(最上层的调用类)处理。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**作业：**

1、跳水比赛，8个评委打分。运动员的成绩是8个成绩去掉一个最高分，去掉一个最低分，剩下的6个分数的平均分就是最后得分。使用一维数组实现打分功能。

2、请把打最高分的评委和最低分的评委打出来。

//使用ArrayList写的[Demo150.java]

import java.util.\*;

public class Demo150 {

public static void main(String[] args) {

Referee rf=new Referee();

try {

while(true){

System.out.println("跳水评分系统");

System.out.println("1、裁判打分");

System.out.println("2、查看得分");

System.out.println("3、查看最高分与最低分");

System.out.println("0、退出");

Scanner sr=new Scanner(System.in);

int num=sr.nextInt();

//裁判打分

if(num==1){

System.out.print("请输入裁判人数:");

int j=sr.nextInt();

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

System.out.print("请输入第"+(i+1)+"位裁判名字:");

String name=sr.next();

System.out.print(name+"裁判请输入评分:");

float fraction=sr.nextFloat();

Dive dive=new Dive(name,fraction);

rf.addDive(dive);

System.out.println("第"+(i+1)+"位裁判:"+name+"打分为:"+fraction);

}

}

//查看得分

else if(num==2){

rf.View();

}

//查看最高分和最低分

else if(num==3){

rf.Cal();

}

//退出系统

else if(num==0){

System.out.println("已正常退出系统");

System.exit(0);

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

//裁判类

class Referee{

private ArrayList<Dive> al=null;

//构造函数，初始化成员变量

public Referee(){

al=new ArrayList<Dive>();

}

//加入裁判

public void addDive(Dive dive){

al.add(dive);

}

//查看选手得分

public void View(){

float allnum=0f,pjnum=0f;

//遍历

for(int i=0;i<al.size()-1;i++){

for(int j=1;j<al.size()-i;j++){

Dive dive1=al.get(j-1);

Dive dive2=al.get(j);

//排序

if(dive1.getFraction()<dive2.getFraction()){

al.set(j, dive1);

al.set(j-1, dive2);

}

}

}

//去掉最高分和最低分，得出选手得分

for(int i=1;i<al.size()-1;i++){

Dive dive=al.get(i);

allnum=dive.getFraction()+allnum;

}

pjnum=allnum/(al.size()-2);

System.out.println("跳水选手得分为："+pjnum);

}

//找出最高分和最低分

public void Cal(){

//遍历

for(int i=0;i<al.size()-1;i++){

for(int j=1;j<al.size()-i;j++){

Dive dive1=al.get(j-1);

Dive dive2=al.get(j);

//排序

if(dive1.getFraction()<dive2.getFraction()){

al.set(j, dive1);

al.set(j-1, dive2);

}

}

}

//找出最高分与最低分

for(int i=0;i<al.size();i++){

if(i==0){

Dive dive=al.get(i);

System.out.println("打分最高的裁判是："+dive.getName()+"\t分数为："+dive.getFraction());

}else if(i==al.size()-1){

Dive dive=al.get(i);

System.out.println("打分最低的裁判是："+dive.getName()+"\t分数为："+dive.getFraction());

}

}

}

}

//裁判信息类

class Dive{

private String name;

private float fraction;

public Dive(String name,float fraction){

this.name=name;

this.fraction=fraction;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getFraction() {

return fraction;

}

public void setFraction(float fraction) {

this.fraction = fraction;

}

}

-------------------------------------------------------------------------------

/\*1、跳水比赛，8个评委打分。运动员的成绩是8个成绩去掉一个最高分，去掉一个最低分，剩下的6个分数的平均分就是最后得分。使用一维数组实现打分功能。

2、请把打最高分的评委和最低分的评委打出来。(使用一维数组)[Demo150\_1]

\*/

import java.util.\*;

public class Demo150\_1 {

public static void main(String[] args) {

Judge judge=new Judge();

System.out.println("得分:"+judge.lastFen());

}

}

class Judge{

//定义一个可以存放8个float类型的数组

float fens[]=null;

int size=8;

//构造函数

public Judge(){

fens=new float[size];

Scanner sr=new Scanner(System.in);

//初始化

try {

for(int i=0;i<fens.length;i++){

System.out.println("请输入第"+(i+1)+"个裁判的成绩：");

fens[i]=sr.nextFloat();

}

} catch (Exception e) {

System.out.println("输入的不是float类型!");

e.printStackTrace();

}finally{

try {

sr.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

//得到最后评分

public float lastFen(){

//3、得到最后分数

float allFen=0;

int minIndex=this.getminFenIndex();

int maxIndex=this.getMaxFenIndex();

//遍历

for(int i=0;i<fens.length;i++){

if(i!=minIndex&&i!=maxIndex){

allFen+=fens[i];

}

}

return allFen/(fens.length-2);

}

//1、去掉最低分(目地就是找到最低分的下标)

public int getminFenIndex(){

//选择法

//认为第一个是最低分

float minFen=fens[0];

int minIndex=0;

for(int i=1;i<fens.length;i++){

if(minFen>fens[i]){

//当minFen大于fens[i]修改最低分minFen

minFen=fens[i];

minIndex=i;

}

}

return minIndex;

}

//2、去掉最高分(目地就是找到最高分的下标)

public int getMaxFenIndex(){

//选择法

//认为第一个是最低分

float maxFen=fens[0];

int maxIndex=0;

for(int i=1;i<fens.length;i++){

if(maxFen<fens[i]){

//当maxFen大于fens[i]修改最低分maxFen

maxFen=fens[i];

maxIndex=i;

}

}

return maxIndex;

}

}

===============================================================================

2、实现一个简单学生成绩管理系统

定义一个数组，用户输入10个同学的成绩，数组下标即代表学生学号

要求：

1、输入学号，打印学生成绩

2、输入成绩，打印学生学号

3、统计各个分数阶段学生人数(不及格，差，中，良，优)

4、输入学号，实现删除学生成绩功能

//ArrayList编写[Demo151.java]

import java.util.\*;

public class Demo151 {

public static void main(String[] args) {

Result rs=new Result();

while(true){

System.out.println("学生成绩管理系统");

System.out.println("1、录入学生成绩");

System.out.println("2、通过学号查成绩");

System.out.println("3、通过成绩查学号");

System.out.println("4、统计成绩分类人数");

System.out.println("5、通过学号删除成绩");

System.out.println("0、退出系统");

System.out.println("请选择操作：");

Scanner sr=new Scanner(System.in);

int num=sr.nextInt();

if(num==1){

System.out.println("需要录入多少学生成绩：");

int j=sr.nextInt();

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

System.out.print("请输入第"+(i+1)+"位学生的成绩：");

float Score=sr.nextFloat();

Student student=new Student(Score);

rs.addResult(student);

}

}else if(num==2){

rs.Id();

}else if(num==3){

rs.Score();

}else if(num==4){

rs.Count();

}else if(num==5){

rs.Del();

}else if(num==0){

System.out.println("退出系统");

System.exit(0);

}

}

}

}

class Result{

Scanner sr=new Scanner(System.in);

private ArrayList<Student> al=null;

//构造函数

public Result(){

al=new ArrayList<Student>();

}

//加入成绩

public void addResult(Student student){

al.add(student);

}

//输入入学号，打印学生成绩

public void Id(){

System.out.println("请输入查询学号进行查询：");

int id=sr.nextInt();

for(int i=0;i<al.size();i++){

if(id==i){

Student st=al.get(i-1);

System.out.println("学号为："+id+"成绩是："+st.getScore());

}

}

}

//输入成绩，打印学号

public void Score(){

System.out.println("请输入成绩进行查询：");

float Score=sr.nextFloat();

for(int i=0;i<al.size();i++){

if(Score==al.get(i).getScore()){

System.out.println("成绩是："+Score+"得此成绩的学号为："+(i+1));

}

}

}

//统计各个分数阶段人数(不及格、及格、中、良、优)

public void Count(){

int a=0,b=0,c=0,d=0,e=0,f=0;

for(int i=0;i<al.size();i++){

if(al.get(i).getScore()==100f&&al.get(i).getScore()>=90f){

a++;

}else if(al.get(i).getScore()<90f&&al.get(i).getScore()>=80f){

b++;

}else if(al.get(i).getScore()<80f&&al.get(i).getScore()>=70f){

c++;

}else if(al.get(i).getScore()<70f&&al.get(i).getScore()>=60f){

d++;

}else if(al.get(i).getScore()<60f&&al.get(i).getScore()>0){

e++;

}else if(al.get(i).getScore()==0){

f++;

}

}

System.out.println("成绩优有："+a+"人");

System.out.println("成绩良有："+b+"人");

System.out.println("成绩中有："+c+"人");

System.out.println("成绩差有："+d+"人");

System.out.println("不及格有："+e+"人");

System.out.println("无成绩有："+f+"人");

}

//通过学号删除成绩

public void Del(){

System.out.println("请输入要删除成绩的学号：");

int id=sr.nextInt();

id=id-1;

for(int i=0;i<al.size();i++){

if(id==i){

al.set(i, null);

System.out.println("学号是："+(id+1)+"的成绩已清除");

}

}

}

}

class Student{

private float Score;

public Student(float Score){

this.Score=Score;

}

public float getScore() {

return Score;

}

public void setScore(float score) {

Score = score;

}

}

-------------------------------------------------------------------------------

/\*

定义一个数组，用户输入10个同学的成绩，数组下标即代表学生学号

要求：

1、输入学号，打印学生成绩

2、输入成绩，打印学生学号

3、统计各个分数阶段学生人数(不及格，差，中，良，优)

4、输入学号，实现删除学生成绩功能

\*/

import java.util.\*;

public class Demo151\_1 {

private static Scanner sr;

public static void main(String[] args) {

Student sd=new Student();

System.out.println("学生成绩管理系统");

System.out.println("1、通过学号查成绩");

System.out.println("2、通过成绩查学号");

System.out.println("3、统计各阶段学生人数");

System.out.println("4、通过学号删除成绩");

System.out.println("0、退出系统");

System.out.print("请选择操作：");

sr = new Scanner(System.in);

int num=sr.nextInt();

if(num==1){

sd.studentNo();

}else if(num==2){

sd.studentScore();

}else if(num==3){

sd.People();

}else if(num==4){

sd.Del();

}else if(num==0){

System.out.println("已正常退出系统!");

System.exit(0);

}

}

}

class Student{

Scanner sr=new Scanner(System.in);

//定义数组arr

float arr[]=null;

//数组大小

int size=10;

//构造函数

public Student(){

arr=new float[size];

for(int i=0;i<arr.length;i++){

System.out.println("输入第"+(i+1)+"个学生的成绩：");

arr[i]=sr.nextFloat();

}

}

//输入学号查询成绩

public void studentNo(){

System.out.println("请输入学号：");

int No=sr.nextInt();

System.out.println("学号为"+No+"的学生成绩是："+arr[No-1]);

}

//通过成绩查学号

public void studentScore(){

System.out.println("请输入学生成绩：");

float Score=sr.nextFloat();

for(int i=0;i<arr.length;i++){

if(Score==arr[i]){

System.out.println("学生成绩为："+Score+"的学号为："+(i+1));

}

}

}

//统计各个分数阶段学生人数

public void People(){

int a=0,b=0,c=0,d=0,e=0,f=0;

for(int i=0;i<arr.length;i++){

if(arr[i]==100f&&arr[i]>=90f){

a++;

}else if(arr[i]<90f&&arr[i]>=80f){

b++;

}else if(arr[i]<80f&&arr[i]>=70f){

c++;

}else if(arr[i]<70f&&arr[i]>=60f){

d++;

}else if(arr[i]<60f&&arr[i]>0){

e++;

}else if(arr[i]==0){

f++;

}

}

System.out.println("成绩优有："+a+"人");

System.out.println("成绩良有："+b+"人");

System.out.println("成绩中有："+c+"人");

System.out.println("成绩差有："+d+"人");

System.out.println("不及格有："+e+"人");

System.out.println("无成绩有："+f+"人");

}

//删除学生成绩

public void Del(){

System.out.println("请输入要删除成绩的学号：");

int No=sr.nextInt();

System.out.println("学号为"+No+"的学生原成绩为："+arr[No]);

System.out.println("学号为"+No+"的学生成绩已清"+(arr[No]=0));

}

}

-------------------------------------------------------------------------------

3、三个同学考试，共考三门课：语文、数学、英语。使用二维整数数组存放三个同学的学号和所有考试成绩。

学号 语文 数学 英语

1002 78 92 76

1003 67 88 80

1007 90 95 80

//程序示例[Demo152.java]

public class Demo152 {

public static void main(String[] args) {

//定义一个二维整数数组并初始化赋值

int stus[][]={{1002,78,92,76},{1003,67,88,80},{1007,90,95,80}};

//也可以下面的方法(不建议使用，效率太慢)

int stus2[][]=new int[3][4];

stus2[0][0]=1002;

stus2[1][0]=1003;

stus2[2][0]=1007;

stus2[0][1]=78;

stus2[1][1]=67;

stus2[2][1]=90;

stus2[0][2]=92;

stus2[1][2]=88;

stus2[2][2]=95;

stus2[0][3]=76;

stus2[1][3]=80;

stus2[2][3]=80;

for(int i=0;i<3;i++){

for(int j=0;j<4;j++){

System.out.print(stus[i][j]+"\t");

}

System.out.println();

}

System.out.println();

for(int i=0;i<3;i++){

for(int j=0;j<4;j++){

System.out.print(stus2[i][j]+"\t");

}

System.out.println();

}

}

}

-------------------------------------------------------------------------------

4、1、跳水比赛，8个评委打分。运动员的成绩是8个成绩去掉一个最高分，去掉一个最低分，剩下的6个分数的平均分就是最后得分。使用一维数组实现打分功能。

2、请把打最高分的评委和最低分的评委打出来。

3、找出最佳评委和最差评委。最佳评委就是打分和最后得分最接近的评委。最差评委就是打分和最后得分相差最大的。[Demo153.java]

import java.util.\*;

public class Demo153 {

public static void main(String[] args) {

Judge judge=new Judge();

System.out.println("得分:"+judge.lastFen());

System.out.println("打分最差的是裁判是："+(judge.getWorst()+1));

System.out.println("打分最好的是裁判是："+(judge.getBest()+1));

}

}

class Judge{

//定义一个可以存放8个float类型的数组

float fens[]=null;

int size=3;//定义几个裁判

//构造函数

public Judge(){

fens=new float[size];

Scanner sr=new Scanner(System.in);

//初始化

try {

for(int i=0;i<fens.length;i++){

System.out.println("请输入第"+(i+1)+"个裁判的成绩：");

fens[i]=sr.nextFloat();

}

} catch (Exception e) {

System.out.println("输入的不是float类型!");

e.printStackTrace();

}finally{

try {

sr.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

//得到打分最差评委

public int getWorst(){

float lastFen=this.lastFen();

float tempCai=0f;

//我认为第一个评委打分是最差的

int worstIndex=0;

float cai=Math.abs(fens[0]-lastFen);//使用Math来返回一个数学绝对值

for(int i=1;i<fens.length;i++){

tempCai=Math.abs(fens[i]-lastFen);

if(cai<tempCai){

worstIndex=i;

cai=tempCai;

}

}

return worstIndex;

}

//得到打分最好评委

public int getBest(){

float lastFen=this.lastFen();

float tempHao=0f;

//我认为第一个评委打分是最好的

int bestIndex=0;

float hao=Math.abs(fens[0]-lastFen);//使用Math来返回一个数学绝对值

for(int i=1;i<fens.length;i++){

tempHao=Math.abs(fens[i]-lastFen);

if(hao>tempHao){

bestIndex=i;

hao=tempHao;

}

}

return bestIndex;

}

//得到最后评分

public float lastFen(){

//3、得到最后分数

float allFen=0;

int minIndex=this.getminFenIndex();

int maxIndex=this.getMaxFenIndex();

//遍历

for(int i=0;i<fens.length;i++){

if(i!=minIndex&&i!=maxIndex){

allFen+=fens[i];

}

}

return allFen/(fens.length-2);

}

//1、去掉最低分(目地就是找到最低分的下标)

public int getminFenIndex(){

//选择法

//认为第一个是最低分

float minFen=fens[0];

int minIndex=0;

for(int i=1;i<fens.length;i++){

if(minFen>fens[i]){

//当minFen大于fens[i]修改最低分minFen

minFen=fens[i];

minIndex=i;

}

}

return minIndex;

}

//2、去掉最高分(目地就是找到最高分的下标)

public int getMaxFenIndex(){

//选择法

//认为第一个是最低分

float maxFen=fens[0];

int maxIndex=0;

for(int i=1;i<fens.length;i++){

if(maxFen<fens[i]){

//当maxFen大于fens[i]修改最低分maxFen

maxFen=fens[i];

maxIndex=i;

}

}

return maxIndex;

}

}

-------------------------------------------------------------------------------

# Java基础测试题

关键字：java

1、下面的语句哪行在编译时没有警告和编译错误

a)float f=1.3;//1.3是double类型，赋给float需要加1.3f

b)char c="a";//char类型赋值是不可以用""只能用''

c)byte b=257;//byte类型赋值不能超过127，否则越界报错

d)boolean b=null;//boolean类型不能赋空值，只能赋true或false

e)int i=10;//正确，无报错

2、下面的代码编译后会出现什么问题

public class MyClass{

public static void main(String arguments[]){

amethod(arguments);

}

public void amethod(String[] arguments){

System.out.println(arguments);

System.out.println(arguments[1]);

}

}

a)错误，不能静态引用amethod方法 //正确

b)错误，main方法不正确

c)错误，数组必须包含参数

d)amethod必须声明为static类型

3、下面的哪一组代码会出现编译错误

a) import java.awt.\*; //会出现编译错误，打包命令package应在首行

package Mypackage;

class Myclass{}

b) package MyPackage;

import java.awt.\*;

class MyClass{}

c) /\*This is a comment\*/

package MyPackage;

import java.awt.\*;

class MyClass{}

4、byte类型的大小是

a)-128～127 //正确

b)-28-1～28

c)-255～256

d)不同的操作系统java虚拟机分配不同的大小

5、下面的代码在输入下面的命令行后输出什么内容

命令行：java myprog good morning

代码：

public class myprog{

public static void main(String argv[]){

System.out.println(argv[2]);//从命令行java myprog后传入两个数

}

}

a)myprog

b)good

c)morning

d)Exception raised:"java.lang.ArrayIndexOutOfBoundsException:2"//正确

6、下面哪个不是java的关键字或者保留字

a)if

b)then //不是java的关键字或者保留字

c)goto

d)while

e)case

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Java关键字列表(依字母排序共51组)

abstract,assert,boolean,break,byte,case,catch,char,class,const,continue,default,do,double,else,enum,extends,final,finally,float,for,if,implements,import,instanceof,int,interface,long,native,new,package,private,protected,public,return,short,static,strictfp,super,switch,synchronized,this,throw,throws,transient,try,void,volatile,while

Java保留字列表(依字母排序共14组：Java保留字是指现有Java版本尚未使用但以后版本可能会作为关键字使用)

byValue,cast,false,future,generic,inner,operator,outer,rest,true,var,goto,const,null

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7、下面哪些是合法的标志符(多选)

a)2variable //不能以数字打头

b)variable2 //正确

c)\_whatavariable//正确

d)\_3\_ //正确

e)$anothervar //正确

f)#myvar //不能存有#号

8、试图编译运行下面的代码会发生什么情况

public class MyClass{

static int i;**//static静态变量可以不给初值，但是final必需要赋值**

public static void main(String argv[]){

System.out.println(i);

}

}

a)错误，变量i没有初始化

b)null

c)1

d)0 //正确

9、试图编译运行下面的代码会发生什么情况

public class Q{

public static void main(String argv[]){

int anar[]=new int[]{1,2,3};

System.out.println(anar[1]);

}

}

a)1

b)Error anar被引用前没有初始化

c)2 //正确

d)Error:数组大小没有定义

10、试图编译运行下面的代码会发生什么情况

public class Q{

public static void main(String argv[]){

int anar[]=new int[5];

System.out.println(anar[0]);

}

}

a)Error:anar is referenced before it is initialized

b)null

c)0 **//正确，数组没有被赋初值默认为0，不能为null空**

d)5

11、试图编译运行下面的代码会发生什么情况

abstract class MineBase{

abstract void amethod();

static int i;

}

public class Mine extends MineBase{

public static void main(String argv[]){

int[] ar=new int[5];

for(i=0;i<ar.length;i++){

System.out.println(ar[i]);

}

}

}

a)5个0的序列会被输出

b)Error:ar在使用前要初始化

c)Error Mine必须被声明为abstract //正确

d)Error数组越界

12、试图编译运行下面的代码会输出什么样的结果

int i=1;

switch(i){

case 0:

System.out.println("zero");

break;

case 1:

System.out.println("one");

case 2:

System.out.println("two");

default:

System.out.println("default");

}

a)one

b)one,default

c)one,two,default //正确

d)default

13、试图编译运行下面的代码会输出什么样的结果

int i=9;

switch(i){

default:

System.out.println("default");

case 0:

System.out.println("zero");

break;

case 1:

System.out.println("one");

case 2:

System.out.println("two");

}

a)default

b)default,zero //正确

c)error default没有定义

d)无输出

14、下面的哪些组代码没有编译错误(多选)

a) int i=0;

if(i){ //错误if中没有判断

System.out.println("Hello");

}

b) boolean b=true;

boolean b2=true;

if(b--b2){

System.out.println("So true"); //正确

}

c) int i=1;

int j=2;

if(i==1||j==2)

System.out.println("OK"); //正确

d) int i=1;

int j=2;

if(i==1&|j==2) //错误，&|非法。**单个&或|则正确**

System.out.println("OK");

15、如果在当前目录下不存在Hello.txt文件，试图编译和运行下面代码会输什么

import java.io.\*;

public class Mine{

public static void main(String argv[]){

Mine m=new Mine();

System.out.println(m.amethod());

}

public int amethod(){

try{

FileInputStream dis=new FileInputStream("Hello.txt");

}catch(FileNotFoundException fne){

System.out.println("No such file found");

return -1;**//虚拟机会记录，但要运行后finally语句块后才返回**

}catch(IOException ioe){

}finally{

System.out.println("Doing finally");

}

return 0;

}

}

a)No such file found

b)No such file found,-1

c)No such file found,Doing finally,-1 //正确

d)0

16、在下面的注释处插入哪些部分代码是合法的(多选)

class Base{

public void amethod(int i){}

}

public class Scope extends Base{

public static void main(String argv[]){

}

//在这里定义一个方法

}

a)void amethod(int i)throws Exception{}//继承后方法重写，**父类用什么访问类型重写也要有相同的类型才可以**

b)void amethod(long i)throws Exception{}//正确，不是方法重写long类型不一样

c)void amethod(long i){}//正确，不是方法重写

d)public void amethod(int i)throws Exception{}//继承后可以**重写但不能抛异常**

17、下面哪行代码输出-4.0

a)System.out.println(Math.floor(-4.7));//floor向下取个比它小的最近double -5.0

b)System.out.println(Math.round(-4.7));//ronud四舍五入返回一个数-5.0

c)System.out.println(Math.ceil(-4.7));//ceil向上取一个比它大的最近double -4.0

d)System.out.println(Math.min(-4.7));//Math.min(x,y)需要两个数进行比较

18、如果运行下面的代码会输出什么内容

String s=new String("Bicycle");

int iBegin=1;

char iEnd=3;

System.out.println(s.substring(iBegin,iEnd));//**解决办法在String的substring方法**

1)Bic

2)ic //正确

3)icy

4)error:没有匹配的方法substring(int,char)

19、给出下面的代码在注释部分放置什么样的代码会输出"Equal"

public class EqTest{

public static void main(String argv[]){

EqTest e=new EqTest();

}

EqTest(){

String s="Java";

String s2="java";

//place test here{

System.out.println("Equal");

}else{

System.out.println("Not equal");

}

}

}

a)if(s==s2)

b)if(s.equals(s2)**//equals比较内容是否相同，区分大小写**

c)if(s.equalsIgnoreCase(s2))//正确,**equalsIgnoreCase不分大小写进行内容比较**。

d)if(s.noCaseMatch(s2))

20、给出下面的代码，怎样调用Base的构造方法会输出"base constructor"

class Base{

Base(int i){

System.out.println("base constructor");

}

Base(){

}

}

public class Sup extends Base{

public static void main(String argv[]){

Sup s=new Sup();

//One

}

Sup(){

//Two

}

public void derived(){

//Three

}

}

a)//One后放置Base(10);//无意义

b)//One后放置super(10);//无法放到One后面，语法错误(super代表直接指向父类)

c)//Two后放置super(10);//正确,**如果希望在子类中去调用父类的构造方法，要求在子类的构造函数中去调用。**

d)//Three后放置super(10);

-------------------------------------------------------------------------------

//**super()的使用**[Demo154.java]

public class Demo154{

public static void main(String []args){

Son son=new Son(11,"aa");

}

}

class Base{

int age;

String name;

public Base(int age,String name){

System.out.println("调用父类Base构造方法");

this.age=age;

this.name=name;

}

}

class Son extends Base{

public Son(int age,String name){

super(age,name);//**super()方法必需放在子类构造函数的第一行**

System.out.println("调用子类Son构造方法");

}

}

-------------------------------------------------------------------------------

**JAVA关键字与保留字说明及使用**

abstract - 1 - boolean - 2 - break - 2 - byte - 2 -

case - 3 - catch - 3 - char - 4 - class - 4 -

continue - 5 - default - 5 - do - 6 - double - 6 -

else - 6 - extends - 6 - false - 7 - final - 7 -

finally - 7 - float - 8 - for - 8 - if - 8 -

implements - 9 - import - 9 - instanceof - 9 -

int - 9 - interface - 10 - long - 10 - native - 10 -

new - 11 - null - 11 - package - 11 - private - 11 -

protected - 12 - public - 12 - return - 13 - short - 13 -

static - 13 - super - 14 - switch - 14 - synchronized - 15 -

this - 16 - throw - 16 - throws - 16 - transient - 17 -

try - 17 - true - 18 - void - 18 - volatile - 18 -

while - 18 -

***1.abstract***

abstract 关键字可以修改类或方法。

abstract 类可以扩展（增加子类），但不能直接实例化。

abstract 方法不在声明它的类中实现，但必须在某个子类中重写。

-示例-

public abstract class MyClass{

}

public abstract String myMethod();

-注释-

采用 abstract 方法的类本来就是抽象类，并且必须声明为 abstract。

abstract 类不能实例化。

仅当 abstract 类的子类实现其超类的所有 abstract 方法时，才能实例化 abstract 类的子类。这种类称为具体类，以区别于 abstract 类 。

如果 abstract 类的子类没有实现其超类的所有 abstract 方法，该子类也是 abstract 类。

abstract 关键字不能应用于 static、private 或 final 方法，因为这些方法不能被重写，因此，不能在子类中实现。

final 类的方法都不能是 abstract，因为 final 类不能有子类。

***2.boolean***

变量的值可以是 true 或 false。

-示例-

boolean valid = true;

if (valid){

<statement>

}

-注释-

boolean 变量只能以 true 或 false 作为值。

boolean 不能与数字类型相互转换。

包含 boolean 操作数的表达式只能包含 boolean 操作数。

Boolean 类是 boolean 原始类型的包装对象类。

***3.break***

用于提前退出 for、while 或 do 循环，或者在 switch 语句中用来结束 case 块。

-示例-

for (i=0; i<max; i++){

if (<loop finished early>){

break;

}

}

int type = <some value>;

switch (type){

case 1:

<statement>

break;

case 2:

<statement>

break;

default:

<statement>

}

-注释-

break 总是退出最深层的 while、for、do 或 switch 语句。

***4.byte***

byte 是 Java 原始类型。

byte 可存储在 [-128, 127] 范围以内的整数值。

-示例-

byte b = 124;

-注释-

Byte 类是 byte 原始类型的包装对象类。它定义代表此类型的值的范围的 MIN\_VALUE 和 MAX\_VALUE 常量。

Java 中的所有整数值都是 32 位的 int 值，除非值后面有 l 或 L（如 235L），这表示该值应解释为 long。

***5.case***

用来标记 switch 语句中的每个分支。

-示例-

int arg = <some value>;

switch (arg){

case 1:

<statements>

break;

case 2:

<statements>

break;

default:

<statements>

break;

}

-注释-

case 块没有隐式结束点。break 语句通常在每个 case 块末尾使用，用于退出 switch 语句。

如果没有 break 语句，执行流将进入所有后面的 case 和/或 default 块。

***6.catch***

catch 关键字用来在 try-catch 或 try-catch-finally 语句中定义异常处理块。

-示例-

try{

<可能引发异常的块>

}catch (<java.lang.Exception 或子类> e){

<处理异常 e 的代码>

}

try{

<可能引发其他异常的块>

}catch (FooException e){

<处理 FooException e 的代码>

}catch (BarException e){

<处理 BarException e 的代码>

}

try{

<可能引发异常的块>

}catch (<java.lang.Exception 或子类> e){

<处理异常 e 的代码>

}finally{

<有异常或无异常情况下都执行的语句>

}

-注释-

开始和结束标记 { 和 } 是 catch 子句语法的一部分，即使该子句只包含一个语句，也不能省略这两个标记。

每个 try 块都必须至少有一个 catch 或 finally 子句。

如果某个特定异常类未被任何 catch 子句处理，该异常将沿着调用栈递归地传播到下一个封闭 try 块。如果任何封闭 try 块都未捕获到异常，Java 解释器将退出，并显示错误消息和堆栈跟踪信息。

***7.char***

char 是 Java 原始类型。

char 变量可以存储一个 Unicode 字符。

-示例-

char delimiter = ';';

-注释-

可以使用下列 char 常量：

\b - 空格

\f - 换页

\n - 换行

\r - 回车

\t - 水平制表符

\' - 单引号

\" - 双引号

\" - 反斜杠

\xxx - 采用 xxx 编码的 Latin-1 字符。\x 和 \xx 均为合法形式，但可能引起混淆。

\uxxxx - 采用十六进制编码 xxxx 的 Unicode 字符。

Character 类包含一些可用来处理 char 变量的 static 方法，这些方法包括 isDigit()、isLetter()、isWhitespace() 和 toUpperCase()。

char 值没有符号。

***8.class***

class 关键字用来声明新的 Java 类，该类是相关变量和/或方法的集合。

类是面向对象的程序设计方法的基本构造单位。类通常代表某种实际实体，如几何形状或人。类是对象的模板。每个对象都是类的一个实例。

要使用类，通常使用 new 操作符将类的对象实例化，然后调用类的方法来访问类的功能。

-示例-

public class Rectangle{

float width;

float height;

public Rectangle(float w, float h){

width = w;

height = h;

}

public float getWidth(){

return width;

}

public float getHeight(){

return height;

}

}

***9.continue***

continue 关键字用来跳转到 for、while 或 do 循环的下一个迭代。

-示例-

for (i=0; i<max; i++){

<statements>

if (<done with this iteration>){

continue;

}

<statements>

}

-注释-

continue 总是跳到最深层 while、for 或 do 语句的下一个迭代。

***10.default***

default 关键字用来标记 switch 语句中的默认分支。

-示例-

int arg = <some value>;

switch (arg){

case 1:

<statements>

break;

case 2:

<statements>

break;

default:

<statements>

break;

}

-注释-

default 块没有隐式结束点。break 语句通常在每个 case 或 default 块的末尾使用，以便在完成块时退出 switch 语句。

如果没有 default 语句，其参数与任何 case 块都不匹配的 switch 语句将不执行任何操作。

***11.do***

do 关键字用于指定一个在每次迭代结束时检查其条件的循环。

-示例-

do{

<statements>

}

while (!found);

-注释-

do 循环体至少执行一次。

条件表达式后面必须有分号。

***12.double***

double 是 Java 原始类型。

double 变量可以存储双精度浮点值。

-示例-

double ratio = .01;

double diameter = 6.15;

double height = 1.35E03; // 1.35 \* 103 或 1350.0

double height = 1e-2; // 1.0 \* 10-2 或 0.01

-注释-

由于浮点数据类型是实际数值的近似值，因此，一般不要对浮点数值进行是否相等的比较。

Java 浮点数值可代表无穷大和 NaN（非数值）。Double 包装对象类用来定义常量 MIN\_VALUE、MAX\_VALUE、NEGATIVE\_INFINITY、POSITIVE\_INFINITY 和 NaN。

***13.else***

else 关键字总是在 if-else 语句中与 if 关键字结合使用。else 子句是可选的，如果 if 条件为 false，则执行该子句。

-示例-

if (condition){

<statements>

}else{

<statements>

}

***14.extends***

extends 关键字用在 class 或 interface 声明中，用于指示所声明的类或接口是其名称后跟有 extends 关键字的类或接口的子类。

-示例-

public class Rectangle extends Polygon{

}

-注释-

在上例中，Rectangle 类继承 Polygon 类的所有 public 和 protected 变量和方法。

Rectangle 类可以重写 Polygon 类的任何非 final 方法。

一个类只能扩展一个其他类。

***15.false***

false 关键字代表 boolean 变量的两个合法值之一。

-示例-

boolean isComplete = false;

***16.final***

final 关键字可以应用于类，以指示不能扩展该类（不能有子类）。

final 关键字可以应用于方法，以指示不能重写任何子类中的方法。

-示例-

public final class MyFinalClass{

}

public class MyClass{

public final String myFinalMethod(){

<statements>

}

}

-注释-

一个类不能同时是 abstract 又是 final。abstract 意味着必须扩展类，final 意味着不能扩展类。

一个方法不能同时是 abstract 又是 final。abstract 意味着必须重写方法，final 意味着不能重写方法。

***17.finally***

finally 关键字用来定义始终在 try-catch-finally 语句中执行的块。

finally 块通常包含清理代码，用在部分执行 try 块后恢复正常运行。

-示例-

try{

<可能引发异常的块>

}catch (<java.lang.Exception 或子类> e){

<处理异常 e 的代码>

}finally{

<有异常或无异常情况下都执行的语句>

}

-注释-

开始和结束标记 { 和 } 是 finally 子句语法的一部分，即使该子句只包含一个语句，也不能省略这两个标记。

每个 try 块都必须至少有一个 catch 或 finally 子句。

如果执行 try 块的任何部分，不论是否出现异常，也不论 try 或 catch 块是否包含 return、continue 或 break 语句，都一定会执行 finally 块中的代码。

如果不出现异常，控件将跳过 try 块，进入 finally 块。

如果在执行 try 块期间出现异常，并且相应的 catch 块包含 break、continue 或 return 语句，控件将首先穿过 finally 块，之后再执行 break、continue 或 return。

***18.float***

float 是 Java 原始类型。

float 变量可以存储单精度浮点值。

-示例-

float ratio = .01;

float diameter = 6.15;

float height = 1.35E03; // 1.35 \* 103 或 1350.0

float height = 1e-2; // 1.0 \* 10-2 或 0.01

-注释-

使用此关键字时应遵循下列规则：

Java 中的浮点文字始终默认为双精度。要指定单精度文字值，应在数值后加上 f 或 F，如 0.01f。

由于浮点数据类型是实际数值的近似值，因此，一般不要对浮点数值进行是否相等的比较。

Java 浮点数值可代表无穷大和 NaN（非数值）。Float 包装对象类用来定义常量 MIN\_VALUE、MAX\_VALUE、NEGATIVE\_INFINITY、POSITIVE\_INFINITY 和 NaN。

***19.for***

for 关键字用于指定一个在每次迭代结束前检查其条件的循环。

-示例-

int i;

for (i=0; i<max; i++){

<statements>

}

-注释-

for 语句的形式为 for(initialize; condition; increment)

控件流进入 for 语句时，将执行一次 initialize 语句。

每次执行循环体之前将计算 condition 的结果。如果 condition 为 true，则执行循环体。

每次执行循环体之后，在计算下一个迭代的 condition 之前，将执行 increment 语句。

***20.if***

if 关键字指示有条件地执行代码块。条件的计算结果必须是布尔值。

-示例-

if (condition){

<statements>

}

if (condition){

<statements>

}else{

<statements>

}

-注释-

if 语句可以有可选的 else 子句，该子句包含条件为 false 时将执行的代码。

包含 boolean 操作数的表达式只能包含 boolean 操作数。

***21.implements***

implements 关键字在 class 声明中使用，以指示所声明的类提供了在 implements 关键字后面的名称所指定的接口中所声明的所有方法的实现。

-示例-

public class Truck implements IVehicle{

}

-注释-

在上例中，Truck 类必须提供在 IVehicle 接口中所声明的所有方法的实现。

否则，Truck 类将是独立的；它可以声明其他方法和变量，并扩展另一个类。

一个类可以实现多个接口。

***22.import***

import 关键字使一个包中的一个或所有类在当前 Java 源文件中可见。可以不使用完全限定的类名来引用导入的类。

-示例-

import java.io.File;

import java.net.\*;

-注释-

当多个包包含同名的类时，许多 Java 程序员只使用特定的 import 语句（没有“\*”）来避免不确定性。

***23.instanceof***

instanceof 关键字用来确定对象所属的类。

-示例-

if (node instanceof TreeNode){

<statements>

}

-注释-

在上例中，如果 node 是 TreeNode 类的实例，或者是 TreeNode 的子类的实例，则 instanceof 表达式的值将为 true。

***24.int***

int 是 Java 原始类型。

int 变量可以存储 32 位的整数值。

-示例-

int number = 5;

int octalNumber = 0377;

int hexNumber = 0xff;

-注释-

Integer 类是 int 原始类型的包装对象类。它定义代表此类型的值的范围的 MIN\_VALUE 和 MAX\_VALUE 常量。

Java 中的所有整数值都是 32 位的 int 值，除非值后面有 l 或 L（如 235L），这表示该值应解释为 long。

***25.interface***

interface 关键字用来声明新的 Java 接口，接口是方法的集合。

接口是 Java 语言的一项强大功能。任何类都可声明它实现一个或多个接口，这意味着它实现了在这些接口中所定义的所有方法。

-示例-

public interface IPolygon{

public float getArea();

public int getNumberOfSides();

public int getCircumference();

}

-注释-

实现了接口的任何类都必须提供在该接口中的所有方法的实现。

一个类可以实现多个接口。

***26.long***

long 是 Java 原始类型。

long 变量可以存储 64 位的带符号整数。

-示例-

long number = 5;

long anotherNumber = 34590L;

long octalNumber = 0377;

long hexNumber = 0xffl;

-注释-

Long 类是 long 原始类型的包装对象类。它定义代表此类型的值的范围的 MIN\_VALUE 和 MAX\_VALUE 常量。

Java 中的所有整数值都是 32 位的 int 值，除非值后面有 l 或 L（如 235L），这表示该值应解释为 long。

***27.native***

native 关键字可以应用于方法，以指示该方法是用 Java 以外的语言实现的。

-示例-

native String getProcessorType();

-注释-

Native 方法不在此文档的讨论范围内。

***28.new***

new 关键字用于创建类的新实例。

-示例-

String sName = new String();

Float fVal = new Float(0.15);

-注释-

new 关键字后面的参数必须是类名，并且类名的后面必须是一组构造方法参数（必须带括号）。

参数集合必须与类的构造方法的签名匹配。

= 左侧的变量的类型必须与要实例化的类或接口具有赋值兼容关系。

***29.null***

null 是 Java 的保留字，表示无值。

-示例-

Integer i;

i = null;

String s;

if (s != null){

<statements>

}

-注释-

将 null 赋给非原始变量相当于释放该变量先前所引用的对象。

不能将 null 赋给原始类型（byte、short、int、long、char、float、double、boolean）变量。

***30.package***

package 关键字指定在 Java 源文件中声明的类所驻留的 Java 包。

-示例-

package com.mycompany;

public class MyClass{

}

-注释-

package 语句（如果出现）必须是 Java 源文件中的第一个非-注释-性文本。

在上面的-示例-中，MyClass 类的完全限定类名是 com.mycompany.MyClass。

如果 Java 源文件不包含 package 语句，在该文件中定义的类将位于“默认包”中。请注意，不能从非默认包中的类引用默认包中的类。

***31.private***

private 关键字是访问控制修饰符，可以应用于类、方法或字段（在类中声明的变量）。

-示例-

public class MyPublicClass{

private class MyPrivateClass{

}

private int i;

private String myMethod(){

<statements>

}

}

-注释-

只能在声明 private（内部）类、方法或字段的类中引用这些类、方法或字段。在类的外部或者对于子类而言，它们是不可见的。

所有类成员的默认访问范围都是 package 访问，也就是说，除非存在特定的访问控制修饰符，否则，可以从同一个包中的任何类访问类成员。

***32.protected***

protected 关键字是可以应用于类、方法或字段（在类中声明的变量）的访问控制修饰符。

-示例-

public class MyPublicClass{

protected class MyPrivateClass{

}

protected int i;

protected String myMethod(){

<statements>

}

}

-注释-

可以在声明 protected 类、方法或字段的类、同一个包中的其他任何类以及任何子类（无论子类是在哪个包中声明的）中引用这些类、方法或字段。

所有类成员的默认访问范围都是 package 访问，也就是说，除非存在特定的访问控制修饰符，否则，可以从同一个包中的任何类访问类成员。

***33.public***

public 关键字是可以应用于类、方法或字段（在类中声明的变量）的访问控制修饰符。

-示例-

public class MyPublicClass{

public class MyPrivateClass{

}

public int i;

public String myMethod(){

<statements>

}

}

-注释-

可能只会在其他任何类或包中引用 public 类、方法或字段。

所有类成员的默认访问范围都是 package 访问，也就是说，除非存在特定的访问控制修饰符，否则，可以从同一个包中的任何类访问类成员。

***34.return***

return 关键字会导致方法返回到调用它的方法，从而传递与返回方法的返回类型匹配的值。

-示例-

public void myVoidMethod(){

<statements>

return;

}

public String myStringMethod(){

String s = "my response";

return s;

}

public int myIntMethod(){

int i = 5;

return(i);

}

-注释-

如果方法具有非 void 的返回类型，return 语句必须具有相同或兼容类型的参数。

返回值两侧的括号是可选的。

***35.short***

short 是 Java 原始类型。

short 变量可以存储 16 位带符号的整数。

-示例-

short number = 5;

short octalNumber = 0077;

short hexNumber = 0xff;

-注释-

Short 类是 short 原始类型的包装对象类。它定义代表此类型的值的范围的 MIN\_VALUE 和 MAX\_VALUE 常量。

Java 中的所有整数值都是 32 位的 int 值，除非值后面有 l 或 L（如 235L），这表示该值应解释为 long。

***36.static***

static 关键字可以应用于内部类（在另一个类中定义的类）、方法或字段（类的成员变量）。

-示例-

public class MyPublicClass{

public final static int MAX\_OBJECTS = 100;

static int \_numObjects = 0;

static class MyStaticClass{

}

static int getNumObjects(){

}

}

-注释-

通常，static 关键字意味着应用它的实体在声明该实体的类的任何特定实例外部可用。

static（内部）类可以被其他类实例化和引用（即使它是顶级类）。在上面的-示例-中，另一个类中的代码可以实例化 MyStaticClass 类，方法是用包含它的类名来限定其名称，如 MyClass.MyStaticClass。

static 字段（类的成员变量）在类的所有实例中只存在一次。

可以从类的外部调用 static 方法，而不用首先实例化该类。这样的引用始终包括类名作为方法调用的限定符。在上面的示例中，MyClass 类外部的代码以 MyClass.getNumObjects() 的形式调用 getNumObjects() static 方法。

模式：

public final static <type> varName = <value>;

通常用于声明可以在类的外部使用的类常量。在引用这样的类常量时需要用类名加以限定。在上面的-示例-中，另一个类可以用 MyClass.MAX\_OBJECTS 形式来引用 MAX\_OBJECTS 常量。

***37.super***

super 关键字用于引用使用该关键字的类的超类。

-示例-

public class MyClass{

public MyClass(String arg){

super(arg);

}

public String myStringMethod(){

return super.otherStringMethod();

}

}

-注释-

作为独立语句出现的 super 表示调用超类的构造方法。

super.<methodName>() 表示调用超类的方法。只有在如下情况中才需要采用这种用法：要调用在该类中被重写的方法，以便指定应当调用在超类中的该方法。

***38.switch***

switch 语句用于基于某个表达式选择执行多个代码块中的某一个。

-示例-

int arg = <some value>;

switch (arg){

case 1:

<statements>

break;

case 2:

<statements>

break;

default:

<statements>

break;

}

char arg = <some value>;

switch (arg){

case 'y':

case 'Y':

<statements>

break;

case 'n':

case 'N':

<statements>

break;

default:

<statements>

break;

}

-注释-

switch 条件的计算结果必须等于 byte、char、short 或 int。

case 块没有隐式结束点。break 语句通常在每个 case 块末尾使用，用于退出 switch 语句。

如果没有 break 语句，执行流将进入所有后面的 case 和/或 default 块。

***39.synchronized***

synchronized 关键字可以应用于方法或语句块，并为一次只应由一个线程执行的关键代码段提供保护。

-示例-

public class MyClass{

public synchronized static String mySyncStaticMethod(){

}

public synchronized String mySyncMethod(){

}

}

public class MyOtherClass{

Object someObj;

public String myMethod(){

<statements>

synchronized (someObj){

<statements affecting someObj>

}

}

}

-注释-

synchronized 关键字可防止代码的关键代码段一次被多个线程执行。

如果应用于静态方法（如上例中的 MySyncStaticMethod），那么，当该方法一次由一个线程执行时，整个类将被锁定。

如果应用于实例方法（如上例中的 MySyncMethod），那么，当该方法一次由一个线程访问时，该实例将被锁定。

如果应用于对象或数组，当关联的代码块一次由一个线程执行时，对象或数组将被锁定。

***40.this***

this 关键字用于引用当前实例。

-示例-

public class MyClass{

int number;

public MyClass(int number){

this.number = number;

}

}

-注释-

当引用可能不明确时，可以使用 this 关键字来引用当前的实例。

在上面的-示例-中，构造方法参数 number 与类的成员变量同名。this.number 明确表示 MyClass 的该实例的 number 成员变量。

***41.throw***

throw 关键字用于引发异常。

-示例-

import java.io.IOException;

public class MyClass{

public method readFile(String filename) throws IOException{

<statements>

if (error){

throw new IOException("error reading file");

}

}

}

-注释-

throw 语句将 java.lang.Throwable 作为参数。Throwable 在调用栈中向上传播，直到被适当的 catch 块捕获。

引发非 RuntimeException 异常的任何方法还必须在方法声明中使用 throws 修饰符来声明它引发的异常。

***42.throws***

throws 关键字可以应用于方法，以便指出方法引发了特定类型的异常。

-示例-

import java.io.IOException;

public class MyClass{

public method readFile(String filename) throws IOException{

<statements>

if (error){

throw new IOException("error reading file");

}

}

}

-注释-

throws 关键字将逗号分隔的 java.lang.Throwables 列表作为参数。

引发非 RuntimeException 异常的任何方法还必须在方法声明中使用 throws 修饰符来声明它引发的异常。

要在 try-catch 块中包含带 throws 子句的方法的调用，必须提供该方法的调用者。

***43.transient***

transient 关键字可以应用于类的成员变量，以便指出该成员变量不应在包含它的类实例已序列化时被序列化。

-示例-

public class MyClass{

private transient String password;

}

***44.try***

try 关键字用于包含可能引发异常的语句块。

-示例-

try{

<可能引发异常的块>

}catch (<java.lang.Exception 或子类> e){

<处理异常 e 的代码>

}

try{

<可能引发其他异常的块>

}catch (FooException e){

<处理 FooException e 的代码>

}catch (BarException e){

<处理 BarException e 的代码>

}

try{

<可能引发异常的块>

}catch (<java.lang.Exception 或子类> e){

<处理异常 e 的代码>

}finally{

<有异常或无异常情况下都执行的语句>

}

-注释-

每个 try 块都必须至少有一个 catch 或 finally 子句。

如果某个特定异常类未被任何 catch 子句处理，该异常将沿着调用栈递归地传播到下一个封闭 try 块。如果任何封闭 try 块都未捕获到异常，Java 解释器将退出，并显示错误消息和堆栈跟踪信息。

***45.true***

true 关键字表示 boolean 变量的两个合法值中的一个。

-示例-

boolean isComplete = true;

***46.void***

void 关键字表示 null 类型。

-示例-

public class MyClass{

public void doSomething(){

<statements>

return;

}

}

-注释-

void 可以用作方法的返回类型，以指示该方法不返回值。

***47.volatile***

volatile 关键字用于表示可以被多个线程异步修改的成员变量。

注意：volatile 关键字在许多 Java 虚拟机中都没有实现。

-示例-

public class MyClass{

volatile int sharedValue;

}

-注释-

volatile 的目标用途是为了确保所有线程所看到的指定变量的值都是相同的。

***48.while***

while 关键字用于指定一个只要条件为真就会重复的循环。

-示例-

while (!found){

<statements>

}

# 图形开发

# 1、图形用户界面(gui)介绍

图形用户界面(Graphics User Interface,GUI)是用户与程序交互的窗口，比命令行的界面更加直观并且更好操作。

awt,swing,swt,Jface是什么？

Sun已经提供了一个跨平台GUI开发工具包AWT抽象窗口工具箱(Abstract Window Toolkit).

↓

Sun又创建了一个新的GUI框架swing.解决了AWT存在的Icd问题.

↓

IBM认为swing比较消耗内存，创建了一个新的GUI库，这就是SWT

↓

IBM为了方便开发SWT程序，在SWT基础上又创建了一个更易用，功能强大的图开包"JFace"

2、eclipse开发工具介绍

eclipse是什么东西，我们用三句话来说明：

1、eclipse最早是IBM附属公司oti开发的，一共投入了4000万美金，后来捐献给开源社区

2、eclipse是一个开源的、可扩展的集成开发环境，已经成为目前最流行的java开发工具

3、eclipse安装后就可以开发java se的项目了，但不能开发java ee项目，需要安装web开发插件(lomboz或是myeclipse..)

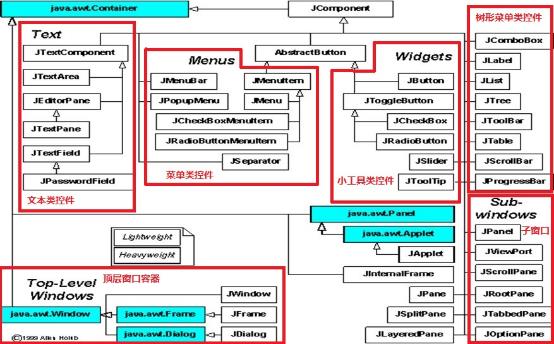
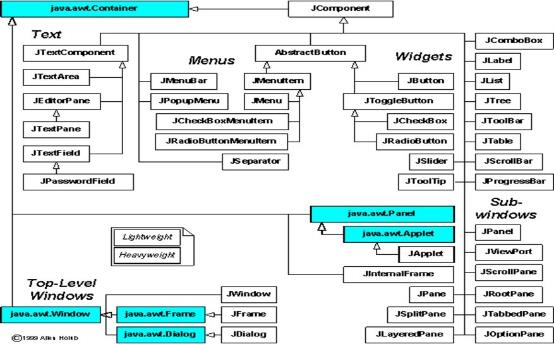
ide(集成开发环境，比如jcreator、vs、myeclipse这些开发工具都是ide)

3、swing组件介绍

1、*JFrame是F*rame的子类

2、属于容器类组件，顶层容器

3、JFrame有一些常用的方法，通过示例与java的帮助文档即可了解。



按钮组件为什么有？

在图形用户界面编程中，我们在窗体中会经常使用到按钮，我们在进行选择的时候，常常需要确认，所以按钮很重要。

按钮组件怎么用？

只需要在窗体中添加按钮组件(JButton)即可完成。

## 按钮组件--深入讨论

1、JButton是AbstractButton的子类

2、属于窗口类组件，可以加入别的组件

3、Swing包的按钮组件不只有JButton，还有单选按钮(JRadioButton)、箭头按钮(BasicArrowButton)、触发器按钮(JToggleButton)..这些按钮我们在后面讲解。

//功能：GUI界面开发演示[Window001.java]

import java.awt.\*;//开发图形要引入java.awt.\*包

import javax.swing.\*;//开发图形要引入javax.swing.\*包

public class Window001 extends JFrame{//继承JFrame顶层容器类(可以添加其它swing组件的类)

//把需要的swing组件，定义到这里

JButton jb1=null;

public static void main(String[] args) {

Window001 win=new Window001();

}

//构造函数

public Window001(){

//创建一个button按钮

jb1=new JButton("按钮");

//添加JButton组件

this.add(jb1);

//给窗体设置标题

this.setTitle("Hello World!");

//设置窗体大小,按像素设置大小

this.setSize(500, 500);

//设置窗体初始位置

this.setLocation(500, 150);

//设置当关闭窗口时，保证JVM也退出

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//显示窗体

this.setVisible(true);//true显示，false不显示

}

}

## 4、三大常用布局管理器

布局管理器--介绍

1、概念

**组件在容器(比如JFrame)中的位置和大小是由布局管理器来决定的。所有的容器都会**

**使用一个布局管理器，通过它来自动进行组件的布局管理。**

2、种类

java共提供了五种布局管理器：流式布局管理器(FlowLayout)、边界布局管理器(BorderLayout)、网格布局管理器(GridLayout)、卡片布局管理器(CardLayout)、网格包布局管理器(GridBagLayout)。其中前三种是最常见的布局管理器。

### 边界布局BorderLayout--介绍

边界布局(BorderLayout)将容器简单的划分为东南西北5个区域，中间区域最大。

JFrame窗体，JDialog对话框组件默认布局方法

边界布局BorderLayout--使用[Window002.java]

/\*\*

\* 边界布局BorderLayout使用演示

\* 1、继承JFrame

\* 2、定义你需要的各个组件

\* 3、创建组件(在构造函数中组件)

\* 4、添加组件

\* 5、对窗体设置

\* 6、显示窗体

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window002 extends JFrame{

//定义组件

JButton jb1,jb2,jb3,jb4,jb5;

public static void main(String[] args) {

Window002 win=new Window002();

}

public Window002(){

//创建组件

jb1=new JButton("中部");

jb2=new JButton("北部");

jb3=new JButton("东部");

jb4=new JButton("南部");

jb5=new JButton("西部");

//添加各个组件

this.add(jb1, BorderLayout.CENTER);//BorderLayout.CENTER添加到中部

this.add(jb2, BorderLayout.NORTH);//BorderLayout.NORTH添加到北部

this.add(jb3, BorderLayout.EAST);//BorderLayout.EAST添加到东部

this.add(jb4, BorderLayout.SOUTH);//BorderLayout.SOUTH添加到南部

this.add(jb5, BorderLayout.WEST);//BorderLayout.WEST添加到西部

//设置窗体属性

this.setTitle("边界布局演示");//窗体标题名称

this.setSize(300, 200);//窗体尺寸

this.setLocation(200, 200);//窗体在屏幕打开时的初始位置

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//退出窗体后将JFrame同时关闭

//显示窗体

this.setVisible(true);

}

}

------------------------------------------------------------------------------

**边界布局BorderLayout--注意事项**

1、不是五个部分都必需添加；

2、中部组件会自动调节大小；

3、JFrame，Jdialog默认布局管理器就是BorderLayout

三大常用布局管理器--流式布局

### 流式布局FlowLayout--介绍

FlowLayout布局，按照组件的添加次序将按钮组件(当然也可以是别的组件)从左到右放置在容器中。当到达容器的边界时，组件将放置到下一行中。FlowLayout可以以左对齐、居中对齐、以右对齐的方式排列组件。

流式布局FlowLayout--使用[Window003.java]

/\*\*

\* 流式布局FlowLayout使用演示

\* 1、继承JFrame

\* 2、定义你需要的各个组件

\* 3、创建组件(在构造函数中组件)

\* 4、添加组件

\* 5、对窗体设置

\* 6、显示窗体

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window003 extends JFrame{

//定义组件

JButton jb1,jb2,jb3,jb4,jb5,jb6;

public static void main(String[] args) {

Window003 win=new Window003();

}

public Window003(){

//创建组件

jb1=new JButton("关羽");

jb2=new JButton("张飞");

jb3=new JButton("赵云");

jb4=new JButton("马超");

jb5=new JButton("黄忠");

jb6=new JButton("魏延");

//添加各个组件

this.add(jb1);

this.add(jb2);

this.add(jb3);

this.add(jb4);

this.add(jb5);

this.add(jb6);

//设置布局管理器,流式布局默认为居中对齐

this.setLayout(new FlowLayout(FlowLayout.LEFT));

//new FlowLayout(FlowLayout.LEFT)流式布局，(FlowLayout.??)??可以设置为不同方式对齐。

//设置窗体属性

this.setTitle("流式布局演示");//窗体标题名称

this.setSize(300, 200);//窗体尺寸

this.setLocation(200, 200);//窗体在屏幕打开时的初始位置

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//退出窗体后将JFrame同时关闭

//禁止用户改变窗体大小

this.setResizable(false);

//显示窗体

this.setVisible(true);

}

}

------------------------------------------------------------------------------

**流式布局FlowLayout--注意事项**

1、不限制他所管理的组件大小，允许他们有最佳大小

2、当容器衩缩放时，组件的位置可能变化，但组件的大小不变。

3、默认组件是居中对齐，可以通过FlowLayout(intalign)函数来指定对齐方式。

三大常用布局管理器--网格

### 网格布局GridLayout--介绍

**GridLayout布局，**听其名而知其意，它将容器分割成多行多列，组件被填充到每个网格中，添加到容器中的组件首先放置在左上角的网格中，然后从左到右放置其它的组件，当占满该行的所有网格后，接着继续在下一行从左到右放置组件。

网格布局GridLayout--使用[Window004.java]

/\*\*

\* 网格布局GridLayout使用演示

\* 1、继承JFrame

\* 2、定义你需要的各个组件

\* 3、创建组件(在构造函数中组件)

\* 4、添加组件

\* 5、对窗体设置

\* 6、显示窗体

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window004 extends JFrame{

//定义组件

int size=9;

JButton jbs[]=new JButton[size];

public static void main(String[] args) {

Window004 win=new Window004();

}

public Window004(){

//创建组件

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

jbs[i]=new JButton(String.valueOf(i+1));

}

//添加各个组件

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

this.add(jbs[i]);

}

//设置网格布局管理器

this.setLayout(new GridLayout(3,3,10,10));//new GridLayout(行,列)设置布局有几行几列

//设置窗体属性

this.setTitle("网格布局演示");//窗体标题名称

this.setSize(300, 200);//窗体尺寸

this.setLocation(200, 200);//窗体在屏幕打开时的初始位置

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//退出窗体后将JFrame同时关闭

//禁止用户改变窗体大小

this.setResizable(false);

//显示窗体

this.setVisible(true);

}

}

------------------------------------------------------------------------------

**网格布局GridLayout--注意事项**

1、组件的相对位置不随容器的缩放而变化，但大小会变化；

2、所有组件的大小相同；

3、可以通过GridLayout(int rows,int cols,int hgap,int vgap)来指定网格的行/列，水平间隙/垂直间隙。

参数说明：rows:行数/cols:列数/hgap:垂直间隙/vgap:水平间隙

三大常用布局管理器--小结

**开发GUI程序步骤**

**1、继承JFrame**

**2、定义需要的组件**

**3、创建组件**

**4、设置布局管理器**

**5、添加组件**

**6、显示窗体**

Swing组件--面板组件

## 面板组件(JPanel)--介绍

在图形用户界面编程中，如果只是普通的组件布局，我们用前面讲的三种布局管理器就可以解决，但在比较复杂的布局要求时，就需要使用布局管理器的组合使用。

面板组件(JPanel)--使用[Window005.java]

JPanel：面板组件，非顶层容器，一个界面只可以有一个JFrame窗体组件，但可以有多个JPanel面板组件，而**JPanel**上也可以使用FlowLayout、BorderLayout、GirdLayout等各种布局管理器，这样可以组合使用达到较为复杂的布局效果。

/\*\*

\* 多种布局管理器，面板组件(JPanel)--使用

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window005 extends JFrame{

//定义组件

JPanel jp1,jp2;

JButton jb1,jb2,jb3,jb4,jb5,jb6;

public static void main(String[] args) {

Window005 win=new Window005();

}

//构造函数

public Window005(){

//创建组件

jp1=new JPanel();//JPanel布局默认是FlowLayout流布局

jp2=new JPanel();

jb1=new JButton("西瓜");

jb2=new JButton("苹果");

jb3=new JButton("荔枝");

jb4=new JButton("葡萄");

jb5=new JButton("桔子");

jb6=new JButton("香蕉");

//设置布局管理器(Jpanel默认流布局)

//添加JPanel

jp1.add(jb1);

jp1.add(jb2);

jp2.add(jb3);

jp2.add(jb4);

jp2.add(jb5);

//把Panel加入JFrame

this.add(jp1, BorderLayout.NORTH);

this.add(jb6, BorderLayout.CENTER);

this.add(jp2, BorderLayout.SOUTH);

//设置窗体

this.setSize(300, 250);//窗体大小

this.setLocation(200, 200);//屏幕显示初始位置

this.setVisible(true);//显示

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//退出窗体后将JFrame同时关闭

}

}

------------------------------------------------------------------------------

**面板(JPanel)组件--注意事项**

1、JPanel是JComponent的子类；

2、属于容器类组件，可以加入别的组件；

3、默认布局管理器是流式布局(FlowLayout)

## **Swing组件--文本框/密码框/标签组件**[Window006.java]

几个常用组件

在图形用户界面编程中，我们常常会提供用户登录界面，比如登录到会员管理系统，登录到工资管理系统、仓库管理系统等。这时候我们就会用到：

1、文本框(JTextField)

2、密码框(JPasswordField)

3、标签(JLable)

/\*\*

\* Swing组件--文本框(JTextField)/密码框(JPasswordField)/标签(JLable)组件使用

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window006 extends JFrame{

//定义组件

JPanel jp1,jp2,jp3;//面板

JLabel jlb1,jlb2;//标签

JButton jb1,jb2;//按钮

JTextField jtf;//文本

JPasswordField jpf;//密码

public static void main(String[] args) {

Window006 win=new Window006();

}

//构造函数

public Window006(){

//创建面板

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//创建标签

jlb1=new JLabel("用户名");

jlb2=new JLabel("密 码");

//创建按钮

jb1=new JButton("登录");

jb2=new JButton("取消");

//创建文本框

jtf=new JTextField(10);

//创建密码框

jpf=new JPasswordField(10);

//设置布局管理

this.setLayout(new GridLayout(3, 1));//网格式布局

//加入各个组件

jp1.add(jlb1);

jp1.add(jtf);

jp2.add(jlb2);

jp2.add(jpf);

jp3.add(jb1);

jp3.add(jb2);

//加入到JFrame

this.add(jp1);

this.add(jp2);

this.add(jp3);

//设置窗体

this.setTitle("用户登录");//窗体标签

this.setSize(300, 150);//窗体大小

this.setLocationRelativeTo(null);//在屏幕中间显示(居中显示)

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//退出关闭JFrame

this.setVisible(true);//显示窗体

//锁定窗体

this.setResizable(false);

}

}

------------------------------------------------------------------------------

## Swing组件--复选框/单选框组件[Window007.java]

几个常用组件

在图形用户界面编程中，我们常常会提供用户注册界面，这时候我们就会用到：

1、复选框组件(JCheckBox)

2、单选框组件(JRadioButton)

**特别说明：**

**同一组单选按钮必需先创建ButtonGroup，然后把单选框组件放入到ButtonGroup中**

/\*\*

\* 复选框组件(JCheckBox)和单选框组件(JRadioButton)使用

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window007 extends JFrame{

//定义组件

JPanel jp1,jp2,jp3;

JLabel jl1,jl2;

JCheckBox jcb1,jcb2,jcb3;

JRadioButton jrb1,jrb2;

ButtonGroup bg;

JButton jb1,jb2;

public static void main(String[] args) {

Window007 win=new Window007();

}

//构造函数

public Window007(){

//创建组件

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

jl1=new JLabel("你最喜欢的运功:");

jl2=new JLabel("你的性别:");

jcb1=new JCheckBox("足球");

jcb2=new JCheckBox("篮球");

jcb3=new JCheckBox("网球");

jrb1=new JRadioButton("男");

jrb2=new JRadioButton("女");

jb1=new JButton("注册用户");

jb2=new JButton("取消注册");

//一定要把jrb1与jrb2放入到一个ButtonGroup中

ButtonGroup bg=new ButtonGroup();

bg.add(jrb1);

bg.add(jrb2);

//创建布局

this.setLayout(new GridLayout(3, 1));

//加入组件

jp1.add(jl1);

jp1.add(jcb1);

jp1.add(jcb2);

jp1.add(jcb3);

jp2.add(jl2);

jp2.add(jrb1);

jp2.add(jrb2);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1);

this.add(jp2);

this.add(jp3);

//设置窗体

this.setTitle("用户注册");

this.setSize(300, 150);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

------------------------------------------------------------------------------

## Swing组件--下拉框/列表框/流动空格组件[Window008.java]

在图形用户界面编程中，我们常常会提供用户调查界面，这个时候我们会用到：

1、下拉框组件(JComboBox)

2、列表框组件(JList)

3、滚动窗格组件(JScrollPane)

**特别说明：**

**一般来说，列表框组件+滚动窗格组件是结合使用的。目的是让列表框中的选项可以有滚动条支持。**

/\*\*

\* 下拉框(JComboBox)/列表框(JList)/滚动空格(JScrollPane)的使用

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window008 extends JFrame{

//定义组件

JPanel jp1,jp2;

JLabel jl1,jl2;

JComboBox jcb;

JList jl;

JScrollPane jsp;

public static void main(String[] args) {

Window008 win=new Window008();

}

//构造函数

public Window008(){

//建立组件

jp1=new JPanel();

jp2=new JPanel();

jl1=new JLabel("你的籍贯:");

jl2=new JLabel("旅游地点:");

String[] jg={"北京","上海","天津","火星"};

jcb=new JComboBox(jg);

String[] dd={"长城","东方明珠","海河","什么什么"};

jl=new JList(dd);

//设置你希望显示多少个选项

jl.setVisibleRowCount(2);//滚动条显示setVisibleRowCount(?)?为显示条数

jsp=new JScrollPane(jl);

//设定布局

this.setLayout(new GridLayout(3, 1));

//加入组件

jp1.add(jl1);

jp1.add(jcb);

jp2.add(jl2);

jp2.add(jsp);

this.add(jp1);

this.add(jp2);

//设置窗体

this.setTitle("用户注册");

this.setSize(300, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

------------------------------------------------------------------------------

Swing组件--总结

**总结：**

**1、本单讲解了Swing的常用组件(JFrame(框架)、Jpanel(面板)、JButton(按钮)、JLable(标签)、JTextField(文本框)、JPasswordField(密码框)、JCheckBox(复选框)、JRadioButton(单选框)、JComboBox(下拉框)、JScrollPane(滚动窗格)、JList(列表框))，当然Swing组件还有很多，在后面使用时讲解。**

**2、三大布局管理器(BorderLayout(边界布局)、FlowLayout(流布局)、GridLayout(网格布局))，要灵活使用，从而满足设计需求。**

## gui编程实践--金山词霸2007

涉及到新的Swing组件：

**1、拆分窗格(JSplitPane)属于容器类组件**[Window009.java]

/\*\*

\* 拆分窗格(JSplitPane)的使用

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window009 extends JFrame{

//定义组件

JSplitPane jsp;

JList jl;

JLabel jlb;

public static void main(String[] args) {

Window009 win=new Window009();

}

//构造函数

public Window009(){

//建立组件

String[] words={"boy","girl","bird","close"};

jl=new JList(words);

jlb=new JLabel(new ImageIcon("images/cb.jpg"));//访问时路径\\或/

//拆分窗格

jsp=new JSplitPane(JSplitPane.HORIZONTAL\_SPLIT,jl,jlb);

//可以收缩变化

jsp.setOneTouchExpandable(true);

//设定布局管理器(JFrame本身就是BorderLayout布局)

//加入组件

this.add(jsp);

//设置窗体

this.setTitle("金山词霸2007");

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

------------------------------------------------------------------------------

## gui编程实战--qq聊天界面

涉及到新的Swing组件

**1、多行文本框(JTextArea)组件**[Window010.java]

/\*\*

\* 多行文本框(JTextArea)组件使用

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window010 extends JFrame{

//定义组件

JTextArea jta=null;

JScrollPane jsp=null;

JPanel jp1=null;

JComboBox jcb=null;

JTextField jtf=null;

JButton jb=null;

public static void main(String[] args) {

Window010 win=new Window010();

}

//构造函数

public Window010(){

//构建组件

jta=new JTextArea();

jsp=new JScrollPane(jta);

jp1=new JPanel();

String[] chatter={"布什","拉登"};

jcb=new JComboBox(chatter);

jtf=new JTextField(10);

jb=new JButton("发送");

//设定布局管理器

//加入组件

jp1.add(jcb);

jp1.add(jtf);

jp1.add(jb);

this.add(jsp);

this.add(jp1,BorderLayout.SOUTH);

//设定窗体

this.setTitle("QQ聊天窗口");

this.setIconImage((new ImageIcon("images\\qe.jpg")).getImage());//设置标题栏内图标

this.setSize(300, 200);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

this.setResizable(false);

}

}

------------------------------------------------------------------------------

## gui编程实战--QQ登录界面

涉及到新的Swing组件：

**1、页签组件(JTabbedPane)选项卡窗格**[Window011.java]

/\*\*

\* 页签组件(JTabbedPane)选项卡窗格--使用QQ登录界面

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window011 extends JFrame{

//设定组件

//主页面北部

JLabel jl1;//标签

//主页面南部

JButton jb1,jb2,jb3;//按钮

JPanel jp1;//面板

//主页面中部

JTabbedPane jtp;//选项卡窗格

JPanel jp2,jp3,jp4;//面板

JLabel jl2,jl3,jl4,jl5;//标签

JTextField jtf;//文本框

JPasswordField jpf;//密码框

JButton jb4;//按钮

JCheckBox jcb1,jcb2;//复选框

//JPanel2组件

JLabel jl6,jl7;//标签

JTextField jtf1;

JPasswordField jpf1;

//JPanel3组件

JLabel jl8,jl9;//标签

JTextField jtf2;

JPasswordField jpf2;

public static void main(String[] args) {

Window011 win=new Window011();

}

//构造函数

public Window011(){

//创建组件

//创建JFrame北部JLabel1组件

jl1=new JLabel(new ImageIcon("images/qqdl.jpg"));

//创建JFrame中部JPanel2组件

jl2=new JLabel("QQ号码",JLabel.CENTER);

jl3=new JLabel("QQ密码",JLabel.CENTER);

jl4=new JLabel("忘记密码",JLabel.CENTER);

jl4.setFont(new Font("宋体",Font.PLAIN,16));//设置字体，字号

jl4.setForeground(Color.BLUE);//设置字体颜色

jl5=new JLabel("<html><a href='www.qq.com'>申请密码保护</a></html>");

jl5.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));//鼠标移动到jl5后会变成手指图标

jtf=new JTextField();

jpf=new JPasswordField();

jb4=new JButton("清除号码");//JButton上可以放图片。new JButton(new ImageIcon("图片路径"));

jcb1=new JCheckBox("隐身登录");

jcb2=new JCheckBox("记住密码");

jtp=new JTabbedPane();//选项卡窗格

jp2=new JPanel();

jp3=new JPanel();

//jp3.setBackground(Color.RED);//给面板设置背景色

jp4=new JPanel();

//jp4.setBackground(new Color(0,0,255));

//创建JFrame中部JTabbedPane选项卡JPanel3组件

jl6=new JLabel("手机号码",JLabel.CENTER);

jl7=new JLabel("密　　码",JLabel.CENTER);

jtf1=new JTextField(20);

jpf1=new JPasswordField(20);

//创建JFrame中部JTabbedPane选项卡JPanel4组件

jl8=new JLabel("电子邮箱",JLabel.CENTER);

jl9=new JLabel("密　　码",JLabel.CENTER);

jtf2=new JTextField(20);

jpf2=new JPasswordField(20);

//创建JFrame南部JPanel1组件

jp1=new JPanel();

jb1=new JButton("登 录");

jb2=new JButton("取 消");

jb3=new JButton("注册向导");

//设置布局管理器

jp2.setLayout(new GridLayout(3, 3));

//jp3.setLayout(new GridLayout(2, 2));

//jp4.setLayout(new GridLayout(2, 2));

//加入组件

//将组件添加到JPanel2中

jp2.add(jl2);//加入QQ号码标签

jp2.add(jtf);//加入文本框

jp2.add(jb4);//加入清除号码按扭

jp2.add(jl3);//加入QQ密码标签

jp2.add(jpf);//加入密码框

jp2.add(jl4);//加入忘记密码

jp2.add(jcb1);//加入隐身登陆复选框

jp2.add(jcb2);//加入记住密码复选框

jp2.add(jl5);//加入申请密码保护标签

//将组件添加到JPanel3中

jp3.add(jl6);

jp3.add(jtf1);

jp3.add(jl7);

jp3.add(jpf1);

//将组件添加到JPanel4中

jp4.add(jl8);

jp4.add(jtf2);

jp4.add(jl9);

jp4.add(jpf2);

//添加到JPanel1中

jp1.add(jb1);

jp1.add(jb2);

jp1.add(jb3);

//将面板添加到选项卡窗格上

jtp.add("QQ号码",jp2);//第一个参数代表选项卡名称，第二个参数代表对应的面板

jtp.add("手机号码",jp3);

jtp.add("电子邮箱",jp4);

//将JLabel1添加到Frame北部

this.add(jl1,BorderLayout.NORTH);

//将JPanle2添加到Frame中部

this.add(jtp,BorderLayout.CENTER);

//将JPanel1添加到Frame南部

this.add(jp1,BorderLayout.SOUTH);

//窗体设置

this.setTitle("QQ登录界面");

//ImageIcon icon=new ImageIcon("图片路径");

//this.setIconImage(icon.getImage());

this.setIconImage((new ImageIcon("images/qe.jpg")).getImage());

this.setSize(350, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

this.setResizable(false);

}

}

------------------------------------------------------------------------------

## gui编程实战--记事本界面[Window012.java]

涉及到新的Swing组件

/\*\*

\* 1、菜单组件

JMenuBar 菜单条组件 树干

JMenu 菜单组件 树枝

JMenuItem 菜单项组件 树叶

2、二级菜单制作

JMenu里面可以嵌套JMenu

3、工具条组件

JToolBar 容器类组件

\*/

import java.awt.\*;

import java.awt.event.InputEvent;

import java.awt.event.InputMethodEvent;

import java.awt.event.KeyEvent;

import javax.swing.\*;

public class Window012 extends JFrame{

//文件组定义组件

JMenuBar jmb;//菜单条组件

JMenu menu1,menu2,menu3,menu4,menu5;//主菜单：文件、编辑、格式、查看、帮助

JMenuItem item2,item3,item4,item5,item6,item7;//子菜单：新建、打开、保存、另存为、页面设置、打印、退出

JMenu xinjian;//二级菜单

JMenuItem file,project;

//编辑组定义组件

//编辑：撤消、剪切、复制、粘贴、删除、查找、查找下一个、替换、转到、全选、时间/日期

JMenuItem eitem1,eitem2,eitem3,eitem4,eitem5,eitem6,eitem7,eitem8,eitem9,eitem10,eitem11;

//格式组定义组件

JMenuItem oitem1,oitem2;//格式：自动换行、字体

//查看组定义组件

JMenuItem vitem1;//查看：状态栏

//帮助组定义组件

JMenuItem hitem1,hitem2;//帮助：查看帮助、关于记事本

JTextArea jta;

//工具条

JToolBar jtb;

JButton jb1,jb2,jb3,jb4,jb5,jb6,jb7,jb8;

public static void main(String[] args) {

Window012 win=new Window012();

}

//构造函数

public Window012(){

//构建组件

//工具条

jtb=new JToolBar();

jb1=new JButton(new ImageIcon("images/new.jpg"));

jb1.setToolTipText("新建");//设置提示信息

jb2=new JButton(new ImageIcon("images/open.jpg"));

jb2.setToolTipText("打开");

jb3=new JButton(new ImageIcon("images/save.jpg"));

jb3.setToolTipText("保存");

jb4=new JButton(new ImageIcon("images/copy.jpg"));

jb4.setToolTipText("复制");

jb5=new JButton(new ImageIcon("images/delete.jpg"));

jb5.setToolTipText("删除");

jb6=new JButton(new ImageIcon("images/modify.jpg"));

jb6.setToolTipText("编辑");

jb7=new JButton(new ImageIcon("images/print.jpg"));

jb7.setToolTipText("打印");

jb8=new JButton(new ImageIcon("images/close.jpg"));

jb8.setToolTipText("关闭");

jmb=new JMenuBar();

//主菜单

menu1=new JMenu("文件(F)");

menu1.setMnemonic('F');//设置助记符

menu2=new JMenu("编辑(E)");

menu2.setMnemonic('E');

menu3=new JMenu("格式(O)");

menu3.setMnemonic('O');

menu4=new JMenu("查看(V)");

menu4.setMnemonic('V');

menu5=new JMenu("帮助(H)");

menu5.setMnemonic('H');

//文件--新建--子目录

xinjian=new JMenu("新建");

file=new JMenuItem("文件");

project=new JMenuItem("工程");

item2=new JMenuItem("打开(O)");

item2.setMnemonic('O');

item2.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_O,InputEvent.CTRL\_MASK));//设置ctrl快捷组合键

item3=new JMenuItem("保存(S)");

item3.setMnemonic('S');

item3.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_S,InputEvent.CTRL\_MASK));

item4=new JMenuItem("另存为(A)");

item4.setMnemonic('A');

item5=new JMenuItem("页面设置(U)");

item5.setMnemonic('U');

item6=new JMenuItem("打印(P)");

item6.setMnemonic('P');

item6.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_P,InputEvent.CTRL\_MASK));

item7=new JMenuItem("退出(X)");

item7.setMnemonic('X');

eitem1=new JMenuItem("撤消(U)");

eitem1.setMnemonic('U');

eitem1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_U,InputEvent.CTRL\_MASK));

eitem2=new JMenuItem("剪切(T)");

eitem2.setMnemonic('T');

eitem2.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_T,InputEvent.CTRL\_MASK));

eitem3=new JMenuItem("复制(C)");

eitem3.setMnemonic('C');

eitem3.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_C,InputEvent.CTRL\_MASK));

eitem4=new JMenuItem("粘贴(P)");

eitem4.setMnemonic('P');

eitem4.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_P,InputEvent.CTRL\_MASK));

eitem5=new JMenuItem("删除(L)");

eitem5.setMnemonic('L');

eitem5.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_DELETE,0));

eitem6=new JMenuItem("查找(F)");

eitem6.setMnemonic('F');

eitem6.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_F,InputEvent.CTRL\_MASK));

eitem7=new JMenuItem("查找下一个(N)");

eitem7.setMnemonic('N');

eitem7.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_F3,0));

eitem8=new JMenuItem("替换(R)");

eitem8.setMnemonic('R');

eitem8.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_R,InputEvent.CTRL\_MASK));

eitem9=new JMenuItem("转到(G)");

eitem9.setMnemonic('G');

eitem9.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_G,InputEvent.CTRL\_MASK));

eitem10=new JMenuItem("全选(A)");

eitem10.setMnemonic('A');

eitem10.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_A,InputEvent.CTRL\_MASK));

eitem11=new JMenuItem("时间/日期(D)");

eitem11.setMnemonic('D');

eitem11.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK\_F5,0));

oitem1=new JMenuItem("自动换行(W)");

oitem1.setMnemonic('W');

oitem2=new JMenuItem("字体(F)");

oitem2.setMnemonic('F');

vitem1=new JMenuItem("状态栏(S)");

vitem1.setMnemonic('S');

hitem1=new JMenuItem("查看帮助(H)");

hitem1.setMnemonic('H');

hitem2=new JMenuItem("关于记事本(A)");

hitem2.setMnemonic('A');

jta=new JTextArea();

//设定布局管理器

//加入组件

//将按钮添加到工具条上

jtb.add(jb1);

jtb.add(jb2);

jtb.add(jb3);

jtb.add(jb4);

jtb.add(jb5);

jtb.add(jb6);

jtb.add(jb7);

jtb.add(jb8);

//将菜单项添加到菜单上

xinjian.add(file);

xinjian.add(project);

menu1.add(xinjian);

menu1.add(item2);

menu1.add(item3);

menu1.add(item4);

menu1.addSeparator();//添加分割线

menu1.add(item5);

menu1.add(item6);

menu1.addSeparator();

menu1.add(item7);

menu2.add(eitem1);

menu2.addSeparator();

menu2.add(eitem2);

menu2.add(eitem3);

menu2.add(eitem4);

menu2.add(eitem5);

menu2.addSeparator();

menu2.add(eitem6);

menu2.add(eitem7);

menu2.add(eitem8);

menu2.add(eitem9);

menu2.addSeparator();

menu2.add(eitem10);

menu2.add(eitem11);

menu3.add(oitem1);

menu3.add(oitem2);

menu4.add(vitem1);

menu5.add(hitem1);

menu5.addSeparator();

menu5.add(hitem2);

//将菜单添加到菜单条上

jmb.add(menu1);

jmb.add(menu2);

jmb.add(menu3);

jmb.add(menu4);

jmb.add(menu5);

//将菜单条添加到窗体上

this.setJMenuBar(jmb);

//将工具条添加到窗体上

this.add(jtb,BorderLayout.NORTH);

JScrollPane jsp=new JScrollPane(jta);

jsp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);

this.add(jsp);

//窗体设置

this.setTitle("Java记事本");

this.setSize(500, 400);

this.setLocationRelativeTo(null);

this.setVisible(true);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

-------------------------------------------------------------------------------

**坦克大战游戏演示**

为什么选择这个项目

1、好玩

2、涉及到java各个方面的技术

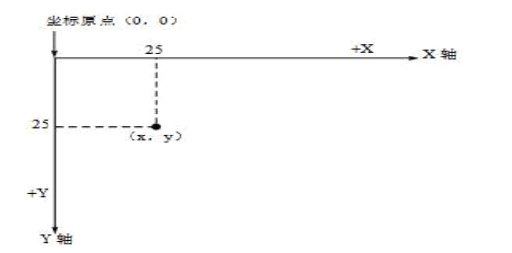
1、java面向对象编程；2、界面编程；3、绘图技术；4、多线程；5、文件i/o操作；6、数据库

3、巩固旧知识，学习新知识

java绘图坐标体系

绘图坐标本系--介绍

坐标原点位于左上角，以像素为单位。像素是计算机屏幕上最小的显示单位。在java的坐标系中，第一个是X坐标，表示当前位置为水平方向，距离坐标原点X个像素；第二个是Y坐标，表示当前位置为垂直方向，距离坐标原点Y个像素。



坐标体系--像素

计算机在屏幕上显示的内容都是由屏幕上的每一个像素组成的。例如，计算机显示器的分辨率是800×600，表示计算机屏幕上的每一行由800个点组成，共有600行，整个计算机屏幕共有480000个像素。现在的计算机可以支持更高的分辨率，也就是说，屏幕上可以显示更多的像素。因此，像素是一个密度单位，而厘米是长度单位，两者无法比较。

绘图原理(1)[Window013.java]

Component类提供了两个和绘图相关最重要的方法：

1、paint(Graphics g)绘制组件的外观

2、repaint()刷新组件的外观

当组件第一次在屏幕显示的时候，程序会自动的调用paint()方法来绘制组件

/\*\*

\* 功能：java绘图原理

\*/

import java.awt.\*;

import javax.swing.\*;

public class Window013 extends JFrame{

//定义组件

MyPanel mp=null;

public static void main(String[] args) {

Window013 th=new Window013();

}

public Thread01(){

//创建组件

mp=new MyPanel();

//加入组件

this.add(mp);

//设置窗体

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

//定义一个MyPanel(我自己的面板是用于绘图和显示绘图的区域)

class MyPanel extends JPanel{

//重写(覆盖)JPanel的paint方法

public void paint(Graphics g){//Graphics是绘图的重要类，可以把它理解成一只画笔

//1、调用父类函数，完成初始化任务

super.paint(g);//**super.paint(g);这句话不能少**

System.out.println("Paint被调用");//用于测试绘图原理2paint调用

//先画一个圆

g.drawOval(10, 10, 30, 30);//drawOval方法是画圆

}

}

------------------------------------------------------------------------------

绘图原理(2)

在以下情况paint()将会被调用：

1、窗口最小化，再最大化

2、窗口的大小发生变化

3、repaint函数被调用

**java绘图技术--Graphics**

Graphics类[Window014.java]

Graphics类你可以理解就是画笔，为我们提供了各种绘制图形的方法：[**多看jdk帮助文档**]

1、画直线 drawLine(int x1,int y1,int x2,int y2);

2、画矩形边框 drawRect(int x,int y,int width,int height);

3、画椭圆边框 drawOval(int x,int y,int width,int height);

4、填充矩形 fillRect(int x,int y,int width,int height);

5、填充椭圆 fillOval(int x,int y,int width,int height);

6、画图片 drawImage(Image img.int x,int y,..);

7、画字符串 drawString(String str,int x,int y);

8、设置画笔的字体 setFont(Font font);

9、设置画笔的颜色 setColor(Color c);

//java画图Graphics绘图类的使用方法

import java.awt.\*;

import javax.swing.\*;

public class Window014 extends JFrame{

//定义组件

MyPanel1 mp=null;

public static void main(String[] args) {

Window014 th=new Window014();

}

public Window014(){

//构建组件

mp=new MyPanel1();

//加入组件

this.add(mp);

//设置窗体

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

//定义一个MyPanel(我自己的面板是用于绘图和显示绘图的区域)

class MyPanel1 extends JPanel{

//重写(覆盖)JPanel的paint方法

public void paint(Graphics g){//Graphics是绘图的重要类，可以把它理解成一只画笔

//1、调用父类函数，完成初始化任务

super.paint(g);

//画一个直线

g.drawLine(10, 10, 40, 10);

//画矩形边框

g.drawRect(50, 50, 40, 40);

//画椭圆边框

g.drawOval(100, 100, 60, 60);

//画填充矩形

g.setColor(Color.blue);

g.fillRect(10, 150, 70, 70);

//画填充椭圆

g.setColor(Color.red);//设置画笔颜色

g.fillOval(200, 50, 80, 80);

}

}

------------------------------------------------------------------------------

//画图Graphics画图类--画图片--画字[Window015.java]

import java.awt.\*;

import javax.swing.\*;

public class Window015 extends JFrame{

//定义组件

MyPanel1 mp=null;

public static void main(String[] args) {

Window015 th=new Window015();

}

public Window015(){

//构建组件

mp=new MyPanel1();

//加入组件

this.add(mp);

//设置窗体

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

//定义一个MyPanel(我自己的面板是用于绘图和显示绘图的区域)

class MyPanel1 extends JPanel{

//重写(覆盖)JPanel的paint方法

public void paint(Graphics g){//Graphics是绘图的重要类，可以把它理解成一只画笔

//1、调用父类函数，完成初始化任务

super.paint(g);

//在面板上画出图片

Image im=Toolkit.getDefaultToolkit().getImage(Panel.class.getResource("/F22.jpg"));

//实现

g.drawImage(im, 0, 0, 300, 200, this);

//如何画出字

g.setColor(Color.red);

g.setFont(new Font("黑体",Font.BOLD,40));

g.drawString("祖国万岁", 100, 100);

//画弧形

g.drawArc(100, 100, 120, 200, -50, -100);

}

}

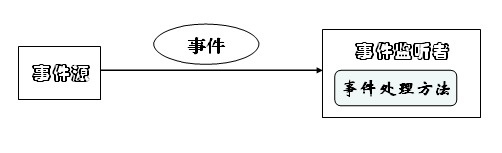
-------------------------------------------------------------------------------

# java事件处理机制

如何通过键盘或鼠标的控制使Java程序中的功能有响应?

java事件处理机制--初步介绍

java事件处理是采取“委派事件模型”。所谓“委派事件模型”是指当事件发生时，产生事件的对象(即事件源)，会把此“信息”传递给“事件的监听者”处理的一种方式，而这里所说的“信息”实际上就是java.awt.event事件类库里某个类所创建的对象，我们暂时把它称为“事件的对象”。



比如：当按下按钮时，就会在按钮触发一个“事件”，程序就会产生一个“事件对象”来表示这个事件，并把这个“事件对象”传递给“事件的监听者”，“事件的监听者”再根据“事件对象”的类型进行相应处理。

为了让“产生事件的对象”[比如按钮]知道要把事件传递给哪一个“事件监听者”，我们必需先把“事件监听者”在“产生事件的对象”注册，这个操作也就是要告知“产生事件的对象”要把事件传递给它。



实例[Window016.java]

/\*\*

\* 功能:事件处理机制

\*/

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Window016 extends JFrame implements ActionListener{

//定义组件

JPanel mp=null;

JButton jb1,jb2;

public static void main(String[] args) {

Window016 win=new Window016();

}

//构造函数

public Window016(){

//创建组件

mp=new JPanel();

jb1=new JButton("黑色");

jb2=new JButton("红色");

//设定布局管理器

//加入组件

mp.setBackground(Color.black);

this.add(mp);

this.add(jb1,BorderLayout.NORTH);

this.add(jb2,BorderLayout.SOUTH);

//猫类在监听

Cat mycat1=new Cat();

jb1.addActionListener(mycat1);

jb2.addActionListener(mycat1);

//注册监听

jb1.addActionListener(this);

jb2.addActionListener(this);

//指定action命令

jb1.setActionCommand("黑色");

jb2.setActionCommand("红色");

//JFrame窗体设置

this.setTitle("事件处理机制");

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setVisible(true);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

//对事件处理的方法

public void actionPerformed(ActionEvent e) {

//判断是哪个按钮被点击

if(e.getActionCommand().equals("黑色")){

System.out.println("点击了黑色按钮");

mp.setBackground(Color.BLACK);

}else if(e.getActionCommand().equals("红色")){

System.out.println("点击了红色按钮");

mp.setBackground(Color.RED);

}else{

System.out.println("不知道");

}

}

}

class Cat implements ActionListener{

public void actionPerformed(ActionEvent arg0) {

if(arg0.getActionCommand().equals("黑色")){

System.out.println("Cat也知道你按下了黑色按钮");

}else if(arg0.getActionCommand().equals("红色")){

System.out.println("Cat也知道你按下了红色按钮");

}else {

System.out.println("Cat也不知道");

}

}

}

------------------------------------------------------------------------------

实例[Window017.java]

/\*\*

\* 功能：加深对事件处理机制的理解

\* 1、通过上下左右键，来控制一个小球的移动。

\*/

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Window017 extends JFrame {

//设定组件

MyPanels mp=null;

public static void main(String[] args) {

Window017 win=new Window017();

}

//构造函数

public Window017(){

//构建组件

mp=new MyPanels();

//监听

this.addKeyListener(mp);

//加入组件

this.add(mp);

//设置窗体

this.setTitle("键盘事件监听");

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setVisible(true);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

//定义自己的面板

class MyPanels extends JPanel implements KeyListener{

int x=10,y=10;

public void paint(Graphics g){

super.paint(g);

g.fillOval(x, y, 10, 10);

}

public void keyPressed(KeyEvent e) {//keyPressed代表键被按下

System.out.println("被按下"+((char)e.getKeyCode()));

if(e.getKeyCode()==KeyEvent.VK\_DOWN){

System.out.println("向下");

y++;

}else if(e.getKeyCode()==KeyEvent.VK\_UP){

System.out.println("向上");

y--;

}else if(e.getKeyCode()==KeyEvent.VK\_LEFT){

System.out.println("向左");

x--;

}else if(e.getKeyCode()==KeyEvent.VK\_RIGHT){

System.out.println("向右");

x++;

}

//调用repaint()函数，来重绘界面

this.repaint();

}

public void keyReleased(KeyEvent e) {//keyReleased代表键被弹起

}

public void keyTyped(KeyEvent e) {//keyTyped代表具体的值被输出

}

}

------------------------------------------------------------------------------

## java事件处理机制--深入理解

前面提到了几个重要的概念

1、事件源；2、事件；3、事件监听器

这几个概念非常重要，必需要搞清楚。

**1、事件源**

事件源是一个产生(或触发)事件的对象，比如前面的JButton的一个对象jb1。当这个事件源对象的某些状态以某种方式发生变化时，就会产生某种类型的事件(一个事件源可能会生成多个不同类型的事件)。如果某个组件(对象)希望得到事件源产生的事件，就需要在这个事件源上注册。

**2、事件**

事件就是承载事件源状态改变时的信息对象。或者说，事件是事件源事件监听器传输事件源状态信息的载体。在用户与GUI组件进行交互时就会生成事件，比如当鼠标在面板中移动时，就会生成一个鼠标移动事件的对象，而这个对象保存着当前鼠标在面板中位置信息。java.awt.event包和javax.swing.event包中定义了各种事件类型，常见的事件类有：

|  |  |
| --- | --- |
| 事件类 | 说明 |
| ActionEvent | 通常在按下按钮，或双击一个列表项或选中某个菜单时发生 |
| AdjustmentEvnet | 当操作一个滚动条时发生 |
| ComponentEvent | 当一个组件隐藏、移动、改变大小时发送 |
| ContainerEvent | 当一个组件从容器中加入或者删除时发生 |
| FocusEvent | 当一个组件获得或是失去焦点时发生 |
| ItemEvent | 当一个复选框或是列表项被选中时，当一个选择框或选择菜单被选中 |
| KeyEvent | 当从键盘的按键被按下，松开时发生 |
| MouseEvent | 当鼠标被拖动、移动、点击、按下.. |
| TextEvent | 当文本区和文本域的文本发生改变时发生 |
| WindowEvent | 当一个窗口激活、关闭、失效、恢复、最小化... |



**3、事件监听器接口**

事件源产生一个事件，可以传送给事件监听者处理，那么怎样才能编写一个事件监听者呢？

事件监听者实际上就是一个类，该类实现了某个事件监听器接口、比如前面我们案例中的MyPanel就是一个类，它实现了ActionListener接口，它就可以作为一个事件监听者，对接受到的事件进行处理。

事件监听器接口有多种，不同的事件监听器接口可以监听不同的事件，一个类可以实现一个事件监听接口，也可以实现多个监听接口，这些接口在java.awt.event和javax.swing.event包中定义。

**java事件处理机制--总结**

总结--事件编程步骤：

1、编写事件处理类(事件监听者)

2、根据需求给事件处理类实现监听器接口

3、在事件处理中重写(实现)其事件处理的函数

4、在事件源类中指定该事件的监听器(响应者)是谁

**注意事项：**

1、java采用委托机制处理事件

2、java中的事件是分类的，比如对窗体事件、鼠标事件、按键事件、操作事件[按按钮]

3、java中一个类要监听某个事件，则必需实现相应的事件监听接口

4、事件监听接口有多种，程序员应当针对不同的情况，实现不同的监听接口，比如监听鼠标事件就应当实现MouseListener;要监听键盘事件，就应当实现KeyListener..

5、在实现监听接口的类(事件监听类/者)中，需要重写处理函数，比如实现了ActionListener接口，就应当重写actionPerformed(ActionEvent e),可以参考前面的事件监听器接口表格

6、在事件源中需要注册事件监听类。否则事件监听类接收不到事件源发生的事件。

**一个类要实现监听的步骤：**

a、实现相应的接口[KeyListener/MouseListener/ActionListener/WindowListener]

b、把接口的处理方法根据需要重新编写(Override)

c、在事件源上注册监听

d、事件传递是靠事件(类)对象

实例[Window018.java]

/\*\*

\* 单事件源、多事件监听，多事件处理

\*/

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Window018 extends JFrame{

//设定组件

MyPanel mp=null;

public static void main(String[] args) {

Window018 win=new Window018();

}

public Window018(){

//创建组件

mp=new MyPanel();

//注册监听

this.addMouseListener(mp);

this.addMouseMotionListener(mp);

this.addKeyListener(mp);

this.addWindowListener(mp);

//加入组件

this.add(mp);

//设置窗体

this.setTitle("事件多监听多处理");

this.setSize(400, 300);

this.setLocationRelativeTo(null);

this.setVisible(true);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

//1、让MyPanel知道鼠标按下的消息，并且知道点击的位置(x,y)

//2、让MyPanel知道哪个键按下了

class MyPanel extends JPanel implements MouseListener, MouseMotionListener,KeyListener,WindowListener{

public void paint(Graphics g){

super.paint(g);

}

//鼠标点击(mouseClicked)

public void mouseClicked(MouseEvent e) {

System.out.println("鼠标点击了x="+e.getX()+"y="+e.getY());

}

//鼠标移动到MyPanel(mouseEntered)

public void mouseEntered(MouseEvent e) {

System.out.println("鼠标移动到MyPanel上了");

}

//鼠标离开MyPanel(mouseExited)

public void mouseExited(MouseEvent e) {

System.out.println("鼠标离开MyPanel上了");

}

//鼠标按下去(mousePressed)

public void mousePressed(MouseEvent e) {

System.out.println("鼠标被按下了");

}

//鼠标松开(mouseReleased)

public void mouseReleased(MouseEvent e) {

System.out.println("鼠标被松开了");

}

//鼠标拖拽(mouseDragged)

public void mouseDragged(MouseEvent e) {

System.out.println("鼠标拖拽x="+e.getX()+"y="+e.getY());

}

//鼠标移动(mouseMoved)

public void mouseMoved(MouseEvent e) {

System.out.println("鼠标移动时X="+e.getX()+"Y="+e.getY());

}

//键输入值(keyTyped)，F(1-12)无响应

public void keyTyped(KeyEvent e) {

System.out.println("按下了"+e.getKeyChar()+"键");

}

//键按下(keyPressed)

public void keyPressed(KeyEvent e) {

System.out.println("按下了"+e.getKeyChar()+"键");

}

//键松开(keyReleased)

public void keyReleased(KeyEvent e) {

System.out.println("按下了"+e.getKeyChar()+"键");

}

//打开窗口(windowOpened)

public void windowOpened(WindowEvent e) {

System.out.println("打开窗口(windowOpened)");

}

//窗口关闭(windowClosing)

public void windowClosing(WindowEvent e) {

System.out.println("窗口关闭(windowClosing)");

}

//窗口关闭(windowClosed)

public void windowClosed(WindowEvent e) {

System.out.println("窗口关闭(windowClosed)");

}

//窗口最小化(windowIconified)

public void windowIconified(WindowEvent e) {

System.out.println("窗口最小化(windowIconified)");

}

//恢复窗口(windowDeiconified)

public void windowDeiconified(WindowEvent e) {

System.out.println("恢复窗口(windowDeiconified)");

}

//激活窗口，使用窗口(windowActivated)

public void windowActivated(WindowEvent e) {

System.out.println("激活窗口(windowActivated)");

}

//窗口停用,切换窗口(windowDeactivated)

public void windowDeactivated(WindowEvent e) {

System.out.println("窗口停用(windowDeactivated)");

}

}

------------------------------------------------------------------------------

# 线程--什么是进程

## 进程--概念

要解释线程，就必须明白什么是进程，就好象要搞清中国历史，就必须要了解春秋战国。

什么是进程呢？

进程是指运行中的应用程序，每个进程都有自己独立的地址空间(内存空间)，比如用户点击桌面的IE浏览器，就启动了一个进程，操作系统就会为该进程分配独立的地址空间。当用户再次点击左面的IE浏览器，又启动了一个进程，操作系统将为新的进程分配新的独立的地址空间。目前操作系统都支持多进程。

**要点：**用户每启动一个进程，操作系统就会为该进程分配一个独立的内存空间。

## 线程--概念

在明白进程后，就比较容易理解线程的概念。

什么是**线程**呢？

是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。线程有就绪、阻塞和运行三种基本状态。

## 线程

1、线程是轻量级的进程

2、线程没有独立的地址空间(内存空间)

3、线程是由进程创建的(寄生在进程)

4、一个进程可以拥有多个线程-->这就是我们常说的多线程编程

5、线程有几种状态：

a、新建状态(new)

b、就绪状态(Runnable)

c、运行状态(Running)

d、阻塞状态(Blocked)

e、死亡状态(Dead)

## 线程有什么用处

java程序中流传一句话，不会使用线程就别跟别人说自己学过java。目前绝大部分应用程序都会涉及到多并发的问题。只要应用程序涉及到并发，就离不开多线程编程。

## 线程--如何使用

**在java中一个类要当作线程来使用有两种方法**。

**1、继承Thread类，并重写run函数**

**2、实现Runnable接口，并重写run函数**

因为java是单继承的，在某些情况下一个类可能已经继承了某个父类，这时在用继承Thread类方法来创建线程显然不可能,java设计者们提供了另外一个方式创建线程，就是通过实现Runnable接口来创建线程。

通过继承Thread类来实现建立线程实例[Thread01.java]

/\*\*

\* 演示如何通过继承Thread来开发线程

\*/

public class Thread01 {

public static void main(String[] args) {

//创建一个 Cat对象

Cat cat=new Cat();

//启动线程

cat.start();//.start()会导致run函数运行

}

}

class Cat extends Thread{

int times=0;

//重写run函数

public void run(){

while(true){

//休眠一秒

//1000表示1000毫秒

try {

Thread.sleep(1000);//sleep就会让该线程进入到Blocked阻塞状态,并释放资源。

} catch (Exception e) {

e.printStackTrace();

}

times++;

System.out.println("hello,world!"+times);

if(times==10){

//退出线程

break;

}

}

}

}

-------------------------------------------------------------------------------

通过Runnable接口来实现建立线程实例[Thread02.java]

/\*\*

\* 演示如何通过Runnable接口来开发线程

\*/

public class Thread02{

public static void main(String []args){

Dog dog=new Dog();

//创建线程

Thread t=new Thread(dog);

//启动线程

t.start();

}

}

class Dog implements Runnable{//创建Runnable接口

public void run(){//重写run函数

int times=0;

while(true){

try{

Thread.sleep(1000);

}catch (Exception e) {

e.printStackTrace();

}

times++;

System.out.println("hello,wrold!"+times);

if(times==10){

break;

}

}

}

}

-------------------------------------------------------------------------------

多线程实例演示[Thread03.java]

/\*\*

\*多线程Thread使用

\*1、一个线程通过接收n来执行1+..+n得到和

\*2、另一线程每隔1秒输出一次hello world!

\*/

public class Thread03 {

public static void main(String[] args) {

Pig pig=new Pig(10);

Bird bird=new Bird(10);

//建立线程

Thread t1=new Thread(pig);

Thread t2=new Thread(bird);

//启动线程

t1.start();

t2.start();

}

}

//打印hello world!

class Pig implements Runnable{

int n=0;

int times=0;

public Pig(int n){

this.n=n;

}

public void run(){

while(true){

try {

Thread.sleep(1000);

} catch (Exception e) {

e.printStackTrace();

}

times++;

System.out.println("我是一个线程，正在输出第"+times+"个hello world!");

if(times==n){

break;

}

}

}

}

//算数学题

class Bird implements Runnable{//多线程时应使用implements接口来实现，不要使用extends继承

int n=0;

int res=0;

int times=0;

public Bird(int n){

this.n=n;

}

public void run() {

while(true){

try {

Thread.sleep(1000);

} catch (Exception e) {

e.printStackTrace();

}

res+=(++times);

System.out.println("当前结果是："+res);

if(times==n){

System.out.println("最后的结果是："+res);

break;

}

}

}

}

-------------------------------------------------------------------------------

## 线程--区别

**线程--继承Thread VS 实现Runnable的区别**

从java的设计来看，通过继承Thread或者实现Runnable接口来创建线程本质上没有区别，从jdk帮助文档我们可以看到**Thread类本身就实现了Runnable接口**，如果一定要说它们有什么**区别，总结几点**：

1、尽可能使用实现Runnable接口的方式来创建线程

2、在使用Thread的时候只需要new一个实例出来，调用start()方法即可以启动一个线程，如： Thread test=new Thread();

test.start();

3、在使用Runnable的时候需要先new一个实现Runnable的实例，之后用Thread调用，如:

Test implements Runnable

Test t=new Test();

Thread test=new Thread(t);

tset.start();

## 线程--深入理解

线程对象只能启动一个线程，见[Thread04.java]

/\*\*

\* 功能:使用线程的注意事项

\* 不论继承Thread或实现Rnunable接口都不能使用start启同一个线程2次

\*/

public class Thread04 {

public static void main(String[] args) {

Cat cat1=new Cat();

cat1.start();

//cat1.start();同一个线程，不能启动2次

Dog dog1=new Dog();

Thread t=new Thread(dog1);

t.start();

//t.start();同一个线程，不能启动2次

}

}

//猫类

class Cat extends Thread{

public void run(){

System.out.println("11");

}

}

//狗类

class Dog implements Runnable{

public void run(){

System.out.println("2");

}

}

-------------------------------------------------------------------------------

**结论：**不管是通过继承Thread，还是通过实现Runnable接口创建线程，它们的一个对象只能启动(即：start())一次。否则就会有异常抛出。

## 两种创建线程的方法的区别

创建线程有两种方法：1、继承Thread；2、实现Runnable接口；这两种方法有什么区别？

**用实现Runnable接口的特点**

1、用实现Runnable接口的方法创建对象可以避免java单继承机制带来的局限；

2、用实现Runnable接口的方法，可以实现多个线程共享同一段代码(数据)；

因此建议大家如果你的程序有同步逻辑需求，则使用Runnable的方法来创建线程。

## java线程的同步--提出问题

多线程的并发，给我们编程带来很多好处，完成更多更有效率的程序。但是也给我们带来线程安全问题。

## java线程的同步--解决问题

解决问题的关键就是要保证容易出问题的代码的原子性，**所谓原子性就是指：当a线程在执行某段代码的时候，别的线程必须等到a线程执行完后，它才能执行这段代码。**也就是排队一个一个解决。

java处理线程两步的方法非常简单，只需要在需要同步的代码段，用:

**synchronized(Object){你要同步的代码}**

即可。

售票案例演示[Thread05.java]

/\*\*

\* 功能:使用线程的注意事项

\* 线程并发同步锁synchronized(Object){}的使用

\*/

public class Thread05 {

public static void main(String[] args) {

//定义一个售票窗口

TicketWindow tw1=new TicketWindow();

//使用三个线程同时启动

Thread t1=new Thread(tw1);

Thread t2=new Thread(tw1);

Thread t3=new Thread(tw1);

t1.start();

t2.start();

t3.start();

}

}

//售票窗口类

class TicketWindow implements Runnable {

//共有2000张票

private int nums=2000;

private Dog myDog=new Dog();

public void run() {

while(true){

//出票速度是1秒出一张

try {

Thread.sleep(1000);

} catch (Exception e) {

e.printStackTrace();

}

//认为if else要保证其原子性

//先判断是否还有票

synchronized(myDog){**//synchronized(this){}为同步代码块**

if(nums>0){

//显示售票信息

//Thread.currentThread().getName()得到当前线程的名字

System.out.println(Thread.currentThread().getName()+"正在售出第"+nums+"张票");

nums--;

}else{

//售票结束

break;

}

}

}

}

}

class Dog{

}

-------------------------------------------------------------------------------

## java线程的同步--解决问题

对同步机制的解释：

java任意类型的对象都有一个标志位，该标志位具有0、1两种状态，其开始状态为1，当某个线程执行了synchronized(Object)语句后，object对象的标志位变为0的状态，直到执行完整个synchronized语句中的代码块后，该对象的标志位又回到1状态。

当一个线程执行到synchronized(Object)语句的时候，先检查Object对象的标志位，如果为0状态，表明已经有另外的线程正在执行synchronized包括的代码，那么这个线程将暂时阻塞，让出CPU资源，直到另外的线程执行完相关的同步代码，并将Object对象的标志位变为状态，这个线程的阻塞就被取消，线程能继续运行，该线程又将Object的标志位变为0状态，防止其它的线程再进入相关的同步代码块中。

如果有多个线程因等待同一个对象的标志位面而处于阻塞状态时，当该对象的标志位恢复到1状态时，只会有一个线程能够进入同步代码执行，其它的线程仍处于阻塞的状态。

**特别说明：**

1、上面所说的标志位用术语讲就是对象锁,文件锁。数据库会有行锁、表锁等

2、synchronized(object)//object(就是对象锁)可以是任意类型对象

# 文件

**--什么是文件**

文件，对我们并不陌生，文件是数据源(保存数据的地方)的一种，比如大家经常使用的word文档、txt文件、excel文件...都是文件。文件最主要的作用就是保存数据，它既可以保存一张图片，也可以保存视频、声音...等

## 文件流--基本概念

文件在程序中是以流的形式来操作的。



流：数据在数据源(文件)和程序(内存)之间经历的路径

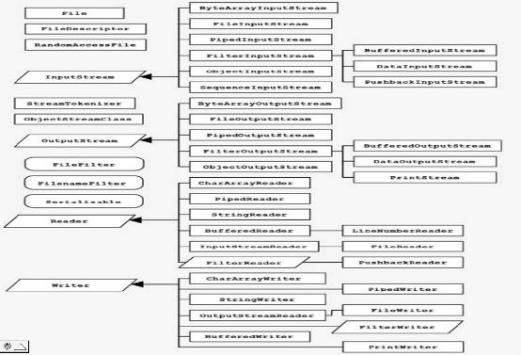
输入流：数据从数据源(文件)到程序(内存)的路径

输出流：数据从程序(内存)到数据源(文件)的路径

如何判断是输入流、输出流？

以内存为参照，如果数据流向内存流动，则是输入流；反之，则是输出流。

文件流--分类



**java流分为两种流**

1、字节流：可以用于读写二进制文件及任何类型文件byte

2、字符流：可以用于读写文本文件，不能操作二进制文件

**字节流 字符流**

**输入 InputStream Reader**

**输出 OutputStream Writer**

常用io流--文件对象

目的：文件数据源File类介绍(文件流对象中最为重要的File类，对File了解后对子类理解会更加容易)

**File类--使用**[Io01.java]

/\*\*

\* File类的基本用法

\*/

import java.io.\*;//必需加载IO包

public class Io01 {

public static void main(String[] args) {

//创建一个文件对象

File f1=new File("e:\\aa.txt");

//得到文件的路径

System.out.println("文件路径"+f1.getAbsolutePath());

//得到文件的大小,字节数

System.out.println("文件的大小"+f1.length());

//创建文件夹

File f3=new File("e:\\ff");

//判断文件夹是否存在

if(f3.isDirectory()){

System.out.println("文件夹存在，不能创建!");

}else{

//创建文件夹

f3.mkdir();

}

//创建文件和创建文件夹

File f2=new File("e:\\ff\\hsp.txt");

//判断文件是否存在

if(!f2.exists()){

//可以创建

try {

f2.createNewFile();//创建一个新文件

} catch (Exception e) {

e.printStackTrace();

}

}else{

System.out.println("文件存在，不能创建!");

}

//列出一个文件夹下面的所有文件

File f4=new File("e:\\ff");

//判断文件夹是事存在

if(f4.isDirectory()){

//将文件夹的文件，传给lists数组

File lists[]=f4.listFiles();

//遍历数组

for(int i=0;i<lists.length;i++){

//输出文件夹下所有文件文件名

System.out.println("显示出文件名是"+lists[i].getName());

}

}

}

}

------------------------------------------------------------------------------

## java文件编程--常用io流

常用io流--文件字节流

1、案例[Io02.java]：读取文件(文件字节输入流使用，目的：FileInputStream类)把用**FileInputStream的对象把文件读入到内存**

/\*\*

\* File类的基本用法

\* io流--文件字节流

\* FileInputStream类的使用

\*/

import java.io.\*;

public class Io02 {

public static void main(String[] args) {

//得到一个文件对象，f指向e:\ff\hsp.txt文件

File f=new File("e:\\ff\\hsp.txt");

FileInputStream fis=null;

try {

//因为**File没有读写的能力，所以需要使用InputStream类**

fis=new FileInputStream(f);

//定义一个字节数组，相当于缓存

byte []bytes=new byte[1024];

int n=0;//得到实际读取到的字节数

//循环读取

while((n=fis.read(bytes))!=**-1**){

//把字节转成String

String s=new String(bytes,0,n);

System.out.println(s);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

//**关闭文件流必需放在finally语句块中**

try {

fis.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

2、案例[Io03.java]：从键盘接收用户输入内容，并保存到文件中(文件字节输出流，目的：**FileOutputStream类**)

/\*\*

\* File类的基本用法

\* io流--文件字节流

\* FileOutputStream类的使用

\*/

import java.io.\*;

public class Io03 {

public static void main(String[] args) {

File f=new File("e:\\ff\\ss.txt");//直接覆盖写同一个文件

//字节输出流

FileOutputStream fos=null;

if(f.exists()){

System.out.println("文件已存在");

}else{

try {

fos=new FileOutputStream(f);

String s="hello,world!\r\n";

String s1="中国人";

fos.write(s.getBytes());

fos.write(s1.getBytes());

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

fos.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

}

}

-------------------------------------------------------------------------------

3、案例[Io04.java]：图片拷贝

/\*\*

\* File类的基本用法

\* io流--文件字节流

\* 图片拷贝--FileInputStream类与 FileOutputStream类

\*/

import java.io.\*;

public class Io04 {

public static void main(String[] args) {

//先将图片读入到内存，再将内存中的图片写入到某个文件

//因为二进制文件只能拿使用字节流来处理

//输入流

FileInputStream fis=null;

//输出流

FileOutputStream fos=null;

try {

fis=new FileInputStream("e:\\ff\\a.jpg");

fos=new FileOutputStream("e:\\a.jpg");

byte buf[]=new byte[1024];

int n=0;//记录实际读取到的字节数

//循环读取图片

while((n=fis.read(buf))!=-1){

//输出到指定文件

fos.write(buf);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

//一定要关闭打开的文件流

try {

fis.close();

fos.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

## java文件编程--常用io流

**常用io流--文件字符流**

案例[Io05.java](文件字符输入、输出流，目的：**FileReader、FileWriter类**)读取一个文件并写入到另一个文件中char[]来中转

/\*\*

\* File类的基本用法

\* io流--文件字符流，只能用于完全为字符的文件

\* TXT文件拷贝--FileReader类与 FileWriter类

\*/

import java.io.\*;

public class Io05 {

public static void main(String[] args) {

//文件取出字符流对象(输入流)

FileReader fr=null;

//写入到文件(输出流)

FileWriter fw=null;

try {

//创建fr对象

fr=new FileReader("e:\\ff\\hsp.txt");

//创建输出对象

fw=new FileWriter("e:\\hsp.txt");

//创建字符数组

char c[]=new char[1024];

int n=0;

//读入到内存

while((n=fr.read(c))!=-1){

//控制台输出TXT文件内容

String s=new String(c,0,n);

System.out.println(s);

fw.write(c, 0, n);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

fr.close();

fw.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

------------------------------------------------------------------------------

## 常用io流--缓冲字符流

为了提高效率引入了缓冲字符流

案例[Io06.java](文件缓冲字符流，目的：**BufferedReader和BufferedWriter类**介绍，直接操作String)

/\*\*

\* File类的基本用法

\* io流--缓冲字符流

\* BufferedReader类与BufferedWriter类

\*/

import java.io.\*;

public class Io06 {

public static void main(String[] args) {

BufferedReader br=null;

BufferedWriter bw=null;

try {

//先创建FileReader对象

FileReader fr=new FileReader("e:\\ff\\hsp.txt");

br=new BufferedReader(fr);

//创建FileWriter对象

FileWriter fw=new FileWriter("e:\\hsp1.txt");

bw=new BufferedWriter(fw);

//循环读取

String s="";

while((s=br.readLine())!=null){

//输出到磁盘

bw.write(s+"\r\n");

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

br.close();

bw.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

## 常用io流--打印输出流PrintWriter

打印输出流PrintWriter可以使用print/println及writer方法。但不换行。需在文本内容中加入\r\n通配符才可以做到。

PrintWriter的使用方法与FileReader、FileWriter/BufferedReader、BufferedWriter基本相同

记事本实例[Io07.java]

/\*\*

\* 我的记事本(界面+功能)

\*/

import java.awt.\*;

import javax.swing.\*;

import java.io.\*;

import java.awt.event.\*;

public class Io07 extends JFrame implements ActionListener{

//定义组件

JTextArea jta=null;//文本框

//菜单条

JMenuBar jmb=null;

//定义第一个JMenu

JMenu jm1=null;

//定义JMenuItem

JMenuItem jmi1=null;

JMenuItem jmi2=null;

JMenuItem jmi3=null;

public static void main(String[] args) {

Io07 io=new Io07();

}

//构造函数

public Io07(){

//创建组件

jta=new JTextArea();

jmb=new JMenuBar();

jm1=new JMenu("文件(F)");

//设置助记符

jm1.setMnemonic('F');

jmi1=new JMenuItem("打开(O)");

//open打开注册监听

jmi1.addActionListener(this);

jmi1.setActionCommand("open");

jmi2=new JMenuItem("保存(S)");

//save保存注册监听

jmi2.addActionListener(this);

jmi2.setActionCommand("save");

jmi3=new JMenuItem("退出(X)");

//exit退出注册监听

jmi3.addActionListener(this);

jmi3.setActionCommand("exit");

//加入到菜单

this.setJMenuBar(jmb);

//把jm1放到jmb

jmb.add(jm1);

//把 jmi1放入jm1

jm1.add(jmi1);

jm1.add(jmi2);

jm1.add(jmi3);

//设置界面管理器(默认BorderLayout边界布局管理器)

//加入组件

this.add(jta);

//设置JFrame面板

this.setTitle("记事本界面与功能");

this.setSize(500, 400);

this.setLocationRelativeTo(null);

this.setVisible(true);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void actionPerformed(ActionEvent e) {

//判断"打开"菜单被选中

if(e.getActionCommand().equals("open")){

/\*\*

\* 隆重推荐JFileChooser组件

\*/

//创建一个文件选择组件

JFileChooser jfc=new JFileChooser();

//设置名字

jfc.setDialogTitle("请选择要打开的文件...");

//使用默认方式

jfc.showOpenDialog(null);

//显示

jfc.setVisible(true);

//得到用户选择的文件全(绝对)路径

String filename=jfc.getSelectedFile().getAbsolutePath();

FileReader fr=null;

BufferedReader br=null;

try {

fr=new FileReader(filename);

br=new BufferedReader(fr);

//从文件中读取信息并显示到jta(JTextArea)中

String s="";

String allCon="";

while((s=br.readLine())!=null){

allCon+=s+"\r\n";//"\r\n"显示文本时将文件中原有的格式显示到jta中

}

//放置到jta即可

jta.setText(allCon);

} catch (Exception e2) {

e2.printStackTrace();

}finally{

try {

br.close();

fr.close();

} catch (Exception e1) {

e1.printStackTrace();

}

}//判断"保存"菜单被选中

}else if(e.getActionCommand().equals("save")){

//创建保存对话框

JFileChooser jfc=new JFileChooser();

//设置名字

jfc.setDialogTitle("将文件保存到...");

//使用默认方式

jfc.showSaveDialog(null);

//显示

jfc.setVisible(true);

//得到用户希望把文件保存到何处，文件全路径

String file=jfc.getSelectedFile().getAbsolutePath();

//准备写入到指定文件

FileWriter fw=null;

BufferedWriter bw=null;

try {

fw=new FileWriter(file);

bw=new BufferedWriter(fw);

//将JtextArea中的内容输出到指定文件中

bw.write(this.jta.getText());

} catch (Exception e2) {

e2.printStackTrace();

}finally{

try {

bw.close();

fw.close();

} catch (Exception e1) {

e1.printStackTrace();

}

}

}else if(e.getActionCommand().equals("exit")){

System.exit(0);

}

}

}

-------------------------------------------------------------------------------

# 作业：

过三关

老孙回到花果山，众猴为迎接大圣的归来，决定列队表演猴接棍，大圣大悦，可一看队伍中高矮参差不齐，皱起眉头说到，先排好队再说，怎么没什么长进呢？

第一关：[Work17.java]

领头的老猴于是从队伍一头开始逐一的比较、交换，开始按高矮排列队伍。(设计冒泡排序的程序给队伍排序)。一柱香后，老猴还在不知疲倦地来回调整，悟空这时已不耐烦了，大叫道：快点!快点!不然我撤了你。

/\*\*

\* 猴子过三关(冒泡排序)

\*/

import java.util.Scanner;

public class Work17 {

public static void main(String[] args) {

//定义可以装五只猴子的对象数组

int size=5;

Monkey []monkeys=new Monkey[size];

Scanner sr=new Scanner(System.in);

//初始化各个猴子

for(int i=0;i<monkeys.length;i++){

System.out.println("请输入第"+(i+1)+"只猴子的高度：");

float height=sr.nextFloat();

monkeys[i]=new Monkey((i+1)+"",height);

}

//让老猴排序

Monkey oldMonkey=new Monkey("1000", 1.2f);

//排序

oldMonkey.bubbleSort(monkeys);

//验证输出

oldMonkey.show(monkeys);

}

}

//猴子类

class Monkey{

private String monkeyId;

private float height;

public Monkey(String mokeyId,float height){

this.monkeyId=mokeyId;

this.height=height;

}

//冒泡排序

public void bubbleSort(Monkey []monkeys){

float tempHeight=0f;

String tempId="";

//外层循环length-1

for(int i=0;i<monkeys.length-1;i++){

//内层循环length-1-i

for(int j=0;j<monkeys.length-1-i;j++){

//判断

if(monkeys[j].height>monkeys[j+1].height){

//身高交换

tempHeight=monkeys[j].height;

monkeys[j].height=monkeys[j+1].height;

monkeys[j+1].height=tempHeight;

//编号交换

tempId=monkeys[j].monkeyId;

monkeys[j].monkeyId=monkeys[j+1].monkeyId;

monkeys[j+1].monkeyId=tempId;

}

}

}

}

//显示队列

public void show(Monkey []monkeys){

for(int i=0;i<monkeys.length;i++){

System.out.println("猴子编号:"+monkeys[i].monkeyId+"身高:"+monkeys[i].height);

}

}

}

-------------------------------------------------------------------------------

第二关：[Work18.java]

老猴没办法，只有再次开始排。这时他用选择法，希望能快一点(你帮他设计这个选择排序程序)可大圣还对速度不满意，说道：你难道不能排的更快些吗？

/\*\*

\* 猴子过三关(选择排序)

\*/

import java.util.Scanner;

public class Work18 {

public static void main(String[] args) {

//定义可以装五只猴子的对象数组

int size=5;

Monkey []monkeys=new Monkey[size];

Scanner sr=new Scanner(System.in);

//初始化各个猴子

for(int i=0;i<monkeys.length;i++){

System.out.println("请输入第"+(i+1)+"只猴子的高度：");

float height=sr.nextFloat();

monkeys[i]=new Monkey((i+1)+"",height);

}

//让老猴排序

Monkey oldMonkey=new Monkey("1000", 1.2f);

//排序

oldMonkey.selectSort(monkeys);

//验证输出

oldMonkey.show(monkeys);

}

}

//猴子类

class Monkey{

private String monkeyId;

private float height;

public Monkey(String mokeyId,float height){

this.monkeyId=mokeyId;

this.height=height;

}

//选择排序(用外循环做为下标交换)

public void selectSort(Monkey []monkeys){

float tempHeight=0f;

String tempId="";

for(int i=0;i<monkeys.length;i++){

//认为下标为i的猴子身高是最低的

float minHeight=monkeys[i].height;

int minIndex=i;

//和后面猴子比较

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

if(minHeight>monkeys[j].height){

//修正最低值

minHeight=monkeys[j].height;

minIndex=j;

}

if(minIndex!=i){

//交换

tempHeight=monkeys[minIndex].height;

monkeys[minIndex].height=monkeys[i].height;

monkeys[i].height=tempHeight;

tempId=monkeys[minIndex].monkeyId;

monkeys[minIndex].monkeyId=monkeys[i].monkeyId;

monkeys[i].monkeyId=tempId;

}

}

}

}

//显示队列

public void show(Monkey []monkeys){

for(int i=0;i<monkeys.length;i++){

System.out.println("猴子编号:"+monkeys[i].monkeyId+"身高:"+monkeys[i].height);

}

}

}

-------------------------------------------------------------------------------

第三关[Work19.java]

老猴只好最后赌一把，采用还不是很熟悉的插入法排序。聪明的你，能够帮助老猴解决这个问题吗？

/\*\*

\* 猴子过三关(插入排序)

\*/

import java.util.Scanner;

public class Work19 {

public static void main(String[] args) {

//定义可以装五只猴子的对象数组

int size=5;

Monkey []monkeys=new Monkey[size];

Scanner sr=new Scanner(System.in);

//初始化各个猴子

for(int i=0;i<monkeys.length;i++){

System.out.println("请输入第"+(i+1)+"只猴子的高度：");

float height=sr.nextFloat();

monkeys[i]=new Monkey((i+1)+"",height);

}

//让老猴排序

Monkey oldMonkey=new Monkey("1000", 1.2f);

//排序

oldMonkey.insertSort(monkeys);

//验证输出

oldMonkey.show(monkeys);

}

}

//猴子类

class Monkey{

private String monkeyId;

private float height;

public Monkey(String mokeyId,float height){

this.monkeyId=mokeyId;

this.height=height;

}

//插入排序

public void insertSort(Monkey []monkeys){

//开始排序

for(int i=1;i<monkeys.length;i++){

//先记录该猴子的身高

float insertHeight=monkeys[i].height;

String insertId=monkeys[i].monkeyId;

int insertIndex=i-1;

while(insertIndex>=0&&monkeys[insertIndex].height>insertHeight){

monkeys[insertIndex+1].height=monkeys[insertIndex].height;

monkeys[insertIndex+1].monkeyId=monkeys[insertIndex].monkeyId;

insertIndex--;

}

//插入

monkeys[insertIndex+1].height=insertHeight;

monkeys[insertIndex+1].monkeyId=insertId;

}

}

//显示队列

public void show(Monkey []monkeys){

for(int i=0;i<monkeys.length;i++){

System.out.println("猴子编号:"+monkeys[i].monkeyId+"身高:"+monkeys[i].height);

}

}

}

-------------------------------------------------------------------------------

# 赛迪网Java程序员笔试题

第一大题：选择题(共20个小题，要求和分值按每题后面的提示，共60分)

1、下面哪些是java语言中的关键字？ B

A、sizeof

B、abstract

C、NULL

D、Native

2、第一个java语言编译器是在()年推出。 D

A、1991

B、1992

C、2001

D、1996

3、以下()是JAVA的保留字 C

A、Java

B、Hello

C、class

D、Class

4、下面程序运行之后，变量x的值是() B

//swap方法的声明

public static void swap(int a,int b){

int t=a;

a=b;

b=t;

}

//main方法

public static void main(String args[]){ A

int x=2;

int y=3;

swap(x,y);//简单数据类型是不会交换

}

A、2 B、3 C、4 D、6

5、下面变量var的作用域范围是() C

//....

int x;

swhitch(x){

case 0:

{

int var;

//process

}

break;

case 1:

{

int var1;

//process

}

break;

}

A、1和16行之间 B、4和8行之间 C、6和8行之间 D、6和14行之间

6、以下的类(接口)定义中正确的是() B

A、

public class a{

private int x;

public getX(){

return x;

}

}

B、

public abstract class a{

private int x;

public abstract int getX();

public int aMethod(){

return 0;

}

}

C、

public class a{

private int x;

public abstract int getX();

}

D、

public interface interfaceA{

private int x;

public int getX(){

return x;

}

}

7、已知A类被打包在packageA,B类被打包在packageB,且B类被声明为public，且有一个成员变量x被声明为protected控制方式。C类也位于packageA包，且继承了B类。则以下说法正确的是() C

A、A类的实例不能访问到B类的实例

B、A类的实例能够访问到B类一个实例的x成员

C、C类的实例可以访问到B类一个实例的x成员

D、C类的实例不能访问到B类的实例

8、以下()类不属于字符流类 C

A、Reader B、FileReader C、BufferedInputStream D、StringReader

9、以下()类属于字节流类 D

A、FileWriter

B、PushbackReader

C、FilterReader

D、FileInputStream

10、编译并运行下面的java代码段：

char c='a';

switch(c){

case 'a':

System.out.println("a");

default:

System.out.println("default");

}

输出结果是()。 B

A、代码无法编译，因为switch语句没有一个合法的表达式

B、a Default

C、a

D、default

11、分析下面的java程序

public class yy{

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

try{

throw new Exception();

}catch(Exception e){

System.out.println("Caught in main()");

}

System.out.println("nothing");

}

}

输出结果为() A

A、Caught in main() Nothing

B、Caught in main()

C、nothing

D、没有任何输出

12、在Java中，关于final关键字的说法正确的是()。A、C

A、如果修饰变量，则一旦赋了值，就等同一个常量

B、如果修饰类，则该类只能被一个子类继承

C、如果修饰方法，则该方法不能在子类中被覆盖

D、如果修饰方法，则该方法所在的类不能被继承

13、在java中，要想使只有定义该类所在的包内的类可以访问该类，应该用()关键字。A

A、不需要任何关键字

B、private

C、final

D、protected

14、在java中，下面关于包的陈述中正确的是() A、D

A、包的声明必须是源文件的第一句代码

B、包的声明必须紧跟在import语句的后面

C、只有公共类才能放在包中

D、可以将多个源文件中的类放在同一个包中

15、public static void main方法的参数描述是正确的()。A、B

A、String args[]

B、String[] args

C、Strings args[]

D、String args

16、在java中，关于CLASSPATH环境变量的说法不正确的是() A

A、CLASSPATH一旦设置之后不可修改，但可以将目录到该环境变量中

B、编译器用它来搜索各自的类文件

C、CLASSPATH是一个目录列表

D、解释器用它来搜索各自的类文件

17、编译并运行下面的java程序

class A{

int var1=1;

int var2;

public static void main(String[] args){

int var3=3;

A a=new A();

System.out.println(a.var1+a.var2+var3);

}

}

将产生()结果 B

A、0

B、4

C、3

D、代码无法编译，因为var2根本没有被初始化

18、在单一文件中import、class和package的正确出现顺序是() A

A、package,import,class

B、class,import,package

C、import,package,class

D、package,class,import

19、编译，运行下列代码后的结果是() D //类的成员变量可以不赋初值，局部变量要赋初值

public class Test{

public static void main(String args[]){

int age;

age=age+1;

System.out.println("The age is"+age);

}

}

A、编译，运行后没有输出

B、编译，运行后输出：The age is1

C、能通过编译，但运行时产生错误

D、不能通过编译

20、下列哪些表达式返回true() A、B

A、"john"=="john"

B、"john".equals("john")

C、"john"="john"

D、"john".equals(new Button("john"))

第二大题：

1、用循环控制语句打印输出：1+3+5+...+99=?结果

答题：

int sum=0;

for(int i=1;i<100;i+=2){

sum+=i;

System.out.println("i="+i+"sum="+sum);

}

结果：2500

2、请编写一个程序，实现从控制台读入字符，接收到"e"或者"E"退出程序

答题：

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Test {

public static void main(String[] args) {

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

while(true){

System.out.println("请输入任意字符(输入e或E退出):");

try {

String chars=br.readLine();

if(chars.equals("e")||chars.equals("E")){

System.out.println("输入的字符是e或是E，退出");

System.exit(0);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

3、编写两个方法分别计算合数n!.一个使用递归方法实现，另一个不用递归方法。

答题：(非递归方法)

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class test {

public static void main(String[] args) {

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

int num=1;

while(true){

System.out.println("请输入整数:");

try {

int note=Integer.parseInt(br.readLine());

if((note%1==0&&note%2==0)||(note%1==0&&note%3==0)||(note%1==0&&note%5==0)||(note%1==0&&note%7==0)){

System.out.println("你输入整数"+note+"是合数!");

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

num=num\*i;

}

System.out.println(note+"的阶乘结果是："+num);

num=1;

}else{

System.out.println(note+"不是合数");

}

if(note==-1){

System.out.println("退出");

break;

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

------------------------------------------------------------------------------

答题：(递归方法)

import java.io.\*;

public class Test {

public static void main(String[] args) {

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

while(true){

System.out.println("请输入一个整数:");

try {

int note=Integer.parseInt(br.readLine());

if((note%1==0&&note%2==0)||(note%1==0&&note%3==0)||(note%1==0&&note%5==0)||(note%1==0&&note%7==0)){

System.out.println("你输入整数"+note+"是合数!");

}

Test t = new Test();

int n = t.calc(note);

System.out.println(note+"阶乘结果是："+n);

if(note==0){

System.out.println("输入0退出");

break;

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

public int calc(int n) {

if(n==0){

return 1;

}

return n\*calc(n-1);

}

}

4、编写一个程序，在控制台接收数据，程序会让用户输入名字，并检查用户的输入：不允许输入空名字：一旦用户完成输入，程序会向该用户发出问题：“Hello，你输入的名字”。(提示：如果希望在控制台中与用户交互，一种可能的做法就是将System.in包装成)BufferedReader要做到这一点，必须使用InputStreamReader类将System.in这个InputStream对象转换成一个Reader即BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

答题：

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Test {

public static void main(String[] args) {

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

while(true){

System.out.println("请输入名字:");

try {

String name=br.readLine();

if(name.equals("")){

System.out.println("你输入的名字是空，退出!");

System.exit(0);

}else{

System.out.println("Hello,"+name+"!");

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

# 坦克大战案例源代码

[MyTank07.java]源代码

/\*\*

\* 功能：坦克游戏的2.01

\* 1、画出我的坦克

\* 2、让我的坦克动起来

\* 3、让我的坦克按下空格(space)键发射子弹

\* 4、让我的坦克可以连发子弹(最多连发5颗子弹)

\* 5、打到敌人坦克，敌人坦克就消失

\* 6、加入坦克被击中爆炸的效果

\* 7、让敌人的坦克可以自由随机移动

\* 8、控制坦克在指定的MyPanel面板中移动，不可越界

\* 9、让敌人的坦克发射子弹

\* 10、当我的坦克被敌人子弹击中，我的坦克爆炸

\* 11、防止敌人坦克重叠运动

\* 12、可以分关--做一个开始的Panel，它是空的主要是提示关卡

\* 13、游戏可以暂停、继续--暂停时将子弹、坦克速度设为0，坦克方向不变

\* 14、可以记录玩家的成绩

\* 15、java如何操作声音文件

\*/

package com.haiding.tank\_7;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.File;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.util.\*;

import javax.sound.sampled.\*;

public class MyTank07 extends JFrame implements ActionListener{

//定义组件

MyPanel mp=null;

//定义一个开始面板

MyStartPanel msp=null;

//做出菜单

JMenuBar jmb=null;

//开始游戏

JMenu jm1=null;

JMenuItem jmi1=null;

//退出游戏

JMenuItem jmi2=null;

//存盘退出

JMenuItem jmi3=null;

//接上局

JMenuItem jmi4=null;

public static void main(String[] args) {

MyTank07 mt=new MyTank07();

}

//构造函数

public MyTank07(){

//创建菜单及菜单选项

jmb=new JMenuBar();

jm1=new JMenu("游戏(G)");

//设置快捷方式

jm1.setMnemonic('G');

jmi1=new JMenuItem("开始新游戏(N)");

jmi1.setMnemonic('N');

//对jmi1相应

jmi1.addActionListener(this);

jmi1.setActionCommand("newgame");

jmi2=new JMenuItem("退出游戏(E)");

jmi2.setMnemonic('E');

jmi2.addActionListener(this);

jmi2.setActionCommand("exit");

jmi3=new JMenuItem("存盘退出(S)");

jmi3.setMnemonic('S');

jmi3.addActionListener(this);

jmi3.setActionCommand("save");

jmi4=new JMenuItem("继续游戏(C)");

jmi4.setMnemonic('C');

jmi4.addActionListener(this);

jmi4.setActionCommand("congame");

jm1.add(jmi1);

jm1.add(jmi2);

jm1.add(jmi3);

jm1.add(jmi4);

jmb.add(jm1);

this.setJMenuBar(jmb);

//构建组件

msp=new MyStartPanel();

Thread t=new Thread(msp);

t.start();

this.add(msp);

//设置JFrame窗体

this.setTitle("坦克大战");//JFrame标题

this.setSize(600, 500);//JFrame窗体大小

this.setLocationRelativeTo(null);//在屏幕中心显示

this.setVisible(true);//显示

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//退出并关闭JFrame

}

public void actionPerformed(ActionEvent e) {

//对用户不同的点击，做出不同的处理

if(e.getActionCommand().equals("newgame")){

//创建战场面板

mp=new MyPanel("newgame");

//启动mp线程

Thread ta=new Thread(mp);

ta.start();

//先删除旧的开始面板

this.remove(msp);

//加入组件

this.add(mp);

//监听

this.addKeyListener(mp);

//显示，刷新JFrame

this.setVisible(true);

}else if(e.getActionCommand().equals("exit")){

//用户点击了退出菜单

//保存用户游戏记录数据

Recorder.keepRecording();

System.exit(0);

}else if(e.getActionCommand().equals("save")){

//用户点击保存菜单

//保存击毁敌人的数量和敌人的坐标

Recorder.setEts(mp.ets);

Recorder.keepRecAndEnemyTank();

System.exit(0);

}else if(e.getActionCommand().equals("congame")){

//继续上次游戏

//创建战场面板

mp=new MyPanel("congame");

//启动mp线程

Thread ta=new Thread(mp);

ta.start();

//先删除旧的开始面板

this.remove(msp);

//加入组件

this.add(mp);

//监听

this.addKeyListener(mp);

//显示，刷新JFrame

this.setVisible(true);

}

}

}

//开始面板--起提示作用

class MyStartPanel extends JPanel implements Runnable{

int times=0;

public void paint(Graphics g){

super.paint(g);

g.setColor(Color.black);

g.fillRect(0, 0, 400, 300);

//提示信息

if(times%2==0){

g.setColor(Color.yellow);

//开关信息的字体

Font myFont=new Font("华文新魏", Font.BOLD, 30);

g.setFont(myFont);

g.drawString("Stage: 1", 150, 150);

}

}

public void run() {

while(true){

//休眠

try {

Thread.sleep(500);

} catch (Exception e) {

e.printStackTrace();

}

times++;

//重画

this.repaint();

}

}

}

//我的面板Panel

class MyPanel extends JPanel implements KeyListener,Runnable{//MyPanel继承Jpanel面板，创建KeyListener键盘接口和Runnable线程接口

//定义一个我的坦克

Hero hero=null;

//定义声音播放

AePlayWave apw=null;

//定义敌人的坦克组

Vector<EnemyTank> ets=new Vector<EnemyTank>();

Vector<Node> nodes=new Vector<Node>();

//定义炸弹集合

Vector<Bomb> bombs=new Vector<Bomb>();

int enSize=3;//初始敌人坦克数量

//定义爆炸图片

Image image1=null;

Image image2=null;

Image image3=null;

//构造函数

public MyPanel(String flag){

hero=new Hero(200, 270);//我的坦克初始位置

//恢复游戏记录

Recorder.getRecoring();

if(flag.equals("newgame")){

//播放开战声音

apw=new AePlayWave("e:\\tankgame\\tankmusic.wav");

apw.start();

//初始化敌人的坦克

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

//创建一辆敌人的坦克对象

EnemyTank et=new EnemyTank((i+1)\*50, 0);

//初始敌人坦克颜色

et.setColor(2);

//初始敌人坦克的方向

et.setDirect(1);

//将MyPanel的敌人向量交给该敌人坦克

et.setEts(ets);

//启动敌人坦克进程

Thread t=new Thread(et);

t.start();

//给敌人坦克添加一颗子弹

Shot s=new Shot(et.x+10, et.y+30, 1);

//把子弹加入给敌人

et.ss.add(s);

Thread t2=new Thread(s);

t2.start();

//加入敌人坦克

ets.add(et);

}

}else if(flag.equals("congame")){

nodes=new Recorder().getNodesAndEnNums();

//初始化敌人的坦克

for(int i=0;i<nodes.size();i++){

Node node=nodes.get(i);

//创建一辆敌人的坦克对象

EnemyTank et=new EnemyTank(node.x, node.y);

//初始敌人坦克颜色

et.setColor(2);

//初始敌人坦克的方向

et.setDirect(node.direct);

//将MyPanel的敌人向量交给该敌人坦克

et.setEts(ets);

//启动敌人坦克进程

Thread t=new Thread(et);

t.start();

//给敌人坦克添加一颗子弹

Shot s=new Shot(et.x+10, et.y+30, 1);

//把子弹加入给敌人

et.ss.add(s);

Thread t2=new Thread(s);

t2.start();

//加入敌人坦克

ets.add(et);

}

}

//初始化爆炸图片

try {

image1=ImageIO.read(new File("bomb1.gif"));

image2=ImageIO.read(new File("bomb2.gif"));

image3=ImageIO.read(new File("bomb3.gif"));

} catch (Exception e) {

e.printStackTrace();

}

}

//画出提示信息

public void showInfo(Graphics g){

//画出提示信息坦克(该坦克不参与战斗，只是用于显示提示信息)

this.drawTank(80, 330, g, 1, 0);

g.setColor(Color.BLACK);

g.setFont(new Font("宋体",Font.BOLD,20));

g.drawString(Recorder.getEnNum()+"", 110, 350);//显示敌人坦克数量

this.drawTank(140, 330, g, 0, 0);

g.setColor(Color.BLACK);

g.setFont(new Font("宋体",Font.BOLD,20));

g.drawString(Recorder.getMyLife()+"", 170, 350);//显示我的坦克数量

//画出玩家的总成绩

g.setColor(Color.black);

g.setFont(new Font("宋体",Font.BOLD,20));

g.drawString("您的总成绩", 420, 30);

this.drawTank(420, 60, g, 1, 0);

g.setColor(Color.black);

g.setFont(new Font("宋体",Font.BOLD,20));

g.drawString(Recorder.getAllEnNum()+"", 460, 80);

}

//重写paint函数

public void paint(Graphics g){

super.paint(g);//调用父类paint方法

//画出提示信息

this.showInfo(g);

//设置Panel底色

g.setColor(Color.black);

g.fillRect(0, 0, 400, 300);//fillRect(0,0,X?,Y?)中X?/Y?为活动区域

//画出自己的坦克

if(hero.isLive==true){

this.drawTank(hero.getX(), hero.getY(), g, 0, this.hero.direct);

}

//从ss向量中取出每一颗子弹并画出

for(int i=0;i<this.hero.ss.size();i++){

Shot myShot=hero.ss.get(i);

//画出一颗子弹，判断子弹是否为空

if(myShot!=null&&myShot.isLive==true){

g.draw3DRect(myShot.x, myShot.y, 1, 1, false);

}

//判断子弹是否死亡

if(myShot.isLive==false){

//从向量ss中删除该子弹

hero.ss.remove(myShot);

}

}

//画出炸弹

for(int i=0;i<bombs.size();i++){

//取出炸弹

Bomb b=bombs.get(i);

if(b.life>6){

g.drawImage(image1, b.x, b.y, 30, 30, this);

}else if(b.life>3){

g.drawImage(image2, b.x, b.y, 30, 30, this);

}else {

g.drawImage(image3, b.x, b.y, 30, 30, this);

}

//让b的生命值减少

b.lifeDown();

//如果炸弹生命值为0，就把该炸弹从bombs向量中去掉

if(b.life==0){

bombs.remove(b);

}

}

//画出敌人的坦克

for(int i=0;i<ets.size();i++){

EnemyTank et=ets.get(i);

if(et.isLive){

this.drawTank(et.getX(), et.getY() , g, 1, et.getDirect());

//再画出敌人坦克的子弹

for(int j=0;j<et.ss.size();j++){

//取出子弹

Shot enemyShot=et.ss.get(j);

if(enemyShot.isLive){

g.draw3DRect(enemyShot.x, enemyShot.y, 1, 1, false);

}else{

//如果敌人的坦克死亡了就从向量Vector中去掉

et.ss.remove(enemyShot);

}

}

}

}

}

//判断我的子弹是否击中敌人的坦克

public void hitEnemyTank(){

//遍历Vector集合类

for(int i=0;i<hero.ss.size();i++){

//先取子弹

Shot myShot=hero.ss.get(i);

//判断子弹是否有效

if(myShot.isLive){

//取出每一个敌人坦克与子弹进行判断

for(int j=0;j<ets.size();j++){

//取出坦克

EnemyTank et=ets.get(j);

//判断敌人坦克是否还活着

if(et.isLive){

if(this.hitTank(myShot, et)){

apw=new AePlayWave("e:\\tankgame\\tank\_explosion.wav");

apw.start();

//调用reduceEnNum()减少敌人坦克统计数

Recorder.reduceEnNum();

//调用addEnNumRec()增加消灭敌人坦克统计数

Recorder.addEnNumRec();

}

}

}

}

}

}

//判断敌人的子弹是否击中我的坦克

public void hitMe(){

//取出每一个敌人的坦克

for(int i=0;i<this.ets.size();i++){

//取出敌人的坦克

EnemyTank et=ets.get(i);

//取出每一颗敌人的子弹

for(int j=0;j<et.ss.size();j++){

//取出子弹

Shot enemyShot=et.ss.get(j);

if(hero.isLive){

if(this.hitTank(enemyShot, hero)){

}

}

}

}

}

//写一个函数专门判断子弹是否击中敌人坦克

public boolean hitTank(Shot s,Tank et){

boolean b=false;

//判断该敌人坦克的方向

switch(et.direct){

case 0://敌人坦克向上或向下

case 1:

if(s.x>et.x&&s.x<et.x+20&&s.y>et.y&&s.y<et.y+29){

//击中方向为上或下的敌人坦克

//子弹死亡

s.isLive=false;

//敌人坦克死亡

et.isLive=false;

b=true;

//创建一颗炸弹

Bomb bomb=new Bomb(et.x, et.y);

//放入Vector向量中管理

bombs.add(bomb);

}

break;

case 2://敌人坦克向左或向右

case 3:

if(s.x>et.x&&s.x<et.x+29&&s.y>et.y&&s.y<et.y+20){

//击中方向为左或右的敌人坦克

//子弹死亡

s.isLive=false;

//敌人坦克死亡

et.isLive=false;

b=true;

//创建一颗炸弹

Bomb bomb=new Bomb(et.x, et.y);

//放入Vector向量中管理

bombs.add(bomb);

}

break;

}

return b;

}

//画出坦克的函数

public void drawTank(int x,int y,Graphics g,int type,int direct){

//判断是什么类型的坦克

switch(type){

case 0:

g.setColor(Color.cyan);//我的坦克颜色

break;

case 1:

g.setColor(Color.yellow);//敌人坦克颜色

break;

case 2:

g.setColor(Color.red);

break;

}

//判断坦克的方向

switch(direct){

//向上走的坦克

case 0:

//画出我的坦克(到时再封装成一个函数)

//1、画出左边的矩形

g.fill3DRect(x, y, 5, 30, false);

//2、画出右边的矩形

g.fill3DRect(x+15, y, 5, 30, false);

//3、画出中间矩形

g.fill3DRect(x+5, y+5, 10, 20, false);

//4、画出中间圆形

g.fillOval(x+5, y+10, 10, 10);

//5、画出线(炮筒)

g.drawLine(x+10, y+15, x+10, y);

break;

//向下走的坦克

case 1:

g.fill3DRect(x, y, 5, 30, false);

g.fill3DRect(x+15, y, 5, 30, false);

g.fill3DRect(x+5, y+5, 10, 20, false);

g.fillOval(x+5, y+10, 10, 10);

g.drawLine(x+10, y+15, x+10, y+29);

break;

//向左走的坦克

case 2:

g.fill3DRect(x, y, 30, 5, false);

g.fill3DRect(x, y+15, 30, 5, false);

g.fill3DRect(x+5, y+5, 20, 10, false);

g.fillOval(x+10, y+5, 10, 10);

g.drawLine(x+15, y+10, x, y+10);

break;

//向右走的坦克

case 3:

g.fill3DRect(x, y, 30, 5, false);

g.fill3DRect(x, y+15, 30, 5, false);

g.fill3DRect(x+5, y+5, 20, 10, false);

g.fillOval(x+10, y+5, 10, 10);

g.drawLine(x+15, y+10, x+29, y+10);

break;

}

}

public void keyPressed(KeyEvent e) {//按下键事件a向左s向下d向右w向上

if(e.getKeyCode()==KeyEvent.VK\_W||e.getKeyCode()==KeyEvent.VK\_UP){

//向上

this.hero.setDirect(0);

this.hero.moveUp();

}else if(e.getKeyCode()==KeyEvent.VK\_S||e.getKeyCode()==KeyEvent.VK\_DOWN){

//设置我的坦克的方向,向下

this.hero.setDirect(1);

this.hero.moveDown();

}else if(e.getKeyCode()==KeyEvent.VK\_A||e.getKeyCode()==KeyEvent.VK\_LEFT){

//向左

this.hero.setDirect(2);

this.hero.moveLeft();

}else if(e.getKeyCode()==KeyEvent.VK\_D||e.getKeyCode()==KeyEvent.VK\_RIGHT){

//向右

this.hero.setDirect(3);

this.hero.moveRight();

}

//判断玩家是否按下空格键,不可接上面else if。不然不能同时按方向键和空格(space)键

if(e.getKeyCode()==KeyEvent.VK\_SPACE){

apw=new AePlayWave("e:\\tankgame\\tank\_Shelling\_sound.wav");

apw.start();

//控制子弹连发

if(this.hero.ss.size()<=4){

//按下空格后开火

this.hero.shotEnemy();

}

}

//调用repaint()函数，来重绘界面

this.repaint();

}

public void keyReleased(KeyEvent e) {//弹起键事件

}

public void keyTyped(KeyEvent e) {//按键输出值

}

//重写run函数

public void run() {

while(true){

try {

Thread.sleep(100);//休息100毫秒后重绘MyPanel面板

} catch (Exception e) {

e.printStackTrace();

}

this.hitEnemyTank();

//判断敌人的子弹是否击中我的坦克

this.hitMe();

//重绘MyPanel面板

this.repaint();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[Members.java]成员类

package com.haiding.tank\_7;

import java.util.Vector;

import java.io.\*;

import javax.sound.sampled.\*;

//播放声音的类

class AePlayWave extends Thread {

private String filename;

public AePlayWave(String wavfile) {

filename = wavfile;

}

public void run() {

File soundFile = new File(filename);

AudioInputStream audioInputStream = null;

try {

audioInputStream = AudioSystem.getAudioInputStream(soundFile);

} catch (Exception e1) {

e1.printStackTrace();

return;

}

AudioFormat format = audioInputStream.getFormat();

SourceDataLine auline = null;

DataLine.Info info = new DataLine.Info(SourceDataLine.class, format);

try {

auline = (SourceDataLine) AudioSystem.getLine(info);

auline.open(format);

} catch (Exception e) {

e.printStackTrace();

return;

}

auline.start();

int nBytesRead = 0;

//这是缓冲

byte[] abData = new byte[2048];

try {

while (nBytesRead != -1) {

nBytesRead = audioInputStream.read(abData, 0, abData.length);

if (nBytesRead >= 0)

auline.write(abData, 0, nBytesRead);

}

} catch (IOException e) {

e.printStackTrace();

return;

} finally {

auline.drain();

auline.close();

}

}

}

//记录恢复点

class Node{

int x,y,direct;

public Node(int x,int y,int direct){

this.x=x;

this.y=y;

this.direct=direct;

}

}

//记录类，同时也可以保存玩家的设置

class Recorder{

//记录每关有多少敌人

private static int enNum=20;

//设置我有多少可用的人

private static int myLife=3;

//记录总共消灭了多少敌人的坦克

private static int allEnNum=0;

//从文件中恢复记录点

private static Vector<Node> nodes=new Vector<Node>();

private static FileWriter fw=null;

private static BufferedWriter bw=null;

private static FileReader fr=null;

private static BufferedReader br=null;

private static Vector<EnemyTank> ets=new Vector<EnemyTank>();

//完成读取任务点任务

public static Vector<Node> getNodesAndEnNums(){

try {

fr=new FileReader("e:\\tankgame\\tanksave.txt");

br=new BufferedReader(fr);

String n="";

//先读一行

n=br.readLine();

allEnNum=Integer.parseInt(n);

while((n=br.readLine())!=null){

String []Recovery=n.split(" ");//split方法可以按一行字符中有多少个空间来返回数组

Node node=new Node(Integer.parseInt(Recovery[0]),Integer.parseInt(Recovery[1]),Integer.parseInt(Recovery[2]));

nodes.add(node);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

br.close();

fr.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

return nodes;

}

//保存击毁敌人的数量和敌人坦克坐标、方向

public static void keepRecAndEnemyTank(){

try {

//创建

fw=new FileWriter("e:\\tankgame\\tanksave.txt");

bw=new BufferedWriter(fw);

bw.write(allEnNum+"\r\n");

//保存当前还活着的敌人坦克坐标、方向

for(int i=0;i<ets.size();i++){

//取出第一个坦克

EnemyTank et=ets.get(i);

if(et.isLive){

//活的保存

String recode=et.x+" "+et.y+" "+et.direct;//得到坐标x,y和方向direct

//写入到文件

bw.write(recode+"\r\n");

}

}

} catch (Exception e) {

e.printStackTrace();

}finally{

//关闭流,先开后关，后开先关

try {

bw.close();

fw.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

//从文件中读取记录

public static void getRecoring(){

try {

fr=new FileReader("e:\\tankgame\\tanksave.txt");

br=new BufferedReader(fr);

String n=br.readLine();

allEnNum=Integer.parseInt(n);

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

br.close();

fr.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

//把玩家击毁敌人坦克数量保存到文件中

public static void keepRecording(){

try {

//创建

fw=new FileWriter("e:\\tankgame\\tanksave.txt");

bw=new BufferedWriter(fw);

bw.write(allEnNum+"\r\n");

} catch (Exception e) {

e.printStackTrace();

}finally{

//关闭流,先开后关，后开先关

try {

bw.close();

fw.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

public static int getAllEnNum() {

return allEnNum;

}

public static void setAllEnNum(int allEnNum) {

Recorder.allEnNum = allEnNum;

}

public static int getEnNum() {

return enNum;

}

public static void setEnNum(int enNum) {

Recorder.enNum = enNum;

}

public static int getMyLife() {

return myLife;

}

public static void setMyLife(int myLife) {

Recorder.myLife = myLife;

}

public static Vector<EnemyTank> getEts() {

return ets;

}

public static void setEts(Vector<EnemyTank> ets) {

Recorder.ets = ets;

}

//敌人坦克死亡就减少坦克数

public static void reduceEnNum(){

enNum--;

}

//当消灭敌人的时候

public static void addEnNumRec(){

allEnNum++;

}

}

//炸弹类

class Bomb{

//定义炸弹的坐标

int x,y;

int life=9;//炸弹的生命

boolean isLive=true;

public Bomb(int x,int y){

this.x=x;

this.y=y;

}

//减少炸弹生命值

public void lifeDown(){

if(life>0){

life--;

}else{

this.isLive=false;

}

}

}

//子弹类

class Shot implements Runnable{

int x,y,direct,speed=3;//初始子弹x,y坐标,direct子弹方向,speed子弹速度

//子弹是否还活着

boolean isLive=true;//默认为活着

public Shot(int x,int y,int direct){

this.x=x;

this.y=y;

this.direct=direct;

}

public void run(){

while(true){

try {

Thread.sleep(50);//让子弹休息50毫秒，防止快速消费内存

} catch (Exception e) {

e.printStackTrace();

}

switch(direct){

case 0://向上

y-=speed;

break;

case 1://向下

y+=speed;

break;

case 2://向 左

x-=speed;

break;

case 3://向右

x+=speed;

break;

}

//子弹何时死亡

//判断该子弹是否碰到MyPanel面板的边缘

if(x<0||x>400||y<0||y>300){

this.isLive=false;

break;

}

}

}

}

//定义坦克类

class Tank{

//表示坦克的X横坐标Y纵坐标

int x=0,y=0;

//direct坦克方向,0向上、1向下、2向左、3向右

int direct=0;

//坦克的速度

int speed=1;

//坦克颜色

int color;

boolean isLive=true;

public Tank(int x,int y){

this.x=x;

this.y=y;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

public int getDirect() {

return direct;

}

public void setDirect(int direct) {

this.direct = direct;

}

public int getSpeed() {

return speed;

}

public void setSpeed(int speed) {

this.speed = speed;

}

public int getColor() {

return color;

}

public void setColor(int color) {

this.color = color;

}

}

//敌人坦克,把敌人坦克做成线程类

class EnemyTank extends Tank implements Runnable{

int move=30;//敌人坦克移动步长(区域)

int times=0;

//定义一个向量，可以访问到MyPanel上所有敌人坦克

Vector<EnemyTank> ets=new Vector<EnemyTank>();

//定义一个向量，可以存放敌人的子弹

Vector<Shot> ss=new Vector<Shot>();

//敌人添加子弹，应在创建坦克和敌人的坦克子弹死亡后

public EnemyTank(int x,int y){

super(x,y);

}

//得到MyPanel的敌人坦克向量

public void setEts(Vector<EnemyTank> tank){

this.ets=tank;

}

//判断敌人坦克是否重叠

public boolean isTouchOtherEnemy(){

boolean b=false;

switch(this.direct){

case 0://敌人当前坦克向上

//取出所有敌人坦克

for(int i=0;i<ets.size();i++){

//取出第一个坦克

EnemyTank et=ets.get(i);

//如果不是当前的坦克

if(et!=this){

//如果敌人同伴的坦克向上或向下

if(et.direct==0||et.direct==1){

//判断敌人当前坦克左轮与同伴坦克(向上或向下)的位置比较

if(this.x>=et.x&&this.x<=et.x+20&&this.y>=et.y&&this.y<=et.y+30){

return true;

}

if(this.x+20>=et.x&&this.x+20<=et.x+20&&this.y>=et.y&&this.y<=et.y+30){

return true;

}

}

//如果敌人同伴的坦克向左或向右

if(et.direct==2||et.direct==3){

//判断敌人当前坦克右轮与同伴坦克(向左或向右)的位置比较

if(this.x>=et.x&&this.x<=et.x+30&&this.y>=et.y&&this.y<=et.y+20){

return true;

}

if(this.x+20>=et.x&&this.x+20<=et.x+30&&this.y>=et.y&&this.y<=et.y+20){

return true;

}

}

}

}

break;

case 1://敌人当前坦克向下

for(int i=0;i<ets.size();i++){

EnemyTank et=ets.get(i);

if(et!=this){

if(et.direct==0||et.direct==1){

if(this.x>=et.x&&this.x<=et.x+20&&this.y+30>=et.y&&this.y+30<=et.y+30){

return true;

}

if(this.x+20>=et.x&&this.x+20<=et.x+20&&this.y+30>=et.y&&this.y+30<=et.y+30){

return true;

}

}

if(et.direct==2||et.direct==3){

if(this.x>=et.x&&this.x<=et.x+30&&this.y+30>=et.y&&this.y+30<=et.y+20){

return true;

}

if(this.x+20>=et.x&&this.x+20<=et.x+30&&this.y+30>=et.y&&this.y+30<=et.y+20){

return true;

}

}

}

}

break;

case 2://敌人当前坦克向左

for(int i=0;i<ets.size();i++){

EnemyTank et=ets.get(i);

if(et!=this){

if(et.direct==0||et.direct==1){

if(this.x>=et.x&&this.x<=et.x+20&&this.y>=et.y&&this.y<=et.y+30){

return true;

}

if(this.x>=et.x&&this.x<=et.x+20&&this.y+20>=et.y&&this.y+20<=et.y+30){

return true;

}

}

if(et.direct==2||et.direct==3){

if(this.x>=et.x&&this.x<=et.x+30&&this.y>=et.y&&this.y<=et.y+20){

return true;

}

if(this.x>=et.x&&this.x<=et.x+30&&this.y+20>=et.y&&this.y+20<=et.y+20){

return true;

}

}

}

}

break;

case 3://敌人当前坦克向右

for(int i=0;i<ets.size();i++){

EnemyTank et=ets.get(i);

if(et!=this){

if(et.direct==0||et.direct==1){

if(this.x+30>=et.x&&this.x+30<=et.x+20&&this.y>=et.y&&this.y<=et.y+30){

return true;

}

if(this.x+30>=et.x&&this.x+30<=et.x+20&&this.y+20>=et.y&&this.y+20<=et.y+30){

return true;

}

}

if(et.direct==2||et.direct==3){

if(this.x+30>=et.x&&this.x+30<=et.x+30&&this.y>=et.y&&this.y<=et.y+20){

return true;

}

if(this.x+30>=et.x&&this.x+30<=et.x+30&&this.y+20>=et.y&&this.y+20<=et.y+20){

return true;

}

}

}

}

break;

}

return b;

}

//重写run()函数

public void run() {

while(true){

switch(this.direct){

case 0://坦克正在向上移动

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

if(y>0&&!this.isTouchOtherEnemy()){//判断敌人坦克向上走不会越界

y-=speed;

}

try {

Thread.sleep(50);

} catch (Exception e) {

e.printStackTrace();

}

}

break;

case 1://向下移动

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

if(y<400&&!this.isTouchOtherEnemy()){//判断敌人坦克向下走不会越界

y+=speed;

}

try {

Thread.sleep(50);

} catch (Exception e) {

e.printStackTrace();

}

}

break;

case 2://向左移动

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

if(x>0&&!this.isTouchOtherEnemy()){//判断敌人坦克向左走不会越界

x-=speed;

}

try {

Thread.sleep(50);

} catch (Exception e) {

e.printStackTrace();

}

}

break;

case 3://向右移动

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

if(x<300&&!this.isTouchOtherEnemy()){//判断敌人坦克向右走不会越界

x+=speed;

}

try {

Thread.sleep(50);

} catch (Exception e) {

e.printStackTrace();

}

}

break;

}

this.times++;

if(times%2==0){

if(isLive){

if(ss.size()<5){

Shot s=null;

//没有子弹,添加子弹

switch(direct){

case 0:

s=new Shot(x+10, y, 0);

ss.add(s);

break;

case 1:

s=new Shot(x+10, y+29, 1);

ss.add(s);

break;

case 2:

s=new Shot(x, y+10, 2);

ss.add(s);

break;

case 3:

s=new Shot(x+29, y+10, 3);

ss.add(s);

break;

}

//启动敌人子弹线程

Thread t=new Thread(s);

t.start();

}

}

}

//让坦克随机产生一个新的行驶方向

this.direct=(int)(Math.random()\*4);

//判断敌人坦克是否已死亡

if(this.isLive==false){

//让坦克死亡后退出线程

break;

}

}

}

}

//我的坦克

class Hero extends Tank{

//子弹

//Shot s=null;

Vector<Shot> ss=new Vector<Shot>();//Vector线程安全的集合类

Shot s=null;

public Hero(int x,int y){

super(x,y);

}

//坦克开火

public void shotEnemy(){

switch(this.direct){//判断坦克方向来对应子弹运行方向

case 0://向上

//创建一颗子弹

s=new Shot(x+10, y, 0);

//将子弹加入到向量集中

ss.add(s);

break;

case 1://向下

s=new Shot(x+10, y+29, 1);

ss.add(s);

break;

case 2://向左

s=new Shot(x, y+10, 2);

ss.add(s);

break;

case 3://向右

s=new Shot(x+29, y+10, 3);

ss.add(s);

break;

}

//启动子弹线程

Thread t=new Thread(s);

t.start();

}

//坦克向上移动

public void moveUp(){

y-=speed;

}

//坦克向右移动

public void moveRight(){

x+=speed;

}

//坦克向下移动

public void moveDown(){

y+=speed;

}

//坦克向左移动

public void moveLeft(){

x-=speed;

}

}

**数据库学习**

sql server数据库基本概念

使用文件保存数据存在几个缺点：

1、文件的安全性问题；

2、文件不利于查询和对数据的管理；

3、文件不利于存放海量数据

4、文件在程序中控制不方便。

数据库的定义(1)

严格地说，数据库是“按照数据结构来组织、存储和管理数据的仓库”。在经济管理的日常工作中，常常需要把某些相关的数据放进这样的“仓库”，并根据管理的需要进行相应的处理。例如，企业或事业单位的人事部门常常要把单位职工的基本情况(职工号、姓名、年龄、性别、籍贯、工资、简历等)存放在表中，这张表就可以看成一个数据库。有了这个“数据仓库”我们就可以根据需要随时查询某职工的基本情况，也可以查询工资在某个范围内的职工人数等等。这些工作如果都能在计算机上自动进行，那我们的人事管理就可以达到极高的水平。此外，在财务管理、仓库管理、生产管理中也需要建立众多的这种“数据库”，使其可以利用计算机实现财务、仓库、生产的自动化管理。

数据库的定义(2)

J.Martin组数据库下了一个比较完整的定义：数据库是存储在一起的相关数据的集合，这些数据是结构化的，无有害的或不必要的冗余，并为多种应用服务；数据的存储独立于使用它的程序；对数据库插入新数据，修改和检索原有的数据均能按一种公用的和可控制的方式进行。当某个系统中存在结构上完全分开的若干个数据库时，则该系统包含一个“数据库集合”。

数据库的基本结构

数据库的基本结构分三个层次，反映了观察数据库的三种不同角度。

1、物理数据层

它是数据库的最内层，是物理存贮设备上实际存储的数据的集合。这些数据的原始数据，是用户加工的对象，由内部模式描述的指令操作处理的位串、字符和字组成。

2、概念数据层

它是数据库的中间一层，是数据库的整体逻辑表示。指出了每个数据的逻辑定义及数据间的逻辑联系，是存贮记录的集合。它所涉及的是数据库所有对象的逻辑关系，而不是它们的物理情况，是数据库管理员概念下的数据库。

3、逻辑数据层

它是用户所看到和使用的数据库，表示了一个或一些特定用户使用的数据集合，即逻辑记录的集合。

数据库不同层次之间的联系是通过映射进行转换的。

数据库的基本特点

1、实现数据共享

数据共享包含所有用户可同时存取数据库中的数据，也包括用户可以用各种方式通过接口使用数据库，并提供数据共享。

2、减少数据的冗余度

同文件系统比，数据库实现了数据共享，从而避免了用户各自建立应用文件。减少了大量重复数据，减少了数据冗余，维护了数据的一致性

3、数据实现集中控制

文件管理方式中，数据处于一种分散的状态，不同的用户或同一用户在不同处理中其文件之间毫无关系。利用数据库可对数据进行集中控制和管理，并通过数据模型表示各种数据的组织以及数据间的联系。

4、数据一致性和可维护性，以确保数据的安全性和可靠性。

5、故障恢复

目前主流数据库

微软： Sql Server和Access

瑞典MySQL： AB公司MySql

IBM公司： DB2

美国Sybase公司： Sybase

IBM公司： Informix

美国Oracle公司： Oracle

数据库选择：

1、成本；2、功能；3、并发性要求；4、安全性；

为什么学习Sql Server？

简单易学

一般来讲，学习数据库很多人都是从微软的sql server数据库开始的，我们知道微软的产品以简单易用见长，事实也是如此，从sql server开始学习数据库是正确的选择，后面我们还要带领大学学习oracle数据库，如果一上手就学习oracle这种大型数据库，会很吃力。

sql server介绍(1)

SQL是英文Structured Query Language的缩写，意思为结构化查询语言。SQL语言的主要功能就是同各种数据库建立联系，进行沟通。按照ANSI(美国国家标准协会)的规定，SQL被作为关系型数据库管理系统的标准语言。SQL语句可以用来执行各种各样的操作，例如更新数据库中的数据，从数据库中提取数据等。目前，绝大多数流行的关系型数据库管理系统，如Oracle、Sybase、Microsoft SQL Server、Access等都采用了SQL语言标准。

sql server介绍(2)

SQL Server是一个关系数据库管理系统。它最初是由Microsoft Sybase和Ashton-Tate三家公司共同开发的，于1988年推出了第一个OS/2版本。在Windows NT推出后，Microsoft与Sybase在SQL Server的开发上就分道扬镳了，Microsoft将SQL Server移植到Windows NT系统上，专注于开发推广SQL Server的Windows NT版本。Sybase则较专注于SQL Server在UNIX操作系统上的应用。

SQL Server2000是Microsoft公司推出的SQL Server数据库管理系统，该版本继承了SQL

Server7.0版本的优点，同时又比它增加了许多更先进的功能。

sql server开发工具

sql server2000为我们提供了两种开发工具

1、企业管理器

企业管理器是微软提供的图形界面方式操作sql server2000的工具

2、查询分析器

查询分析器是微软提供的用命令行(sql语句)操作sql server2000的工具

企业管理器的使用(1)

数据库用户和管理员的概念

1、管理员sa的介绍

2、如何修改sa的密码

企业管理器的使用(2)

为了让大家能迅速掌握企业管理器的使用，我们使用管理器。

以一个梁山好汉管理为例

1、建库

2、建表

英雄表 hero

[这里涉及到对表的几个重要概念：a、字段；b、字段类型(简单介绍)；c、表行；d、表列；e、记录]

3、对表进行增加、删除、修改、查询的操作

这里可以针对真实班级的学生来做，并增加一定的互动，比如用点名机来抽学生操作，或是用点名机看看那个学生被删除或是添加、修改...这样可能学生更有兴趣。

查询分析器的使用(1)

企业管理器给用户提供了一个很方便的图形界面管理工具，大家用起来感觉直观方便，可是它也有缺点，当一个表的记录非常大的时候，对表的各种操作，都显得不方便了，比如：

1、要求从1000行记录中查询是否存在名为“孙小明”的人

2、要求把1000行记录中工资低于100的人，工资增加10%

3、要求把年龄大于30的人从数据表中删除

这时，你会发现使用企业管理器是很不方便的

所以微软为我们提供另外一个操作数据库的工具，查询分析器。查询分析器可以非常方便的完成上述任务。

查询分析器的使用(2)--sql语句

使用查询分析器，我们就必需要了解sql语句。

什么是sql语句呢？

人与人交流是使用语言进行交流。我们与sql数据库交流就要使用对应的计算机语言进行交流，所以访问sql数据库要使用sql语句进行指令的发送。

查询分析器的使用(3)--sql语句

SQL全称是“结构化查询语言(Structured Query Language)”

SQL(Structured Query Language)是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。构化查询语言(Structured Query Language)最早是IBM的圣约瑟研究实验室为其关系数据库管理系统SYSTEM R开发的一种查询语言，它的前身是SQUARE语言。SQL语言结构简洁，功能强大，简单易学，所以自从IBM公司1981年推出以来，SQL语言得到了广泛的应用。如今无论是像Oracle、Sybase、Informix、SQL Server这些大型的数据库管理系统，还是像Visual Foxpro、PowerBuilder这些PC上常用的数据库开发系统，都支持SQL语言作为查询语言。

查询分析器的使用(4)--sql语句

SQL语言包含4个部分：

数据定义语言(DDL)，例如：CREATE、DROP、ALTER等语句。

数据操作语言(DML)，例如：INSERT、UPDATE、DELETE语句。

数据查询语言(DQL)，例如：SELECT语句。

数据控制语言(DCL)，例如：GRANT、REVOKE、COMMIT、ROLLBACK等语句。

查询分析器的使用(5)

使用查询分析器再次完成梁山好汉管理数据库

1、建库

2、建表

英雄表 hero

[这里涉及到对表的几个重要概念：a、字段；b、字段类型(简单介绍)；c、表行；d、表列；e、记录]

1. 对表进行增加、删除、修改、查询的操作

查询分析器中创建数据库

--创建数据库

create database LiangshanHero2

--创建表

use LiangshanHero2--使用指定数据库

go--执行

create table hero--表名

(heroId int ,--英雄排名

heroName varchar(50),--名字

heroNickName varchar(50),--绰号

sex char(2),--性别

sal int

)

--删除一张表(把表的结构和表的数据删除)

drop table hero

--使用sql语句来添加数据

insert into hero values(1,'宋江','及时雨','男',20000)

insert into hero values(2,'卢俊义','玉麒麟','男',15000)

insert into hero values(3,'吴用','智多星','男',30)

insert into hero values(4,'公孙胜','入云龙','男',80)

--使用sql语句查询数据,最基础的查询语句

select \* from hero

--1、查询工资低于100的同志

select \* from hero where sal<100

--2、把工资低于100的人工资提高10%(update)

--语法结构：update 表名 set 字段名1=？,字段名2=？...where 条件语句

update hero set sal=sal\*1.1 where sal<100

--3、请删除工资性别为女的

delete from hero where sex='男'

表的管理--表名和列的命名规则

1、必需以字母,\_开头

2、长度不能超过128个字符

3、不要使用sql server的保留字

4、只能使用如下字符A-Z，a-z，0-9，$，#，\_等

表的管理--支持的数据类型

字符型

char 定长 最大8000字符(非unicode编码)

char(10)'小韩'前四个字符放'小韩'，后添6个空格补全

varchar 变长 最大8000字符(非unicode编码)

varchar(10)'小寒'sql server分配四个字符，这样可以节省空间

ntext可变长度Unicode数据的最大长度为2的30次方-1(1,073,741,823)个字符

text可变长度非Unicode数据的最大长度为2的31次方-1(2,147,483,647)个字符

区别：

1、text是字节格式存储英文的，也可以存中文但有时候会显示成乱码

2、ntext是多字节格式存储unicode的，也就是存储各种文字用的。

在什么时候使用char型而不使用varchar,在知道字段定长固定时就使用char

字符型

nchar 定长 最大4000字符(unicode编码)

nchar(10)'小韩'前四个字符放'小韩'，后添6个空格补全

nvarchar 变长 最大4000字符(unicode编码)

nvarchar(10)'小寒'sql server分配四个字符，这样可以节省空间

**特别说明：**

1、一般带有汉字的字段用nvarchar，全英文或符号的用varchar，因为nvarchar为unicode字符集，该类型的字段无论是单个字母还是单个汉字都占两个字节，而varchar，字母占一个字节，汉字占两个，nvarchar处理汉字或其它unicode字符集的速度要比varchar字段快。

2、如果有一些特殊字符在nvarchar中没有的，如日文的某些名片，那当然只能选合适的代码页用varchar了，而且这些特有语言特有的字符转换到nvarchar会消失

表的管理--支持的数据类型

数字型

1、bit 范围 0到1

2、int 范围 负的2的31次方到正的 2的31次方-1

3、bigint 范围 负的2的63次方到正的 2的63次方-1

4、float 存放小数，不推荐使用

5、numeric 小数

强烈建议 如果要去存放小数最好使用numeric

日期类型

datetime(表示日期)

timestamp(时间戳)

一般情况下 用datetime表示日期

--sql server为我们提供一个专门的时间函数

getdate()

create table spname--建表

(bir datetime)--字段

insert into spname values(getdate())--添加时间

图片

image保存图片，但是用的很少，一般用路径保存图片，在软件公司往往使用图片服务器和图床技术

视频

binary字段可以存放，但是我们往主，将视频文件保存在文件服务器上，sql server中只保留文件路径，存取效率高。

表的管理--怎样创建表

建表(公司管理系统)

--职员表

sql server建议表的设计者，最好给表定义一个主键，用来标示唯一的一条记录。

1、主键(primary key)，不能够重复出现；

2、主键必需给值，换言之主键不能为null

3、主键可以修改，但不能修改为同名的，同时要明确字段不有重复，否则报错。

表的管理--查询

查询表是对表操作的最重要的操作，这个后面详细讲解

表的管理--添加数据

所有字段都插入

insert into emp values(7521,'ward','salesman',7698,'1981-2-22',1250.00,500.00,30)

插入部分字段(语法)

insert into 表名 (字段名1,字段名2,...) values (对应字段数据1,对应字段数据2,...)

表的管理--修改数据

改一个字段(语法)

update 表名 set 字段名='新值' where 字段='值'

例：update student set sex='女' where xh='A001'

update clerk set age=38 where cleName='贾政'

修改多个字段(语法)

update 表名 set 字段名1='新值',字段名2='新值'... where 字段='值'

例：update student set sex='男',birthday='1980-04-01' where xh='A001'

update clerk set cleName='薛蟠',age=40 where cleId=2

修改含有null值的数据(语法)

update 表名 set 字段名='新值' where 字段 is null

例：update clerk set cleName='薛宝钗' where age is null

表的管理--删除数据

删除全部数据

delete from 表名

删除指定数据

delete from 表名 where 字段名='值'

删除多个指定数据(and是条件满足，or是前提条件不满足时使用)

delete from 表名 where 字段名1='值' and 字段名2='值' and或or ...

表的基本查询--介绍(重点)

select语句在软件编程中非常的有用，要好好掌握。

介绍主键和外键

主键，一张表中只能有一个主键

1、主键(primary key)，不能够重复出现；

2、主键必需给值，换言之主键不能为null

3、主键可以修改，但不能修改为同名的，同时要明确字段不有重复，否则报错。

外键

1、外键只能指向主键

2、外键和主键的数据类型要一致

数据库例解主键、外键、多功能查询，见下例：

--建库

create database test

--创建dept表

create table dept--部门表

(deptno int primary key,--部门编号

dname nvarchar(30),--部门名称

loc nvarchar(30)--所在地

)

--建表emp

create table emp--表名

(empno int primary key,--编号

ename nvarchar(30),--名字

job nvarchar(30),--职位

mgr int,--上级编号

hiredate datetime,--入职日期

sal numeric(10,2),--薪水

comm numeric(10,2),--奖金

deptno int foreign key references dept(deptno)--部门编号(做成外键)

)

--针对外键，请注意：

--1、外键只能指向主键

--2、外键和主键的数据类型要一致

--向dept表中添加数据

insert into dept values (10,'accounting','new york')

insert into dept values (20,'research','dallas')

insert into dept values (30,'sales','chicago')

insert into dept values (40,'operations','boston')

--向emp表中添加数据

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7369,'smith','clerk',7902,'1980-12-17',800.00,20)

insert into emp values(7499,'allen','salesman',7698,'1981-2-20',1600.00,300.00,30)

insert into emp values(7521,'ward','salesman',7698,'1981-2-22',1250.00,500.00,30)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7566,'jones','manager',7839,'1981-4-2',2975.00,20)

insert into emp values(7654,'martin','salesman',7698,'1981-9-28',1250.00,1400.00,30)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7698,'blake','manager',7839,'1981-5-1',2850.00,30)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7782,'clark','manager',7839,'1981-6-9',2450.00,10)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7788,'scott','analyst',7566,'1987-4-19',3000.00,20)

insert into emp (empno,ename,job,hiredate,sal,deptno) values (7839,'king','president','1981-11-17',5000.00,10)

insert into emp values (7844,'turner','salesman',7698,'1981-9-8',1500.00,0.00,30)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7876,'adams','clerk',7788,'1987-5-23',1100.00,20)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7900,'james','clerk',7698,'1981-12-3',950.00,30)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7902,'ford','analyst',7566,'1981-12-3',3000.00,20)

insert into emp (empno,ename,job,mgr,hiredate,sal,deptno) values (7934,'miller','clerk',7782,'1982-1-23',1300.00,10)

--查看dept表内容

select \* from dept

--查看emp表内容

select \* from emp

--查询所有列

--语法：select \* from 表名 where 条件

select \* from emp

--查询指定列(sql sever不区分大小写,oracle区分大小写)

--语法：select 字段1,字段2 from 表名 where 条件

--查询smith的薪水，工作，所在部门

select sal,job,deptno from emp where ename='smith'

--取消重复行(distinct只能消除完全一样的行，保留一行)

--语法：select distinct 字段 from 表名 where 条件

--统计共有多少个部门编号

select distinct deptno from emp

--使用算数表达式

--显示每个雇员的年工资+奖金

select ename"姓名",sal\*13+isnull(comm,0)\*13"年收入" from emp

--使用where子句

--如何显示工资高于3000的员工

select ename"员工姓名",sal"工资高于3000" from emp where sal>3000

--如何查找1982-1-1后入职的员工

select ename"员工姓名",hiredate"1982-1-1后入职日期" from emp where hiredate>'1982-1-1'

--如何显示工资在2000到2500的员工情况

select \* from emp where sal>=2000 and sal<=2500

select \* from emp where sal between 2000 and 2500

--如何使用like操作符(模糊查询)

--%：表示0到多个字符 \_:表示单个字符

--如何显示首字符为S的员工姓名和工资

select ename,sal from emp where ename like 's%'

--如何显示第三个字符为o的所有员工的姓名和工资

select ename,sal from emp where ename like '\_\_O%'

--在where条件中使用in

--如何显示empno为123,345,800...的雇员情况

select \* from emp where empno in(123,345,800)

--使用is null的操作符

--如何显示没有上级的雇员的情况

select \* from emp where mgr is null

--使用逻辑操作符号

--查询工资高于500或是岗位为manager的雇员，同时还要满足他们的姓名首写字母为大写的J

select \* from emp where (sal>500 or job='manager') and ename like 'J%'

--使用order by字句(asc默认是升序排列,desc为降序排列)

--如何按照工资的从低到高的顺序显示雇员的信息

select sal from emp order by sal asc

--按照部门号升序而雇员的工资降序排列

--order by可以根据不同的字段排序

select deptno,sal from emp order by deptno,sal desc

--使用列的别名排序(别名需要使用""号圈中)

select ename,sal\*12"年薪" from emp order by "年薪"

表的复杂查询

说明：

在实际应用中经常需要执行复杂的数据统计，经常需要显示多张表的数据，现在我们给

大家介绍较为复杂的select语句

数据分组 -max(最大),min(最小),avg(平均),sum(和),count(统计)

见下例：

--如何显示所有员工中最高工资和最低工资

select min(sal)"最低工资",max(sal)"最高工资" from emp

--显示最低工资并显示出雇员名字

select ename"姓名",sal"工资" from emp where sal=(select min(sal) from emp)

--显示所有员工的平均工资和工资总和

select avg(sal)"平均工资",sum(sal)"工资总和" from emp

--把高于平均工资的雇员的名字和他的工资显示出来

select ename,sal from emp where sal>(select avg(sal) from emp) order by sal

--计算共有多少员工

select count(\*)"雇员数" from emp

--扩展要求

--请显示工资最高的员工的名字、工作岗位

select ename,job from emp where sal=(select max(sal) from emp)

--请显示工资高于平均工资的员工信息,并显示平均工资(效率不高)

select ename"姓名",job"岗位",sal"工资",(select avg(sal) from emp)"平均工资" from emp where sal>(select avg(sal) from emp) order by sal

表的复杂查询

group by和having子句

group by用于对查询的结果分组统计

having子句用于限制分组显示结果

例：

--group by和having子句使用(having与group by结合使用，可以对分组后的查询结果进行筛选)

--如何显示每个部门的平均工资和最高工资

select deptno"部门号",avg(sal)"平均工资",max(sal)"最高工资",min(sal)"最低工资" from emp group by deptno

--如何显示每个部门的平均工资和最高工资并显示部门名称(多表查询)

select emp.deptno"部门号",avg(sal)"平均工资",max(sal)"最高工资",min(sal)"最低工资",dept.dname"部门名称" from emp,dept where emp.deptno=dept.deptno group by emp.deptno,dept.dname

--显示每个部门的每种岗位的平均工资和最低工资

select deptno"部门号",job"岗位",avg(sal)"平均工资",min(sal)"最低工资" from emp group by deptno,job order by deptno

--显示平均工资低于2000的部门号和它的平均工资

select deptno"部门号",avg(sal)"平均工资" from emp group by deptno having avg(sal)<2000

--显示平均工资大于2000的部门号和它的平均工资并按升序排列

select deptno"部门号",avg(sal)"平均工资" from emp group by deptno having avg(sal)>2000 order by "平均工资"

表的复杂查询

对数据分组的总结

1、分组函数只能出现在选择列表，having、order by子句中

2、如果在select语句中同时包含有group by，having，ovrder by那么他们的顺序是group by，having，order by

3、在选择列中如果有列、表达式、和分组函数，那么这些列和表达式必需有一个出现在group by子句中，否则就会出错

如：select deptno,avg(sal),max(sal) from emp group by deptno having avg(sal)<2000

这里deptno就一定要出现在group by中

表的复杂查询--多表查询

说明：

多表查询是指基于两个或两个以上的表或是视图的查询，在实际应用中，查询单个表可能不能满足你的需求，(如显示sales部门位置和其员工的姓名)，这种情况下需要使用到(dept表和emp表)

--多表查询

--如果多张表都有相同名字的字段，则需要带表名(别名)

--显示sales部门位置和其员工的姓名

select emp.ename"员工姓名",dept.dname"部门名称",dept.loc"部门所在地" from emp,dept where dept.dname='sales' and emp.deptno=dept.deptno

--显示雇员名，雇员工资及所在部门的名字

select emp.ename"雇员名",emp.sal"雇员工资",dept.dname"部门名称" from emp,dept where emp.deptno=dept.deptno

--如何显示部门号为10的部门名、员工名和工资

select emp.deptno"部门号",dept.dname"部门名称",emp.ename"员工名称",emp.sal"工资" from emp,dept where emp.deptno=dept.deptno and emp.deptno=10

--显示雇员名、雇员工资及所在部门的名字并按部门排序

select e.ename"雇员名字",e.sal"工资",d.dname"部门名称" from emp e,dept d where e.deptno=d.deptno order by d.dname

表的复杂查询--多表查询

自连接

自连接是指在同一张表的连接查询

例：

--自连接

--显示某个员工的上级领导的姓名，比如显示"ford"的上级

select (select ename from emp where ename='ford')"员工姓名",ename"上级领导" from emp where empno=(select mgr from emp where ename='ford')

--显示公司每个员工姓名和他的上级的名字

--分析，把emp表看成两张表分别是worker/boss

--外连接(左外连接、右外连接)

select worker.ename"员工名字",boss.ename"领导名字" from emp worker,emp boss where worker.mgr=boss.empno

表的复杂查询--子查询

什么是子查询

子查询是批嵌入在其它sql语句中的select语句，也叫嵌套查询

单行子查询

单行子查询是指只返回一行数据的子查询语句

--如何显示与smith同一部门的所有员工？

select deptno"部门号",ename"员工名字" from emp where deptno=(select deptno from emp where ename='smith')

多行子查询

多行子查询指返回多行数据的子查询

--如何查询和部门10的工作相同的雇员的名字、岗位、工资、部门号

select \* from emp where job in(select distinct job from emp where deptno=10)

在from子句中使用子查询

--如何显示高于部门平均工资的员工名字、薪水、部门的平均工资

--分析：1、首先要知道各个部门的平均工资

select avg(sal)"部门平均工资",deptno from emp group by deptno

--2、把上面的查询结果当作一个临时表对待

select e.ename"员工名字",e.sal"薪水",temp.myavg"部门平均工资",e.deptno from emp e,(select avg(sal) myavg,deptno from emp group by deptno) temp where e.deptno=temp.deptno and e.sal>temp.myavg

在from子句中使用子查询

这里需要说明的当在from子句中使用子查询时，该子查询会被作为一个临时表来对待，当在from子句中使用子查询时，必需给子查询指定别名

分面查询

按雇员的id号升序取出

--请显示第5个到第10个入职的雇员信息(按照入职的时间先后顺序查找)

--分析：1、显示第1个到第4个入职的雇员

select top 4 \* from emp order by hiredate

--top后的数表示要取出几条记录

select top 6 \* from emp where empno not in(select top 4 empno from emp order by hiredate) order by hiredate

--请显示第11个到13个入职的雇员信息

select top 3 \* from emp where empno not in(select top 10 empno from emp order by hiredate) order by hiredate

--请显示第5个到9个入职的雇员信息(按薪水高低排序)

select top 5 \* from emp where empno not in(select top 4 empno from emp order by sal desc) order by sal desc

用查询结果创建新表

这个命令是一种快捷的建表方法

select \*(这里可以选择字段) into 另一个表名 from 表

--如何删除掉一张表重复记录

create table cat(

catId int,

catName varchar(40)

)

insert into cat values(1,'aa')

select \* from cat

--1、把cat表的记录distinct后的结果，放到临时表中

select distinct \* into #temp from cat

--2、把cat表的记录清空

delete from cat

--3、把临时表中的数据信息加入到cat表中

insert into cat select \* from #temp

--4、删除临时表

drop table #temp

--左外连接和右外连接

--思考题：显示公司每个员工和他的上级领导的名字

--内连接的处理方式(内连接只显示匹配的信息)

select worker.ename"员工名字",boss.ename"领导名字" from emp worker,emp boss where worker.mgr=boss.empno

--思考题：显示公司每个员工和他的上级领导的名字，没有上级领导的也要显示出来

--左外连接：指如果左边的表记录全部显示，如果没有匹配的记录，就用null填写

select worker.ename"员工名字",boss.ename"领导名字" from emp worker left join emp boss on worker.mgr=boss.empno

--右外连接：指如果右边的表记录全部显示，如果没有匹配的记录，就用null填写

select worker.ename"员工名字",boss.ename"领导名字" from emp worker right join emp boss on worker.mgr=boss.empno

**维护数据的完整性--约束**

约束用于确保数据库数据满足特定的商业规则。在sql server和oracle中，约束包括：not null、unique，primary key，foreign key和check五种

维护数据的完整性--使用

not null(非空)

如果在列上定义了not null，那么当插入数据时，必需为列提供数据。

--约束机制--not null(非空)

--创建一张表

create table test1

(test1Id int primary key identity(1,1),

testname varchar(30) not null,--not null不能为空

testpass varchar(30) not null,

testage int --不写代表可以为空

)

create table test1

(test1Id int primary key identity(1,1),--identity(1,1)自增长1条记录

testname varchar(30),

testpass varchar(30),

testage int --不写代表可以为空

)

--删除表

drop table test1

--向表插入数据

insert into test1 (testage) values (3)

insert into test1 (testname,testpass,testage) values ('','',5)--''空与null空是不一样的

--查询表内容

select \* from test1

unique(唯一)(一张表中可以有多个)

当定义了唯一约束后，该列值是不能重复的，但是可以为null,并只能有一个空值

--约束机制--unique(唯一)

--建表

create table test2

(test1Id int primary key identity(1,1),--identity(1,1)自增长1条记录

testname varchar(30) unique, --unique唯一的，数据不允许重复,但可以为空

testpass varchar(30),

testage int --不写代表可以为空

)

insert into test2 (testname,testpass,testage) values ('aa','123',45)

insert into test2 (testpass,testage) values ('123',45)

select \* from test2

primary key(主键)(一张表中只可以有一个主键)

用于唯一的标示表行的数据，当定义主键约束后，该列不但不能重复而且不能为null

需要说明的是：一张表最多只能有一个主键，但是可以有多个unqiue约束。

表可以有复合主键，有多个列构成一个主键

--复合主键

create table test3

(testId int,

testname varchar(30),

testpass varchar(30),

testage int,

primary key (testId,testname)--复合主键，需单独声明

)

--行级定义和表级定义

create table test4

(testId int,--在字段中定义主键为行级定义.例：testId int primary key

testname varchar(30),

testpass varchar(30),

testage int,

primary key (testId,testname)--复合主键，需单独声明为表级定义

)

foreign key(外键)(外键在从表上，要配合主表，但主表要有主键或unique约束)

用于定义主表和从表之间的关系。外键约束要定义在从表上，主表则必需具有主键约束或是unique约束，当定义外键约束后，要求外键列数据必需在主表的主键列存在或是null

check(强制条件)

用于强制行数据必需满足的条件，假定在sal列上定义了check约束，并要求sal列值在1000-2000之间如果不再1000-2000之间就会提示出错。

--check(强制条件)

create table test5

(testId int,

testname varchar(30),

testpass varchar(30),

sal int check(sal>=1000 and sal<=2000),--规定sal的值在1000-2000之间

)

insert into test5 values (4,'aa','aa',1200)--可以加入sal值在范围之内

insert into test5 values (5,'bb','bb',2200)--不可加入sal值在范围之外

select \* from test5

default使用(默认值)

--default使用

create table mes

(mesId int primary key identity(1,1),

mesccn varchar(2000) not null,

mesDate datetime default getdate()--在不指定值是default可以直接取默认值，也可以由用户指定值

)

insert into mes(mesccn) values('abc')

insert into mes(mesccn,mesDate) values('cba','1976-1-6')

select \* from mes

商店售货系统表设计案例

现有一个商店的数据库，记录客户及其购物情况，由下面三个表组成：

商品goods(商品号goodsId，商品名goodsName，单价unitprice，商品类别category，供应商provider)；

客户customer(客户号customerId，姓名name，住址address，电邮email，性别sex，身份证cardId)；

购买purchase(客户号customerId，商品号goodsId，购买数量nums)；

请使用sql语言完成下列功能：

1、建表，在定义中要求声明：

1、每个表的主外键；

2、客户的姓名不能为空值；

3、单价必需大于0，购买数量大于0；

4、电邮不能够重复；

5、客户的性别必需是男或者女，默认为男

6、商品类别是'食物'，'日用品'

--商店售货系统表设计案例

create table goods--商品表

(goodsId nvarchar(50) primary key,--商品Id为主键

goodsName nvarchar(80) not null,--商品名不为空

unitprice numeric(10,2) check (unitprice>0),--商品单价限制大于0

category nvarchar(3) check(category in('食物','日用品')),--商品类别需在指定的范围内

provide nvarchar(50)--供应商

)

create table customer--客户表

(custcmerId nvarchar(50) primary key,--客户Id为主键

cusname nvarchar(50) not null,--客户名不能为空

address nvarchar(100),--地址

email nvarchar(100) unique,--邮件不为重复

sex nchar(1) check(sex in('男','女')) default '男',--性别只能为男或女，默认为男

cardId nvarchar(18)--身份证

)

create table purchase--购买表

(custcmerId nvarchar(50) foreign key references customer(custcmerId),--客户Id为外键，并指向customer表的custcmerId主键

goodsId nvarchar(50) foreign key references goods(goodsId),--商品Id为外键，并指向goods表的goodsId主键

nums int check(nums>0)--购买数量限定大于0

)

表的管理--修改表

添加一个字段

ALTER TABLE distributors ADD COLUMN address varchar(30);

修改字段的类型/或是名字(不能有数据)

ALTER TABLE distributors

ALTER COLUMN address TYPE varchar(80),

ALTER COLUMN name TYPE varchar(100);

ALTER TABLE distributors RENAME COLUMN address TO city;

删除一个字段

ALTER TABLE distributors DROP COLUMN address RESTRICT;

修改表的名字

ALTER TABLE distributors RENAME TO suppliers;

删除表

drop table 表名;

数据库的备份和恢复

使用企业管理器完成备份和恢复

使用企业管理器有两种方式完成备份和恢复

1、分离/附加

分离完后，请到sql server安装的目录下去找两个文件数据库名.mdf和数据库名.ldf，这两个文件就是分离后的文件，数据库分离后，该数据库就不能再使用了。

附加是指，当用户需要重新使用某个分离的数据库时进行的操作，就是让sql server数据库重新关联该数据库。

2、备份/恢复

备份数据库是指，把某个数据库文件从sql server中备份出来，这样用户可以根据需要再使用(用于恢复、复用..)，备份数据库不会影响到源数据库的使用

恢复数据库是指，当源数据库因为某种原因(比如源数据库毁坏、数据丢失、系统崩溃)需要恢复时进行的操作

使用查询分析器完成备份和恢复

用企业管理器完成对数据库的备份和恢复简单直观，同样我们也可以在查询分析器中完成类似的任务。

--使用查询分析器对数据库进行备份和恢复

--数据库备份

--**语法：backup database 数据库名 to disk='存储路径'**

backup database LiangshanHero2 to disk='f:/liangshan.bak'

--数据库恢复

--**语法：restore database 数据库名 from disk='读取路径'**

restore database LiangshanHero2 from disk='f:/liangshan.bak'

--新建数据库

--**语法：create datebase 数据库名**

create database LiangshanHero2

--删除数据库

--**语法：drop database 数据库名**

drop database LiangshanHero2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**java程序操作sql server**

crud介绍(增、删、改、查操作)

CRUD是指在做计算处理时的增加(Create)、查询(Retrieve)(重新得到数据)、更新(Update)和删除(Delete)几个单记事的首字母简写。主要被用在描述软件系统中数据库或者持久层的基本操作功能。

Incomputing,CRUD is an acronym for create,retrieve,update,and delete.It is used to refer to the basic functions of a database or persistence layer in a software system.

Create new records

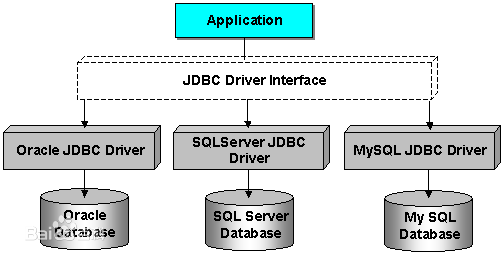
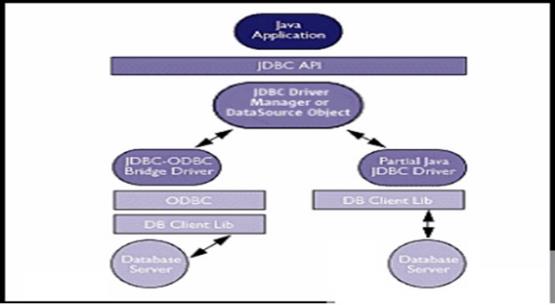
Rctricvc cxisting rccords

Update existing records

Delete existing records.

crud介绍

要对数据表进行增、删、改、查，我们首先要清楚jdbc基本的概念：



JDBC有两种，一种原sun公司提供的数据库连接api但不是直接连接sql server而是先连接ODBC再通过ODBC对sql server进行操作；一种是由微软提供的JDBC数据库连接api可直接对sql server数据库进行操作。

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序，同时，JDBC也是个商标名。

有了JDBC，向各种关系数据发送SQL语句就是一件很容易的事。换言之，有了JDBC API，就不必为访问Sybase数据库专门写一个程序，为访问Oracle数据库又专门写一个程序，或为访问Informix数据库又编写另一个程序等等，程序员只需用JDBC API写一个程序就够了，它可向相应数据库发送SQL调用。同时，将Java语言和JDBC结合起来使程序员不必为不同的平台编写不同的应用程序，只须写一遍程序就可以让它在任何平台上运行，这也是Java语言“编写一次，处处运行”的优势。

注：JDBC访问不同的数据库使用的JDBC API都有所不同，虽然统称为JDBC。如：访问sql server数据库的JDBC，是不能访问oracle数据库的，同理也是也不访问mysql、db2、Informix数据库等。只是访问方式不同，对数据库的操作依然是使用sql语句操作

Java 具有坚固、安全、易于使用、易于理解和可从网络上自动下载等特性，是编写数据库应用程序的杰出语言。所需要的只是 Java应用程序与各种不同数据库之间进行对话的方法。而JDBC 正是作为此种用途的机制。

java程序操作sql server

jdbc的驱动的分类

目前比较常见的JDBC驱动程序可分为以下四个种类

1、jdbc-odbc桥连接

2、本地协议纯java驱动程序

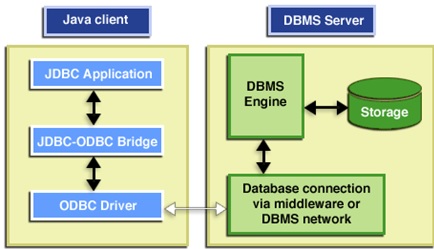
3、网络协议纯java驱动程序

4、本地api

jdbc不足

尽管JDBC在JAVA语言层面实现了统一，但不同数据库仍旧有许多差异。为了更好地实现跨数据库操作，于是诞生了Hibernate项目，Hibernate是对JDBC的再封装，实现了对数据库操作更宽泛的统一和更好的可移植性。

java操作sql server数据库(表)



1、我们先使用jdbc-odbc桥连的方式来操作sql server数据库(表)完成对emp表的crud操作[Sql\_test1.java]

2、再用jdbc驱动程序直接操作sql server数据库(表)完成对emp表的crud操作。

/\*\*

\* 演示使用jdbc-odbc桥连方式操作sql server数据库

\* 具体操作test数据库的emp表和dept表

\* 1、配置数据源--windows下在控制面板-->管理工具-->ODBC数据源-->用户DSN--添加

\* sql server

\* 2、在程序中去连接数据源

\* 3、使用Statement(不安全)连接数据库

\*/

package com.sqlserver;

import java.sql.\*;

public class Sql\_test1 {

public static void main(String[] args) {

Connection ct=null;

Statement sm=null;

ResultSet rs=null;

try {

//1、加载驱动(把需要的驱动程序加入内存)

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

//2、得到连接(指定连接到哪个数据源、数据库的用户名和密码)

//如果配置数据源的时候选择的是windows NT验证方式，则不需要数据库的用户名和密码

//Connection ct=DriverManager.getConnection("jdbc:odbc:mytest");

ct=DriverManager.getConnection("jdbc:odbc:mytest","sa","sa");

//3、创建Statement或者PreparedStatement(区别)

//Statement用处：主要用于发送SQL语句到数据库

sm=ct.createStatement();

//4、执行(CRUD，创建数据库、备份数据库、删除数据库)

//演示1：添加一条数据到dept表中

//executeUpdate可以执行CUD操作(添加、删除、修改)

int i=sm.executeUpdate("insert into dept values('60','保安部','西永')");

if(i==1){

System.out.println("数据添加成功");

}else{

System.out.println("添加失败");

}

//演示2：从dept表中删除一条记录

int j=sm.executeUpdate("delete from dept where deptno='50'");

if(j==1){

System.out.println("数据删除成功");

}else{

System.out.println("删除失败");

}

//演示3：从dept表中修改deptno=40 loc改为beijing

int k=sm.executeUpdate("update dept set loc='beijing' where deptno='40'");

if(k==1){

System.out.println("数据修改成功");

}else{

System.out.println("修改失败");

}

//演示4：查询,显示所有的部门信息

//ResultSet结果集,大家可以把ResultSet理解成返回一张表行的结果集

rs=sm.executeQuery("select \* from dept");

//循环取出

while(rs.next()){

int a=rs.getInt(1);

String b=rs.getString(2);

String c=rs.getString(3);

System.out.println(a+"\t"+b+"\t"+c);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

//关闭资源，关闭顺序先创建后关闭，后创建先关闭

try {

//为了程序健壮

if(rs!=null){

rs.close();

}

if(sm!=null){

sm.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

java程序操作sql server

Statement和PreparedStatement的区别(1)

Statemen和PreparedStatement都可以用于把sql语句从java程序中发送到指定数据库，并执行sql语句，但是他们也存在区别：

1、直接使用Statement，驱动程序一般不会对sql语句作处理而直接交给数据库；使用PreparedStamen，形成预编译的过程，并且会对语句作字符集的转换(至少在sql server)中如此。

如此，有两个好处：对于多次重复执行的语句，使用PreparedStament效率会更高一点，并且在这种情况下也比较适合使用batch；另外，可以比较好地解决系统的本地化问题。

2、PreparedStatement还能有效的防止危险字符的注入，也就是sql注入的问题。

PreparedStatement的使用[Sql\_test2.java]

/\*\*

\* PreparedStatement使用CRUD

\* 1、PreparedStatement可以提高执行效率(因为它有预编译的功能)

\* 2、PreparedStatement可以防止SQL注入，但是要求用?赋值的方式才可以

\*/

package com.sqlserver;

import java.sql.\*;

public class Sql\_test2 {

public static void main(String[] args) {

Connection ct=null;

PreparedStatement ps=null;

ResultSet rs=null;

try {

//1、加载驱动(把需要的驱动程序加入内存)

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

//2、得到连接(指定连接到哪个数据源、数据库的用户名和密码)

ct=DriverManager.getConnection("jdbc:odbc:mytest","sa","sa");

//3、创建PreparedStatement

//PreparedStatement用处：主要用于发送SQL语句到数据库

// ps=ct.prepareStatement("select \* from dept where deptno=? and loc=?");

// //给?赋值(可防止SQL注入漏洞问题)，不要直接使用拼接的方式

// ps.setInt(1, 20);

// ps.setString(2, "dallas");

// //演示：查询,显示所有的部门信息

// //ResultSet结果集,大家可以把ResultSet理解成返回一张表行的结果集

// rs=ps.executeQuery();

// //循环取出

// while(rs.next()){

// int a=rs.getInt(1);

// String b=rs.getString(2);

// String c=rs.getString(3);

// System.out.println(a+"\t"+b+"\t"+c);

// }

//使用PreparedStetement添加一条记录

// ps=ct.prepareStatement("insert into dept values(?,?,?)");

// ps.setInt(1, 60);

// ps.setString(2, "安全部");

// ps.setString(3, "上海");

// //执行

// int i=ps.executeUpdate();

// if(i==1){

// System.out.println("添加成功");

// }else{

// System.out.println("添加失败");

// }

//使用PreparedStetement修改一条记录从dept表中修改loc=上海 deptno改为50

// ps=ct.prepareStatement("update dept set deptno=? where loc='上海'");

// ps.setInt(1, 50);

// //执行

// int i=ps.executeUpdate();

// if(i==1){

// System.out.println("修改成功");

// }else{

// System.out.println("修改失败");

// }

//使用PreparedStetement删除一条记录

ps=ct.prepareStatement("delete from dept where deptno=?");

ps.setInt(1, 50);

int i=ps.executeUpdate();

if(i==1){

System.out.println("删除成功");

}else{

System.out.println("删除失败");

}

} catch (Exception e) {

e.printStackTrace();

}finally{

//关闭资源，关闭顺序先创建后关闭，后创建先关闭

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

JDBC-ODBC桥连操作sql server与JDBC驱动直连操作sql server的区别：

1、JDBC-ODBC桥连sql server无需引入外部驱动

2、JDBC直连需要引入微软提供的JDBC驱动

JDBC直连sqlserver示例[Sql\_test3.java]

/\*\*

\* JDBC方式去操作数据库(单表操作)

\* 1、引入sql包

\* 2、sqlserver2000需要引入三个jar包,分别是msbase.jar和mssqlserver.jar和msutil.jar

\* 3、sqlserver2005/2008/2012版本中可以引入sqljdbc.jar或sqljdbc4.jar两个微软提供的JDBC包，官方目前推出2.0/3.0/4.0版本

\* 4、使用sqljdbc4.jar后可以不使用加载驱动Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");而直接连接数据库

\* 5、使用sqljdbc.jar则需要加载驱动

\* 6、特别说明，如果取值是按编号取，则需一一对应；如果按字段列名取值，则可以灵活取值

\*/

package com.sqlserver;

import java.sql.\*;

public class Sql\_test3 {

public static void main(String[] args) {

//定义需要的对象

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

try {

//初始化我们的对象

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=test;user=sa;password=sa;");

if(!ct.isClosed()){

System.out.println("数据库连接成功");

}else{

System.out.println("数据库连接失败");

}

//3、创建PreparedStatement

// ps=ct.prepareStatement("select \* from emp");

ps=ct.prepareStatement("select ename,sal,deptno from emp");

//4、执行(查询就用executeQuery()，增加、删除、修改就用executeUpdate())

rs=ps.executeQuery();

//循环取出,雇员的名字、雇员的薪水，部门编号

while(rs.next()){

// String ename=rs.getString(2);

// float sal=rs.getFloat(6);

// int deptno=rs.getInt(8);

// String ename=rs.getString(1);//对应的列号可以取出数据

// float sal=rs.getFloat(2);

// int deptno=rs.getInt(3);

String ename=rs.getString("ename");//使用字段名也可以取出数据

float sal=rs.getFloat("sal");

int deptno=rs.getInt("deptno");

System.out.println(ename+"\t"+sal+"\t"+deptno);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

多表操作，示例[Sql\_test4.java]

/\*\*

\* JDBC方式去操作数据库(多表操作)

\* 1、引入sql包

\* 2、sqlserver2000需要引入三个jar包,分别是msbase.jar和mssqlserver.jar和msutil.jar

\* 3、sqlserver2005/2008/2012版本中可以引入sqljdbc.jar或sqljdbc4.jar两个微软提供的JDBC包，官方目前推出2.0/3.0/4.0版本

\* 4、使用sqljdbc4.jar后可以不使用加载驱动Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");而直接连接数据库

\* 5、使用sqljdbc.jar则需要加载驱动

\* 6、特别说明，如果取值是按编号取，则需一一对应；如果按字段列名取值，则可以灵活取值

\*/

package com.sqlserver;

import java.sql.\*;

public class Sql\_test4 {

public static void main(String[] args) {

//定义需要的对象

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

try {

//初始化我们的对象

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=test;user=sa;password=sa;");

if(!ct.isClosed()){

System.out.println("数据库连接成功");

}else{

System.out.println("数据库连接失败");

}

// //3、创建PreparedStatement

// ps=ct.prepareStatement("select ename,sal,dname from emp,dept where emp.deptno=dept.deptno");

// //4、执行(查询就用executeQuery()，增加、删除、修改就用executeUpdate())

// rs=ps.executeQuery();

// //循环取出,雇员的名字、雇员的薪水，部门名称

// while(rs.next()){

// String ename=rs.getString("ename");//使用字段名也可以取出数据

// float sal=rs.getFloat("sal");

// String dname=rs.getString("dname");

// System.out.println(ename+"\t"+sal+"\t"+dname);

// }

// //5、创建PreparedStatement

// ps=ct.prepareStatement("insert into dept values(?,?,?)");

// //6、给?赋值

// ps.setInt(1, 60);

// ps.setString(2, "Finance");

// ps.setString(3, "Alaska");

// //7、执行(查询就用executeQuery()，增加、删除、修改就用executeUpdate())

// int i=ps.executeUpdate();

// if(i==1){

// System.out.println("添加数据成功");

// }else{

// System.out.println("添加数据失败");

// }

// //8、创建PreparedStatement

// ps=ct.prepareStatement("delete from dept where deptno=?");

// //9、给?赋值

// ps.setInt(1, 50);

// //10、执行(查询就用executeQuery()，增加、删除、修改就用executeUpdate())

// int i=ps.executeUpdate();

// if(i==1){

// System.out.println("删除数据成功");

// }else{

// System.out.println("删除数据失败");

// }

//11、创建PreparedStatement

ps=ct.prepareStatement("update dept set deptno=? where loc=?");

//12、给?赋值

ps.setInt(1, 50);

ps.setString(2, "Alaska");

//13、执行(查询就用executeQuery()，增加、删除、修改就用executeUpdate())

int i=ps.executeUpdate();

if(i==1){

System.out.println("修改数据成功");

}else{

System.out.println("修改数据失败");

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

java程序操作sql server

Statement和PreparedStatement的区别(2)

看下面两段程序片断：

Code Fragment 1:

String updateString="UPDATE COFFEES SET SALES=75"+"WHERE COF\_NAME LIKE 'Colombian'";

stmt.executeUpdate(updateString);

Code Fragment 2:

PreparedStatement updateSales=con.prepareStatement("UPDATE COFFEES SET SALES=? WHERE COF\_NAME LIKE ?");

updateSales.setInt(1,75);

updateSales.setString(2,"Colombian");

updateSales.executeUpdate();

后者使用了PreparedStatement，而前者是Statement，PreparedStatement不仅包含了SQL语句，而且大多数情况下这个语句已被预编译过，当其执行时，只需DBMS运行SQL语句，而不必先编译。当你需要执行Statement对象多次的时候，PreparedStatement对象将会降低运行时间，加快了访问数据库的速度。

好处是，不必重复SQL语句的句法，而只需要改其中变量的值，便可重新执行SQL语句。选择PreparedStatement对象与否，在于相同句法的SQL语句是否执行了多次，而且两次之间的差别仅是变量的不同。如仅执行一次的话，它和普通的对象无差异，体现不出预编译的优越性。

java操作sql server数据库(表)[Sql\_test5.java]

在软件公司实际开发过程中，也许需要你在java程序中来控制对数据库(表)的创建、删除、备份、恢复工作，这是我们又该怎样完成呢？

/\*\*

\* 在java中如何使用ddl语句(credate(创建),drop(删除),backup(备份),restore(恢复))数据库

\*/

package com.sqlserver;

import java.sql.\*;

public class Sql\_test5 {

public static void main(String[] args) {

//定义需要的对象

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

try {

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=test;user=sa;password=sa;");

if(!ct.isClosed()){

System.out.println("数据库连接成功");

}else{

System.out.println("数据库连接失败");

}

// //3、创建ps，创建数据库

// ps=ct.prepareStatement("create database vvv");

// //如果执行的是ddl语句

// if(!ps.execute()){

// System.out.println("创建数据库成功");

// }else{

// System.out.println("创建数据库失败");

// }

//4、创建ps，创建表

// ps=ct.prepareStatement("create table xxx(aa int)");

// //如果执行的是ddl语句

// if(!ps.execute()){

// System.out.println("创建表成功");

// }else{

// System.out.println("创建表失败");

// }

//5、备份数据库

// ps=ct.prepareStatement("backup database vvv to disk='e:/vvv.bak'");

// if(!ps.execute()){

// System.out.println("备份数据库成功");

// }else{

// System.out.println("备份数据库失败");

// }

//6、恢复数据库

ps=ct.prepareStatement("restore database vvv from disk='e:/vvv.bak'");

if(!ps.execute()){

System.out.println("恢复数据库成功");

}else{

System.out.println("恢复数据库失败");

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

sql server作业点评：

使用scott/tiger用户下的emp表完成下列练习，表的结构说明如下

emp员工表(empno员工号/ename员工姓名/job工作/mgr上级编号/hiredate受雇日期/sal薪金/comm佣金/deptno部门编号)

//把课堂上创建的emp,dept表自己创建一下(参考图)

//根据emp表和dept表，完成下面的练习

--1、选择部门30中的所有员工

select \* from emp where deptno=30

--2、列出所有办事员(CLERK)的姓名，编号和部门编号

select ename,empno,deptno from emp where job='clerk'

--3、找出佣金高于薪金的员工(佣金是奖金)

select \* from emp where isnull(comm,0)>sal

--4、找出佣金高于薪金的60%的员工

select \* from emp where comm>sal\*0.6

--5、找出部门10中所有经理(MANAGER)和部门20中所有办事员(CLERK)的详细资料

select \* from emp where (deptno=10 and job='manager')or(deptno=20 and job='clerk')

--6、找出部门10中所有经理(MANAGER)部门20中所有办事员(CLERK)既不是经理又不是办事员但其薪金大于或等于2000的所有员工的详细资料。

select \* from emp where (deptno=10 and job='manager') or (deptno=20 and job='clerk') or (job<>'manager' and job<>'clerk' and sal>=2000)

--7、找出收取佣金的员工的不同工作

select distinct job from emp where comm>0

--8、找出不收取佣金或收取的佣金低于100的员工

select \* from emp where comm<100 or comm is null

--9、找出各月倒数第3天受雇的所有员工

//oracle中有一个函数可以得到某个日期的最后一天

select \* from emp where hiredate=dateadd(day,-3,datename(year,dateadd(month,1,hiredate))+datename(month,dateadd(month,1,hiredate))+'01')

--10、找出早于12年前受雇的员工

select \* from emp where hiredate<dateadd(dd,-12\*365,getdate())

select \* from emp where datediff(year,hiredate,getdate())>12

--11、以首字母大写的方式显示所有员工的姓名

select upper(substring(ename,1,1))+lower(substring(ename,2,len(ename))) from emp

--12、显示正好为5个字符的员工的姓名

select \* from emp where ename like'\_\_\_\_\_'

select \* from emp where len(ename)=5

--13、显示不带有R的员工的姓名

select ename from emp where ename not like'%R%'

--14、显示所有员工姓名的前三个字符

select substring(ename,1,3) from emp

--15、显示所有员工的姓名，用a替换所有A

select replace(ename,'A','a') from emp

--16、显示满10年服务年限的员工的姓名和受雇日期

select ename,hiredate from emp where datediff(year,hiredate,getdate())>10

--17、显示员工的详细资料，按姓名排序

select \* from emp order by ename

--18、显示员工的姓名和受雇日期，根据其服务年限，将最老的员工排在最前面

select ename,hiredate from emp order by hiredate

--19、显示所有员工的姓名、工作和薪金，按工作的降序排序，若工作相同则按薪金排序

select ename,job,sal from emp order by job desc,sal

--20、显示所有员工的姓名、加入公司的年份和月份，按受雇日期所在月排序,若月份相同则将最早年份的员工排在最前面

select ename,datepart(year,hiredate) y,datepart(month,hiredate) m from emp order by m,y

--21、列出至少有一个员工的所有部门

select count(\*),deptno from emp group by deptno having count(\*)>1

--22、列出薪金比“SMITH”多的所有员工

select \* from emp where sal>(select sal from emp where ename='smith')

--23、列出所有员工的姓名及其直接上级的姓名

select e1.ename,e2.ename from emp e1,emp e2 where e1.mgr=e2.empno

--24、列出受雇日期晚于其直接上级的所有员工

select e1.ename,e1.hiredate,e2.ename,e2.hiredate from emp e1,emp e2 where e1.mgr=e2.empno and e1.hiredate>e2.hiredate

--25、列出部门名称和这些部门的员工信息，同时列出那些没有员工的部门。

select d.dname,e.ename,e.job from emp e right join dept d on e.deptno=d.deptno

--26、列出所有“clerk”(办事员)的姓名及其部门名称

select e.ename,d.dname from emp e,dept d where e.deptno=d.deptno and job='clerk'

--27、列出最低薪金大于1500的各种工作。

select min(sal),job from emp group by job having min(sal)>1500

--28、列出在部门“SALES”(销售部)工作的员工的姓名，假定不知道销售部的部门编号。

select ename,'sales' from emp where deptno=(select deptno from dept where dname='sales')

--29、列出薪金高于公司平均薪金的所有员工。

select \* from emp where sal>(select avg(sal) from emp)

--30、列出与“SCOTT”从事相同工作的所有员工。

select \* from emp where job=(select job from emp where ename='scott')

--31、列出薪金等于部门30中员工的薪金的所有员工的姓名和薪金

select ename,sal from emp where sal in(select sal from emp where deptno=30)

--32、列出薪金高于在部门30工作的所有员工的薪金的员工姓名和薪金

select ename,sal from emp where sal>(select max(sal) from emp where deptno=30)

--33、列出在每个部门工作的员工数量、平均工资和平均服务期限

select count(\*)"员工人数",avg(sal)"部门平均工资",avg(datediff(year,hiredate,getdate()))"部门平均服务期限",deptno from emp group by deptno

--34、列出所有员工的姓名、部门名称和工资

select e.ename,d.dname,e.sal from emp e,dept d where e.deptno=d.deptno

--35、列出从事同一种工作但属于不同部门的员工的一种组合。

select w1.ename,w1.job,w1.deptno,w2.ename,w2.job,w2.deptno from emp w1,emp w2 where w1.job=w2.job and w1.deptno<>w2.deptno

--36、列出所有部门的详细信息和部门人数。

select d2.dname,d2.loc,isnull(d.c,0) from dept d2 left join (select count(\*) c,deptno de from emp group by deptno) d on d2.deptno=d.de

--37、列出各种工作的最低工资

select min(sal),job from emp group by job

--38、列出MANAGER(经理)的最低薪金

select min(sal) from emp where job='manager'

--39、列出所有员工的年工资，按年薪从低到高排序

select ename,(sal+isnull(comm,0))\*12"年工资" from emp order by "年工资"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

项目开发--流程

1、项目需求分析--充分理解客户对项目的要求是什么；形成分析文档

2、设计阶段--决定用什么技术、框架；操作系统；数据库；形成设计文档，开发队伍

3、编码阶段--程序员编码

4、测试阶段

5、实施阶段

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

学生管理系统需求文档

1、功能说明(使用use case图来说明 uml(统一建模语言))

uml(统一建模语言)有两款较好的开发工具来做建模

uml-->IBM-rational rose 2003或7.0

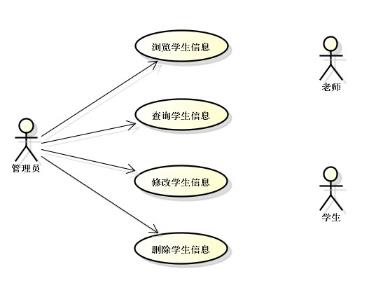
uml-->jude

uml可以做出这样几种图

1、use case图--用例图(可以清晰的描述该系统有什么角色和功能)

2、时序图

3、类图



2、设计界面(原型开发[先搞定界面，在写代码])



3、设计数据库

学生表(stus)

字段名 类型 备注 是否为空字段 字段默认值

stuId varchar(30) 学生ID not null

stuName nvarchar(50) 学生名字 not null

stuSex nchar(1) 性别 not null default'男'

stuAge int 年龄 not null default>1

stuJg nvarchar(20) 籍贯 not null

stuDept nvarchar(30) 所在系 not null

JTable讲解[JTable\_test1.java]

/\*\*

\* JTable的使用

\*/

import javax.swing.\*;

import java.util.\*;

import java.sql.\*;

import java.awt.\*;

import java.awt.event.\*;

public class JTable\_Test1 extends JFrame{

//定义组件

//rowData用来存放行数据、columnNames存放列名

Vector rowData,columnNames;

JTable jt=null;

JScrollPane jsp=null;

public static void main(String[] args) {

JTable\_Test1 sms=new JTable\_Test1();

}

//构造函数

public JTable\_Test1(){

columnNames=new Vector<>();

//设置列名

columnNames.add("学号");

columnNames.add("名字");

columnNames.add("性别");

columnNames.add("年龄");

columnNames.add("籍贯");

columnNames.add("系别");

rowData=new Vector<>();

//rowData可以存放多行

Vector hang=new Vector<>();

hang.add("sp001");

hang.add("孙悟空");

hang.add("男");

hang.add("500");

hang.add("花果山");

hang.add("少林派");

//加入rowData

rowData.add(hang);

//初始化JTable

jt=new JTable(rowData,columnNames);

//初始化jsp JScrollPane

jsp=new JScrollPane(jt);

//把jsp放入到jframe

this.add(jsp);

this.setSize(400, 300);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

-------------------------------------------------------------------------------

JTable与数据库的使用[JTable\_test2.java]

/\*\*

\* JTable的使用，从sql server数据库中取数据

\*/

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTable;

public class JTable\_Test2 extends JFrame{

//定义组件

//rowData用来存放行数据、columnNames存放列名

Vector rowData,columnNames;

JTable jt=null;

JScrollPane jsp=null;

//定义操作数据库需要的组件

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

public static void main(String[] args) {

new JTable\_Test2();

}

public JTable\_Test2(){

columnNames=new Vector<>();

//设置列名

columnNames.add("学号");

columnNames.add("名字");

columnNames.add("性别");

columnNames.add("年龄");

columnNames.add("籍贯");

columnNames.add("系别");

rowData=new Vector<>();

//rowData可以存放多行

try {

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;");

if(!ct.isClosed()){

System.out.println("数据库连接成功");

}else{

System.out.println("数据库连接失败");

}

ps=ct.prepareStatement("select \* from stu");

rs=ps.executeQuery();

while(rs.next()){

Vector hang=new Vector();

hang.add(rs.getString(1));

hang.add(rs.getString(2));

hang.add(rs.getString(3));

hang.add(rs.getInt(4));

hang.add(rs.getString(5));

hang.add(rs.getString(6));

//加入rowData

rowData.add(hang);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

//初始化JTable

jt=new JTable(rowData,columnNames);

//初始化jsp JScrollPane

jsp=new JScrollPane(jt);

//把jsp放入到jframe

this.add(jsp);

this.setSize(400, 300);

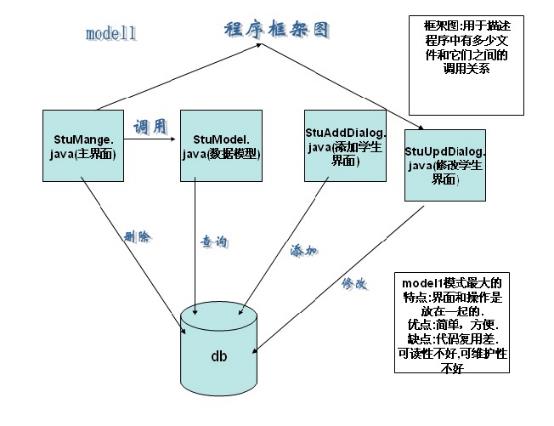
this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

}

**Mini学生管理系统**



[JTable\_Test3.java]源码

/\*\*

\* 完成一个mini版本的学生管理系统(MODEL1模式)

\* 1、查询任务

\* 2、添加功能

\* 3、使用的是sql server2012，数据库驱动与连接与视频教程中的sql server2000连接\* 有所不同

\*/

package com.student1;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.\*;

import javax.swing.table.AbstractTableModel;

public class JTable\_Test3 extends JFrame implements ActionListener{

//定义组件

JPanel jp1,jp2;

JLabel jl1;

JButton jb1,jb2,jb3,jb4;

JTable jt;

JScrollPane jsp;

JTextField jtf;

StuModel sm;

//定义操作数据库需要的组件

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

public static void main(String[] args) {

try {

// 将当前窗体外观设置为所在操作系统的外观

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

new JTable\_Test3();

}

//构造函数

public JTable\_Test3(){

jp1=new JPanel();

jtf=new JTextField(10);

jb1=new JButton("查询");

jb1.addActionListener(this);

jl1=new JLabel("请输入名字");

//把各个空间加入列

jp1.add(jl1);

jp1.add(jtf);

jp1.add(jb1);

jp2=new JPanel();

jb2=new JButton("添加");

jb2.addActionListener(this);

jb3=new JButton("修改");

jb3.addActionListener(this);

jb4=new JButton("删除");

jb4.addActionListener(this);

//把各个按钮加入到jp2中

jp2.add(jb2);

jp2.add(jb3);

jp2.add(jb4);

//创建一个数据模型对象

sm=new StuModel();

//初始化JTable

jt=new JTable(sm);

//初始化jsp JScrollPane

jsp=new JScrollPane(jt);

//把jsp放入到jframe

this.add(jsp);

this.add(jp1,"North");

this.add(jp2,"South");

this.setSize(400, 300);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jb1){

//因为把对表的数据封装到StuModel中，我们就可以比较简单的完成查询

String name=this.jtf.getText();

//写一个SQL语句

String sql="select \* from stu where stuName='"+name+"'";

//构建新的数据模型类，并更新

sm=new StuModel(sql);

//更新JTable

jt.setModel(sm);

}

//用户点击添加时

else if(e.getSource()==jb2){

StuAddDialog sa=new StuAddDialog(this, "添加学生", true);

//重新再获得新的数据模型

//构建新的数据模型类，并更新

sm=new StuModel();

//更新JTable

jt.setModel(sm);

}

//用户修改数据

else if(e.getSource()==jb3){

int rowNum=this.jt.getSelectedRow();

if(rowNum==-1){

//提示

JOptionPane.showMessageDialog(this, "请选择一行", "提示", JOptionPane.INFORMATION\_MESSAGE);

return;

}

//显示修改对话框

new StuUpdDialog(this,"修改学生信息",true,sm,rowNum);

//更新数据模型

sm=new StuModel();

//更新JTable

jt.setModel(sm);

}

//用户点击删除时，删除一条选中的数据

else if(e.getSource()==jb4){

//1、得到学生的ID号

//getSelectedRow会返回用户点中的行

//如果该用户一行都没有选择，就会返回-1

int rowNum=this.jt.getSelectedRow();

if(rowNum==-1){

//提示

JOptionPane.showMessageDialog(this, "请选择一行", "提示", JOptionPane.INFORMATION\_MESSAGE);

return;

}

//得到学生编号

String stuId=(String)sm.getValueAt(rowNum, 0);

//连接数据库，完成删除任务

try {

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;");

ps=ct.prepareStatement("delete from stu where stuid=?");

ps.setString(1, stuId);

ps.executeUpdate();

} catch (Exception e2) {

e2.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

//更新数据模型

sm=new StuModel();

//更新JTable

jt.setModel(sm);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuModel.java]源码

/\*\*

\* 这是一个stu表的模型

\* 可以把对student表的各种操作封装到该模型中

\*/

package com.student1;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.table.AbstractTableModel;

public class StuModel extends AbstractTableModel{

//rowData用来存放行数据、columnNames存放列名

Vector rowData,columnNames;

//定义操作数据库需要的组件

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

public void init(String sql){

if(sql==""||sql.equals(null)){

sql="select \* from stu";

}

//中间

columnNames=new Vector<>();

//设置列名

columnNames.add("学号");

columnNames.add("名字");

columnNames.add("性别");

columnNames.add("年龄");

columnNames.add("籍贯");

columnNames.add("系别");

rowData=new Vector<>();

//rowData可以存放多行

try {

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;");

ps=ct.prepareStatement(sql);

rs=ps.executeQuery();

while(rs.next()){

Vector hang=new Vector();

hang.add(rs.getString(1));

hang.add(rs.getString(2));

hang.add(rs.getString(3));

hang.add(rs.getInt(4));

hang.add(rs.getString(5));

hang.add(rs.getString(6));

//加入rowData

rowData.add(hang);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

//构造函数,用于初始我们的数据模型

public StuModel(String sql){

this.init(sql);

}

//构造函数

public StuModel(){

this.init("");

}

//得到共有多少列

public int getColumnCount() {

return this.columnNames.size();

}

public String getColumnName(int column) {

return (String)this.columnNames.get(column);

}

//得到共有多少行

public int getRowCount() {

return this.rowData.size();

}

//得到某行某列的数据

public Object getValueAt(int rowIndex, int columnIndex) {

return ((Vector)this.rowData.get(rowIndex)).get(columnIndex);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuAddDialog.java]源码

/\*\*

\* 添加数据

\*/

package com.student1;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StuAddDialog extends JDialog implements ActionListener{

//定义我需要的swing组件

JLabel jl1,jl2,jl3,jl4,jl5,jl6;

JButton jb1,jb2;

JTextField jtf1,jtf2,jtf3,jtf4,jtf5,jtf6;

JPanel jp1,jp2,jp3;

//owner它的父窗口;title窗口名;model指定是模态窗口，还是非模态

public StuAddDialog(Frame owner,String title,boolean modal){

super(owner,title,modal);//调用父类构造方法，达到模式对话框效果

jl1=new JLabel("学号");

jl2=new JLabel("名字");

jl3=new JLabel("性别");

jl4=new JLabel("年龄");

jl5=new JLabel("籍贯");

jl6=new JLabel("系别");

jtf1=new JTextField();

jtf2=new JTextField();

jtf3=new JTextField();

jtf4=new JTextField();

jtf5=new JTextField();

jtf6=new JTextField();

jb1=new JButton("添加");

jb2=new JButton("取消");

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//设置布局

jp1.setLayout(new GridLayout(6,1));

jp2.setLayout(new GridLayout(6,1));

//添加组件

jp1.add(jl1);

jp1.add(jl2);

jp1.add(jl3);

jp1.add(jl4);

jp1.add(jl5);

jp1.add(jl6);

jp2.add(jtf1);

jp2.add(jtf2);

jp2.add(jtf3);

jp2.add(jtf4);

jp2.add(jtf5);

jp2.add(jtf6);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1,BorderLayout.WEST);

this.add(jp2,BorderLayout.CENTER);

this.add(jp3,BorderLayout.SOUTH);

jb1.addActionListener(this);

jb2.addActionListener(this);

//展现

this.setSize(300, 250);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

//用户点击添加按钮后的响应动作

if(e.getSource()==jb1){

//连接数据库

Connection ct=null;

Statement stmt=null;

ResultSet rs=null;

PreparedStatement ps=null;

//连接数据库

try {

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;");

String strsql="insert into stu values(?,?,?,?,?,?)";

ps=ct.prepareStatement(strsql);

ps.setString(1,jtf1.getText());

ps.setString(2,jtf2.getText());

ps.setString(3,jtf3.getText());

ps.setInt(4,Integer.parseInt(jtf4.getText()));

ps.setString(5,jtf5.getText());

ps.setString(6,jtf6.getText());

ps.executeUpdate();

this.dispose();

} catch (Exception e2) {

e2.printStackTrace();

}finally{

try {

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

}

else if(e.getSource()==jb2){

this.dispose();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuUpdDialog.java]源码

/\*\*

\* 修改学生信息

\*/

package com.student1;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StuUpdDialog extends JDialog implements ActionListener{

//定义我需要的swing组件

JLabel jl1,jl2,jl3,jl4,jl5,jl6;

JButton jb1,jb2;

JTextField jtf1,jtf2,jtf3,jtf4,jtf5,jtf6;

JPanel jp1,jp2,jp3;

//owner它的父窗口;title窗口名;model指定是模态窗口，还是非模态

public StuUpdDialog(Frame owner,String title,boolean modal,StuModel sm,int rowNum){

super(owner,title,modal);//调用父类构造方法，达到模式对话框效果

jl1=new JLabel("学号");

jl2=new JLabel("名字");

jl3=new JLabel("性别");

jl4=new JLabel("年龄");

jl5=new JLabel("籍贯");

jl6=new JLabel("系别");

jtf1=new JTextField();

//初始化数据

jtf1.setText((String)sm.getValueAt(rowNum, 0));

//让jtf1不能修改

jtf1.setEditable(false);

jtf2=new JTextField();

jtf2.setText((String)sm.getValueAt(rowNum, 1));

jtf3=new JTextField();

jtf3.setText((String)sm.getValueAt(rowNum, 2));

jtf4=new JTextField();

jtf4.setText(sm.getValueAt(rowNum, 3).toString());

jtf5=new JTextField();

jtf5.setText((String)sm.getValueAt(rowNum, 4));

jtf6=new JTextField();

jtf6.setText((String)sm.getValueAt(rowNum, 5));

jb1=new JButton("修改");

jb2=new JButton("取消");

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//设置布局

jp1.setLayout(new GridLayout(6,1));

jp2.setLayout(new GridLayout(6,1));

//添加组件

jp1.add(jl1);

jp1.add(jl2);

jp1.add(jl3);

jp1.add(jl4);

jp1.add(jl5);

jp1.add(jl6);

jp2.add(jtf1);

jp2.add(jtf2);

jp2.add(jtf3);

jp2.add(jtf4);

jp2.add(jtf5);

jp2.add(jtf6);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1,BorderLayout.WEST);

this.add(jp2,BorderLayout.CENTER);

this.add(jp3,BorderLayout.SOUTH);

jb1.addActionListener(this);

jb2.addActionListener(this);

//展现

this.setSize(300, 250);

this.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

//用户点击添加按钮后的响应动作

if(e.getSource()==jb1){

//连接数据库

Connection ct=null;

Statement stmt=null;

ResultSet rs=null;

PreparedStatement ps=null;

//连接数据库

try {

//1、加载驱动

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

//2、得到连接

ct=DriverManager.getConnection("jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;");

String strsql="update stu set stuName=?,stuSex=?,stuAge=?,stuJg=?,stuDept=? where stuId=?";

ps=ct.prepareStatement(strsql);

ps.setString(1,jtf2.getText());

ps.setString(2,jtf3.getText());

ps.setInt(3,Integer.parseInt(jtf4.getText()));

ps.setString(4,jtf5.getText());

ps.setString(5,jtf6.getText());

ps.setString(6, jtf1.getText());

ps.executeUpdate();

this.dispose();

} catch (Exception e2) {

e2.printStackTrace();

}finally{

try {

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

}

else if(e.getSource()==jb2){

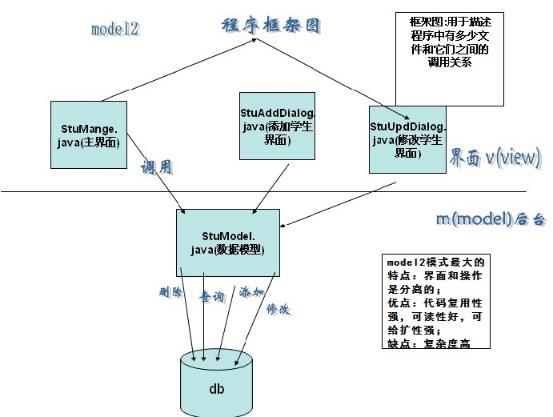
this.dispose();

}

}

}

**2、Mini学生管理系统**



[JTable\_Test3.java]源码

/\*\*

\* 完成一个mini版本的学生管理系统(MODEL2模式)

\* 1、查询任务

\* 2、添加功能

\*/

package com.student2;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.\*;

import javax.swing.table.AbstractTableModel;

public class JTable\_Test3 extends JFrame implements ActionListener{

//定义组件

JPanel jp1,jp2;

JLabel jl1;

JButton jb1,jb2,jb3,jb4;

JTable jt;

JScrollPane jsp;

JTextField jtf;

StuModel sm;

public static void main(String[] args) {

try {

// 将当前窗体外观设置为所在操作系统的外观

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

new JTable\_Test3();

}

//构造函数

public JTable\_Test3(){

jp1=new JPanel();

jtf=new JTextField(10);

jb1=new JButton("查询");

jb1.addActionListener(this);

jl1=new JLabel("请输入名字");

//把各个空间加入列

jp1.add(jl1);

jp1.add(jtf);

jp1.add(jb1);

jp2=new JPanel();

jb2=new JButton("添加");

jb2.addActionListener(this);

jb3=new JButton("修改");

jb3.addActionListener(this);

jb4=new JButton("删除");

jb4.addActionListener(this);

//把各个按钮加入到jp2中

jp2.add(jb2);

jp2.add(jb3);

jp2.add(jb4);

//创建一个数据模型对象

sm=new StuModel();

//初始化JTable

jt=new JTable(sm);

//初始化jsp JScrollPane

jsp=new JScrollPane(jt);

//把jsp放入到jframe

this.add(jsp);

this.add(jp1,"North");

this.add(jp2,"South");

this.setSize(400, 300);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jb1){

//因为把对表的数据封装到StuModel中，我们就可以比较简单的完成查询

String name=this.jtf.getText();

//写一个SQL语句

String sql="select \* from stu where stuName='"+name+"'";

//构建新的数据模型类，并更新

sm=new StuModel(sql);

//更新JTable

jt.setModel(sm);

}

//用户点击添加时

else if(e.getSource()==jb2){

StuAddDialog sa=new StuAddDialog(this, "添加学生", true);

//重新再获得新的数据模型

//构建新的数据模型类，并更新

sm=new StuModel();

//更新JTable

jt.setModel(sm);

}

//用户修改数据

else if(e.getSource()==jb3){

int rowNum=this.jt.getSelectedRow();

if(rowNum==-1){

//提示

JOptionPane.showMessageDialog(this, "请选择一行", "提示", JOptionPane.INFORMATION\_MESSAGE);

return;

}

//显示修改对话框

new StuUpdDialog(this,"修改学生信息",true,sm,rowNum);

//更新数据模型

sm=new StuModel();

//更新JTable

jt.setModel(sm);

}

//用户点击删除时，删除一条选中的数据

else if(e.getSource()==jb4){

//1、得到学生的ID号

//getSelectedRow会返回用户点中的行

//如果该用户一行都没有选择，就会返回-1

int rowNum=this.jt.getSelectedRow();

if(rowNum==-1){

//提示

JOptionPane.showMessageDialog(this, "请选择一行", "提示", JOptionPane.INFORMATION\_MESSAGE);

return;

}

//得到学生编号

String stuId=(String)sm.getValueAt(rowNum, 0);

//创建一个sql语句

String sql="delete from stu where stuid=?";

String []paras={stuId};

StuModel temp=new StuModel();

if(temp.updStu(sql, paras)){

JOptionPane.showMessageDialog(this,"删除数据成功","删除数据提示",JOptionPane.INFORMATION\_MESSAGE);

}else{

JOptionPane.showMessageDialog(this,"删除数据失败","删除数据提示",JOptionPane.ERROR\_MESSAGE);

}

//更新数据模型

sm=new StuModel();

//更新JTable

jt.setModel(sm);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuModel.java]源码

/\*\*

\* 这是一个stu表的模型

\* 可以把对student表的各种操作封装到该模型中

\*/

package com.student2;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.JOptionPane;

import javax.swing.table.AbstractTableModel;

public class StuModel extends AbstractTableModel{

//rowData用来存放行数据、columnNames存放列名

Vector rowData,columnNames;

//定义操作数据库需要的组件

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

String sqlDriver="com.microsoft.sqlserver.jdbc.SQLServerDriver";

String url="jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;";

//添加学生(增、删、改)

public boolean updStu(String sql,String []paras){

boolean b=true;

try {

//1、加载驱动

Class.forName(sqlDriver);

//2、得到连接

ct=DriverManager.getConnection(url);

//3、创建ps

ps=ct.prepareStatement(sql);

//给ps的问号赋值

for(int i=0;i<paras.length;i++){

ps.setString(i+1, paras[i]);

}

//4、执行操作

if(ps.executeUpdate()!=1){

b=false;

}

} catch (Exception e) {

b=false;

JOptionPane.showMessageDialog(null, "数据源错误或数据库用户名、密码错误", "数据库连接错误提示", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

return b;

}

public void init(String sql){

if(sql==""||sql.equals(null)){

sql="select \* from stu";

}

//中间

columnNames=new Vector<>();

//设置列名

columnNames.add("学号");

columnNames.add("名字");

columnNames.add("性别");

columnNames.add("年龄");

columnNames.add("籍贯");

columnNames.add("系别");

rowData=new Vector<>();

//rowData可以存放多行

try {

//1、加载驱动

Class.forName(sqlDriver);

//2、得到连接

ct=DriverManager.getConnection(url);

ps=ct.prepareStatement(sql);

rs=ps.executeQuery();

while(rs.next()){

Vector hang=new Vector();

hang.add(rs.getString(1));

hang.add(rs.getString(2));

hang.add(rs.getString(3));

hang.add(rs.getInt(4));

hang.add(rs.getString(5));

hang.add(rs.getString(6));

//加入rowData

rowData.add(hang);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

//构造函数,用于初始我们的数据模型

public StuModel(String sql){

this.init(sql);

}

//构造函数

public StuModel(){

this.init("");

}

//得到共有多少列

public int getColumnCount() {

return this.columnNames.size();

}

public String getColumnName(int column) {

return (String)this.columnNames.get(column);

}

//得到共有多少行

public int getRowCount() {

return this.rowData.size();

}

//得到某行某列的数据

public Object getValueAt(int rowIndex, int columnIndex) {

return ((Vector)this.rowData.get(rowIndex)).get(columnIndex);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuAddDialog.java]源码

/\*\*

\* 添加数据

\*/

package com.student2;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StuAddDialog extends JDialog implements ActionListener{

//定义我需要的swing组件

JLabel jl1,jl2,jl3,jl4,jl5,jl6;

JButton jb1,jb2;

JTextField jtf1,jtf2,jtf3,jtf4,jtf5,jtf6;

JPanel jp1,jp2,jp3;

//owner它的父窗口;title窗口名;model指定是模态窗口，还是非模态

public StuAddDialog(Frame owner,String title,boolean modal){

super(owner,title,modal);//调用父类构造方法，达到模式对话框效果

jl1=new JLabel("学号");

jl2=new JLabel("名字");

jl3=new JLabel("性别");

jl4=new JLabel("年龄");

jl5=new JLabel("籍贯");

jl6=new JLabel("系别");

jtf1=new JTextField();

jtf2=new JTextField();

jtf3=new JTextField();

jtf4=new JTextField();

jtf5=new JTextField();

jtf6=new JTextField();

jb1=new JButton("添加");

jb2=new JButton("取消");

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//设置布局

jp1.setLayout(new GridLayout(6,1));

jp2.setLayout(new GridLayout(6,1));

//添加组件

jp1.add(jl1);

jp1.add(jl2);

jp1.add(jl3);

jp1.add(jl4);

jp1.add(jl5);

jp1.add(jl6);

jp2.add(jtf1);

jp2.add(jtf2);

jp2.add(jtf3);

jp2.add(jtf4);

jp2.add(jtf5);

jp2.add(jtf6);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1,BorderLayout.WEST);

this.add(jp2,BorderLayout.CENTER);

this.add(jp3,BorderLayout.SOUTH);

jb1.addActionListener(this);

jb2.addActionListener(this);

//展现

this.setSize(300, 250);

this.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

//用户点击添加按钮后的响应动作

if(e.getSource()==jb1){

StuModel temp=new StuModel();

String sql="insert into stu values(?,?,?,?,?,?)";

String []paras={jtf1.getText(),jtf2.getText(),jtf3.getText(),jtf4.getText(),jtf5.getText(),jtf6.getText()};

if(!temp.updStu(sql, paras)){

JOptionPane.showMessageDialog(this, "添加数据失败", "添加数据提示", JOptionPane.ERROR\_MESSAGE);

}else{

JOptionPane.showMessageDialog(this,"添加数据成功","添加数据提示",JOptionPane.INFORMATION\_MESSAGE);

}

//关闭对话框

this.dispose();

}

else if(e.getSource()==jb2){

this.dispose();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuUpdDialog.java]源码

/\*\*

\* 修改学生信息

\*/

package com.student2;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StuUpdDialog extends JDialog implements ActionListener{

//定义我需要的swing组件

JLabel jl1,jl2,jl3,jl4,jl5,jl6;

JButton jb1,jb2;

JTextField jtf1,jtf2,jtf3,jtf4,jtf5,jtf6;

JPanel jp1,jp2,jp3;

//owner它的父窗口;title窗口名;model指定是模态窗口，还是非模态

public StuUpdDialog(Frame owner,String title,boolean modal,StuModel sm,int rowNum){

super(owner,title,modal);//调用父类构造方法，达到模式对话框效果

jl1=new JLabel("学号");

jl2=new JLabel("名字");

jl3=new JLabel("性别");

jl4=new JLabel("年龄");

jl5=new JLabel("籍贯");

jl6=new JLabel("系别");

jtf1=new JTextField();

//初始化数据

jtf1.setText((String)sm.getValueAt(rowNum, 0));

//让jtf1不能修改

jtf1.setEditable(false);

jtf2=new JTextField();

jtf2.setText((String)sm.getValueAt(rowNum, 1));

jtf3=new JTextField();

jtf3.setText((String)sm.getValueAt(rowNum, 2));

jtf4=new JTextField();

jtf4.setText(sm.getValueAt(rowNum, 3).toString());

jtf5=new JTextField();

jtf5.setText((String)sm.getValueAt(rowNum, 4));

jtf6=new JTextField();

jtf6.setText((String)sm.getValueAt(rowNum, 5));

jb1=new JButton("修改");

jb2=new JButton("取消");

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//设置布局

jp1.setLayout(new GridLayout(6,1));

jp2.setLayout(new GridLayout(6,1));

//添加组件

jp1.add(jl1);

jp1.add(jl2);

jp1.add(jl3);

jp1.add(jl4);

jp1.add(jl5);

jp1.add(jl6);

jp2.add(jtf1);

jp2.add(jtf2);

jp2.add(jtf3);

jp2.add(jtf4);

jp2.add(jtf5);

jp2.add(jtf6);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1,BorderLayout.WEST);

this.add(jp2,BorderLayout.CENTER);

this.add(jp3,BorderLayout.SOUTH);

jb1.addActionListener(this);

jb2.addActionListener(this);

//展现

this.setSize(300, 250);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

//用户点击添加按钮后的响应动作

if(e.getSource()==jb1){

//做一个sql语句

String sql="update stu set stuName=?,stuSex=?,stuAge=?,stuJg=?,stuDept=? where stuId=?";

String []paras={jtf2.getText(),jtf3.getText(),jtf4.getText(),jtf5.getText(),jtf6.getText(),jtf1.getText()};

StuModel temp=new StuModel();

if(temp.updStu(sql, paras)){

JOptionPane.showMessageDialog(this,"修改数据成功","修改数据提示",JOptionPane.INFORMATION\_MESSAGE);

}else{

JOptionPane.showMessageDialog(this,"修改数据失败","修改数据提示",JOptionPane.ERROR\_MESSAGE);

}

this.dispose();

}

else if(e.getSource()==jb2){

this.dispose();

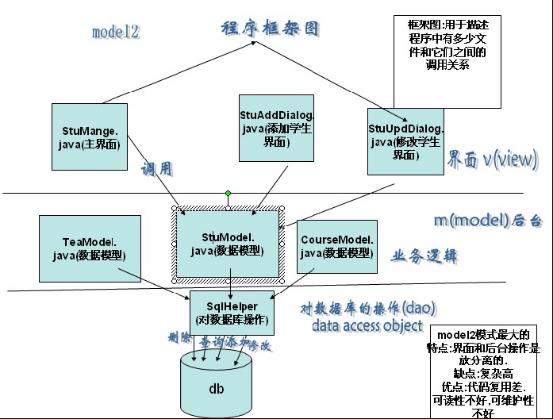
}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3、Mini学生管理系统**



[JTable\_Test3.java]源码

/\*\*

\* 完成一个mini版本的学生管理系统(MODEL2-2模式)

\* 1、查询任务

\* 2、添加功能

\*/

package com.student3;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.\*;

import javax.swing.table.AbstractTableModel;

public class JTable\_Test3 extends JFrame implements ActionListener{

//定义组件

JPanel jp1,jp2;

JLabel jl1;

JButton jb1,jb2,jb3,jb4;

JTable jt;

JScrollPane jsp;

JTextField jtf;

StuModel sm;

public static void main(String[] args) {

try {

// 将当前窗体外观设置为所在操作系统的外观

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

new JTable\_Test3();

}

//构造函数

public JTable\_Test3(){

jp1=new JPanel();

jtf=new JTextField(10);

jb1=new JButton("查询");

jb1.addActionListener(this);

jl1=new JLabel("请输入名字");

//把各个空间加入列

jp1.add(jl1);

jp1.add(jtf);

jp1.add(jb1);

jp2=new JPanel();

jb2=new JButton("添加");

jb2.addActionListener(this);

jb3=new JButton("修改");

jb3.addActionListener(this);

jb4=new JButton("删除");

jb4.addActionListener(this);

//把各个按钮加入到jp2中

jp2.add(jb2);

jp2.add(jb3);

jp2.add(jb4);

//创建一个数据模型对象

sm=new StuModel();

String []paras={"1"};

sm.queryStu("select \* from stu where 1=?", paras);

//初始化JTable

jt=new JTable(sm);

//初始化jsp JScrollPane

jsp=new JScrollPane(jt);

//把jsp放入到jframe

this.add(jsp);

this.add(jp1,"North");

this.add(jp2,"South");

this.setSize(400, 300);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jb1){

//因为把对表的数据封装到StuModel中，我们就可以比较简单的完成查询

String name=this.jtf.getText();

//写一个SQL语句

String sql="select \* from stu where stuName=?";

String []paras={name};

//构建新的数据模型类，并更新

sm=new StuModel();

sm.queryStu(sql, paras);

//更新JTable

jt.setModel(sm);

}

//用户点击添加时

else if(e.getSource()==jb2){

StuAddDialog sa=new StuAddDialog(this, "添加学生", true);

//重新再获得新的数据模型

//构建新的数据模型类，并更新

sm=new StuModel();

String []paras2={"1"};

sm.queryStu("select \* from stu where 1=?", paras2);

//更新JTable

jt.setModel(sm);

}

//用户修改数据

else if(e.getSource()==jb3){

int rowNum=this.jt.getSelectedRow();

if(rowNum==-1){

//提示

JOptionPane.showMessageDialog(this, "请选择一行", "提示", JOptionPane.INFORMATION\_MESSAGE);

return;

}

//显示修改对话框

new StuUpdDialog(this,"修改学生信息",true,sm,rowNum);

//更新数据模型

sm=new StuModel();

String []paras2={"1"};

sm.queryStu("select \* from stu where 1=?", paras2);

//更新JTable

jt.setModel(sm);

}

//用户点击删除时，删除一条选中的数据

else if(e.getSource()==jb4){

//1、得到学生的ID号

//getSelectedRow会返回用户点中的行

//如果该用户一行都没有选择，就会返回-1

int rowNum=this.jt.getSelectedRow();

if(rowNum==-1){

//提示

JOptionPane.showMessageDialog(this, "请选择一行", "提示", JOptionPane.INFORMATION\_MESSAGE);

return;

}

//得到学生编号

String stuId=(String)sm.getValueAt(rowNum, 0);

//创建一个sql语句

String sql="delete from stu where stuid=?";

String []paras={stuId};

StuModel temp=new StuModel();

if(temp.updStu(sql, paras)){

JOptionPane.showMessageDialog(this,"删除数据成功","删除数据提示",JOptionPane.INFORMATION\_MESSAGE);

}else{

JOptionPane.showMessageDialog(this,"删除数据失败","删除数据提示",JOptionPane.ERROR\_MESSAGE);

}

//更新数据模型

sm=new StuModel();

String []paras2={"1"};

sm.queryStu("select \* from stu where 1=?", paras2);

//更新JTable

jt.setModel(sm);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuModel.java]源码

/\*\*

\* 这是一个stu表的模型

\* 可以把对student表的各种操作封装到该模型中

\*/

package com.student3;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.JOptionPane;

import javax.swing.table.AbstractTableModel;

public class StuModel extends AbstractTableModel{

//rowData用来存放行数据、columnNames存放列名

Vector rowData,columnNames;

//添加学生(增、删、改)

public boolean updStu(String sql,String []paras){

//创建SqlHelper(如果程序并发性不考虑,可以把SqlHelper做成static)

SqlHelper sqlHelper=new SqlHelper();

return sqlHelper.updExecute(sql, paras);

}

//查询的本质就是用来初始化

public void queryStu(String sql,String []paras){

SqlHelper sqlHelper=null;

//中间

columnNames=new Vector<>();

//设置列名

columnNames.add("学号");

columnNames.add("名字");

columnNames.add("性别");

columnNames.add("年龄");

columnNames.add("籍贯");

columnNames.add("系别");

rowData=new Vector<>();

//rowData可以存放多行

try {

sqlHelper=new SqlHelper();

ResultSet rs=sqlHelper.queryExectue(sql, paras);

while(rs.next()){

Vector hang=new Vector();

hang.add(rs.getString(1));

hang.add(rs.getString(2));

hang.add(rs.getString(3));

hang.add(rs.getInt(4));

hang.add(rs.getString(5));

hang.add(rs.getString(6));

//加入rowData

rowData.add(hang);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

sqlHelper.close();

}

}

//得到共有多少列

public int getColumnCount() {

return this.columnNames.size();

}

@Override

public String getColumnName(int column) {

return (String)this.columnNames.get(column);

}

//得到共有多少行

public int getRowCount() {

return this.rowData.size();

}

//得到某行某列的数据

public Object getValueAt(int rowIndex, int columnIndex) {

return ((Vector)this.rowData.get(rowIndex)).get(columnIndex);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[SqlHelper.java]源码

/\*\*

\* 这是一个对数据库进行操作的类(SqlHelper)

\*/

package com.student3;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.swing.JOptionPane;

public class SqlHelper {

//定义操作数据库需要的组件

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

String sqlDriver="com.microsoft.sqlserver.jdbc.SQLServerDriver";

String url="jdbc:sqlserver://127.0.0.1:1433;databaseName=stussystem;user=sa;password=sa;";

public SqlHelper(){

try {

//1、加载驱动

Class.forName(sqlDriver);

//2、得到连接

ct=DriverManager.getConnection(url);

} catch (Exception e) {

e.printStackTrace();

}

}

//关闭数据库资源

public void close(){

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

//写一个不需要注入的方法(由于数据量少，所以写了一个这个方法。一般都带有条件的注入)

public ResultSet queryExectue(String sql){

try {

//3、创建ps

ps=ct.prepareStatement(sql);

rs=ps.executeQuery();

} catch (Exception e) {

e.printStackTrace();

}finally{

//关闭资源？？？

}

return rs;

}

//对数据库的查询操作

public ResultSet queryExectue(String sql,String []paras){

try {

//3、创建ps

ps=ct.prepareStatement(sql);

//给ps的问号赋值

for(int i=0;i<paras.length;i++){

ps.setString(i+1, paras[i]);

}

rs=ps.executeQuery();

} catch (Exception e) {

e.printStackTrace();

}finally{

//关闭资源？？？

}

return rs;

}

//把对数据库的增、删、改合在一起

public boolean updExecute(String sql,String []paras){

boolean b=true;

try {

//3、创建ps

ps=ct.prepareStatement(sql);

//给ps的问号赋值

for(int i=0;i<paras.length;i++){

ps.setString(i+1, paras[i]);

}

//4、执行操作

if(ps.executeUpdate()!=1){

b=false;

}

} catch (Exception e) {

b=false;

JOptionPane.showMessageDialog(null, "数据源错误或数据库用户名、密码错误", "数据库连接错误提示", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}finally{

this.close();

}

return b;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuAddDialog.java]源码

package com.student3;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StuAddDialog extends JDialog implements ActionListener{

//定义我需要的swing组件

JLabel jl1,jl2,jl3,jl4,jl5,jl6;

JButton jb1,jb2;

JTextField jtf1,jtf2,jtf3,jtf4,jtf5,jtf6;

JPanel jp1,jp2,jp3;

//owner它的父窗口;title窗口名;model指定是模态窗口，还是非模态

public StuAddDialog(Frame owner,String title,boolean modal){

super(owner,title,modal);//调用父类构造方法，达到模式对话框效果

jl1=new JLabel("学号");

jl2=new JLabel("名字");

jl3=new JLabel("性别");

jl4=new JLabel("年龄");

jl5=new JLabel("籍贯");

jl6=new JLabel("系别");

jtf1=new JTextField();

jtf2=new JTextField();

jtf3=new JTextField();

jtf4=new JTextField();

jtf5=new JTextField();

jtf6=new JTextField();

jb1=new JButton("添加");

jb2=new JButton("取消");

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//设置布局

jp1.setLayout(new GridLayout(6,1));

jp2.setLayout(new GridLayout(6,1));

//添加组件

jp1.add(jl1);

jp1.add(jl2);

jp1.add(jl3);

jp1.add(jl4);

jp1.add(jl5);

jp1.add(jl6);

jp2.add(jtf1);

jp2.add(jtf2);

jp2.add(jtf3);

jp2.add(jtf4);

jp2.add(jtf5);

jp2.add(jtf6);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1,BorderLayout.WEST);

this.add(jp2,BorderLayout.CENTER);

this.add(jp3,BorderLayout.SOUTH);

jb1.addActionListener(this);

jb2.addActionListener(this);

//展现

this.setSize(300, 250);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

//用户点击添加按钮后的响应动作

if(e.getSource()==jb1){

StuModel temp=new StuModel();

String sql="insert into stu values(?,?,?,?,?,?)";

String []paras={jtf1.getText(),jtf2.getText(),jtf3.getText(),jtf4.getText(),jtf5.getText(),jtf6.getText()};

if(!temp.updStu(sql, paras)){

JOptionPane.showMessageDialog(this, "添加数据失败", "添加数据提示", JOptionPane.ERROR\_MESSAGE);

}else{

JOptionPane.showMessageDialog(this,"添加数据成功","添加数据提示",JOptionPane.INFORMATION\_MESSAGE);

}

//关闭对话框

this.dispose();

}

else if(e.getSource()==jb2){

this.dispose();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StuUpdDialog.java]源码

/\*\*

\* 修改学生信息

\*/

package com.student3;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StuUpdDialog extends JDialog implements ActionListener{

//定义我需要的swing组件

JLabel jl1,jl2,jl3,jl4,jl5,jl6;

JButton jb1,jb2;

JTextField jtf1,jtf2,jtf3,jtf4,jtf5,jtf6;

JPanel jp1,jp2,jp3;

//owner它的父窗口;title窗口名;model指定是模态窗口，还是非模态

public StuUpdDialog(Frame owner,String title,boolean modal,StuModel sm,int rowNum){

super(owner,title,modal);//调用父类构造方法，达到模式对话框效果

jl1=new JLabel("学号");

jl2=new JLabel("名字");

jl3=new JLabel("性别");

jl4=new JLabel("年龄");

jl5=new JLabel("籍贯");

jl6=new JLabel("系别");

jtf1=new JTextField();

//初始化数据

jtf1.setText((String)sm.getValueAt(rowNum, 0));

//让jtf1不能修改

jtf1.setEditable(false);

jtf2=new JTextField();

jtf2.setText((String)sm.getValueAt(rowNum, 1));

jtf3=new JTextField();

jtf3.setText((String)sm.getValueAt(rowNum, 2));

jtf4=new JTextField();

jtf4.setText(sm.getValueAt(rowNum, 3).toString());

jtf5=new JTextField();

jtf5.setText((String)sm.getValueAt(rowNum, 4));

jtf6=new JTextField();

jtf6.setText((String)sm.getValueAt(rowNum, 5));

jb1=new JButton("修改");

jb2=new JButton("取消");

jp1=new JPanel();

jp2=new JPanel();

jp3=new JPanel();

//设置布局

jp1.setLayout(new GridLayout(6,1));

jp2.setLayout(new GridLayout(6,1));

//添加组件

jp1.add(jl1);

jp1.add(jl2);

jp1.add(jl3);

jp1.add(jl4);

jp1.add(jl5);

jp1.add(jl6);

jp2.add(jtf1);

jp2.add(jtf2);

jp2.add(jtf3);

jp2.add(jtf4);

jp2.add(jtf5);

jp2.add(jtf6);

jp3.add(jb1);

jp3.add(jb2);

this.add(jp1,BorderLayout.WEST);

this.add(jp2,BorderLayout.CENTER);

this.add(jp3,BorderLayout.SOUTH);

jb1.addActionListener(this);

jb2.addActionListener(this);

//展现

this.setSize(300, 250);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

//用户点击添加按钮后的响应动作

if(e.getSource()==jb1){

//做一个sql语句

String sql="update stu set stuName=?,stuSex=?,stuAge=?,stuJg=?,stuDept=? where stuId=?";

String []paras={jtf2.getText(),jtf3.getText(),jtf4.getText(),jtf5.getText(),jtf6.getText(),jtf1.getText()};

StuModel temp=new StuModel();

if(temp.updStu(sql, paras)){

JOptionPane.showMessageDialog(this,"修改数据成功","修改数据提示",JOptionPane.INFORMATION\_MESSAGE);

}else{

JOptionPane.showMessageDialog(this,"修改数据失败","修改数据提示",JOptionPane.ERROR\_MESSAGE);

}

this.dispose();

}

else if(e.getSource()==jb2){

this.dispose();

}

}

}

**满汉楼餐饮系统设计文档**

目 录

[第 1 章 软件设计概要 1](#_Toc249238483)

[1.1 设计目的 1](#_Toc249238484)

[1.2 特别说明 1](#_Toc249238485)

[1.3 性能需求 1](#_Toc249238486)

[1.4 环境需求 1](#_Toc249238487)

[第 2 章 界面设计 2](#_Toc249238488)

[2.1 登录启动界面 2](#_Toc249238489)

[2.2 登录界面 2](#_Toc249238490)

[2.3 主界面 3](#_Toc249238491)

[2.4 人事管理界面 4](#_Toc249238492)

[2.5 登录管理界面 4](#_Toc249238493)

[2.6 菜谱管理界面 5](#_Toc249238494)

[2.7 报表统计界面 6](#_Toc249238495)

[2.8 成本和库房管理界面 7](#_Toc249238496)

[2.9 动画帮助界面 8](#_Toc249238497)

[2.10 收款界面 9](#_Toc249238498)

[第 3 章 系统模块分析 11](#_Toc249238499)

[3.1 画图面板类 11](#_Toc249238500)

[3.2 数据库连接类 11](#_Toc249238501)

[3.3 音乐控制类 11](#_Toc249238502)

[第 4 章 数据库设计 12](#_Toc249238503)

[4.1 本系统拥有如下20个数据表 12](#_Toc249238504)

[4.2 数据表的连接关系 12](#_Toc249238505)

[第 5 章 功能模块设计 13](#_Toc249238506)

[5.1 管理界面 13](#_Toc249238507)

[5.1.1 人事管理 13](#_Toc249238508)

[5.1.2 登录管理 13](#_Toc249238509)

[5.1.3 菜谱价格 13](#_Toc249238510)

[5.1.4 报表统计 14](#_Toc249238511)

[5.1.5 成本控制 14](#_Toc249238512)

[5.1.6 系统设置 14](#_Toc249238513)

[5.1.7 系统帮助 14](#_Toc249238514)

[5.2 收款界面 14](#_Toc249238515)

[5.2.1 预定位置 14](#_Toc249238516)

[5.2.2 退订位置 14](#_Toc249238517)

[5.2.3 点菜系统 14](#_Toc249238518)

[5.2.4 结帐服务 14](#_Toc249238519)

[5.2.5 客户服务 14](#_Toc249238520)

[5.2.6 结帐服务 14](#_Toc249238521)

[第 6 章 具体模块设计 16](#_Toc249238522)

[6.1 管理界面主界面 16](#_Toc249238523)

[6.1.1 模块功能 16](#_Toc249238524)

[6.1.2 访问模块说明 16](#_Toc249238525)

[6.1.3 主界面功能说明 16](#_Toc249238526)

[6.2 人事管理 16](#_Toc249238527)

[6.3 收款界面主界面 19](#_Toc249238528)

[第 7 章 人性化设计分析 22](#_Toc249238529)

[第 8 章 总结 23](#_Toc249238530)

[第 9 章 备注 24](#_Toc249238531)

# 软件设计概要

* 1. 设计目的

在餐饮业中主要有4大类信息：人事管理、销售统计、成本管理和库房管理。

人工处理这些信息不仅工作量繁重、复杂，而且容易产生错误和混乱。采用计算机进行信息的有效管理不但能实现管理的规范化，而且还可以避免错误，提高效率，本系统即针对餐饮业的信息管理而设计制作的。

* 1. 特别说明
  2. 性能需求

本系统容量小，运行速度快，界面美观，人性化，操作方便，易于使用；经理及主管可拥有所有的权限（能同时操作管理和收款界面），收款员只能操作收款界面，其他人员则不能进入。

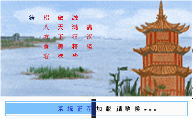
* 1. 环境需求

硬件环境：任何平台，256内存

软件环境： Windows 2000 + SQL Server2000 + JDK 1.5

开发环境：JAVA + SQL Server2000

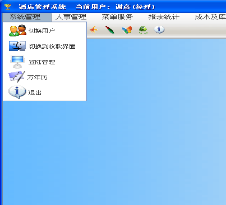
1. 界面设计
   1. 登录启动界面



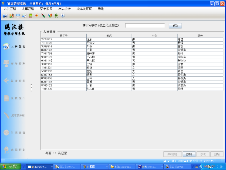
* 1. 登录界面



* 1. 主界面



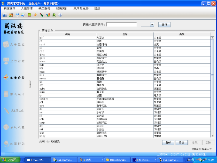
* 1. 人事管理界面



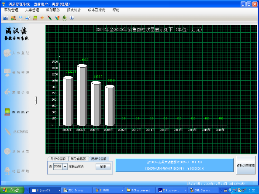
* 1. 登录管理界面



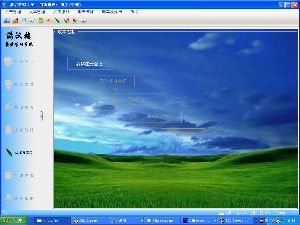
* 1. 菜谱管理界面



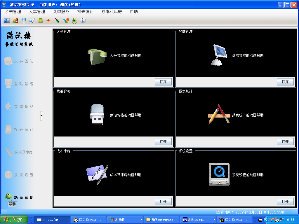
* 1. 报表统计界面



* 1. 成本和库房管理界面



* 1. 动画帮助界面



* 1. 收款界面



1. 系统模块分析

为了编程的过程中，程序结构更加的简洁明了，本软件系统编制了以下几个系统模块：

* 1. 画图面板类

作用：在各种面板加载背景和图片时调用，避免重复定义和类中过多的冗余代码

* 1. 数据库连接类

作用：在各个类进行数据库连接时调用，避免重复定义和类中过多的冗余代码

* 1. 音乐控制类

在各个组件加载音效和加载背景音乐时调用，避免重复定义和类中过多的冗余代码

1. 数据库设计
   1. 本系统拥有如下21个数据表

|  |  |
| --- | --- |
| 表名 | 介绍 |
| 01临时库存 | 存放本次盘库的临时信息 |
| 02人事资料 | 用于存放公司员工的详细信息 |
| 03价格 | 用于存放每种菜品的价格 |
| 04出入库单 | 处入库单明细 |
| 05单号流水 | 用于临时存放帐单的信息 |
| 06客户折扣 | 用于存放客户的累计消费及折扣信息 |
| 07帐单流水 | 用于存放帐单的相关信息 |
| 08客户资料 | 用于存放客户的相关信息，如姓名，联系电话 |
| 09成本名称 | 每项成本的名称及相关信息 |
| 10成本登记 | 每项成本的初始库存及单价，现在库存及单价 |
| 11桌子状态 | 用于存放各个桌子的相关信息 |
| 12点菜 | 用于存放所点菜的详细信息 |
| 13用餐 | 用于存放相关用餐信息 |
| 14登录 | 用于存放可用此软件员工的姓名及密码 |
| 15盘点明细 | 盘点明细帐 |
| 16盘点登记 | 盘点次数的登记，和盘点人 |
| 17菜谱 | 用于存放各种菜及饮品的相关信息 |
| 18销售量统计 | 用于统计每天的各菜品的销售量情况 |
| 19销售额统计 | 用于统计每天的销售额情况 |
| 20预定 | 用于保存预定位置的相关信息 |
| 21初始状态 | 存放成本录入的状态 |

* 1. 数据表的连接关系

1. 功能模块设计

闪 屏

登 录

收款界面

操作管理界面

预定

结帐统计

客户服务

结帐服务

预定位置

退定位置

点菜用餐



系统帮助

系统设置

成本控制

报表统计

菜谱价格

登录管理

人事管理

* 1. 管理界面
     1. 人事管理

新管理员或者经理或主管可以在这里进行注册，以获得注册帐号并获得软件的管理权限。

* + 1. 登录管理

已注册的用户可以在这里修改自己的密码，并核查已注册人员。

* + 1. 菜谱价格

菜谱价格面板用于添加或者修改各种菜品并为其制定价格

* + 1. 报表统计

报表统计面板主要用于显示每日或者每月或每年的销售额柱状图、曲线图，方便统计人员的直观查阅。

* + 1. 成本控制

成本控制面板用于各个阶段销售额及成本的计算，统计和归总。

* + 1. 系统设置

系统设置用于对系统的各种音乐进行选择或者设置音效。

* + 1. 系统帮助

主要进行各种帮助文件的调用和选择

* 1. 收款界面
     1. 预定位置

用于完成位置的预定操作

* + 1. 退订位置

用于完成位置的退定

* + 1. 点菜系统

用于完成点菜的相关操作，可以采用下拉式菜单，也可直接输入菜名或编号

* + 1. 结帐服务

用于完成结帐的相关操作

* + 1. 客户服务

用于添加客户及查找客户信息等操作

* + 1. 结帐服务

用语完成结帐统计及查找某个帐单的相关信息

1. 具体模块设计
   1. 管理界面主界面
      1. 模块功能

调用各个模块函数，并加载各个模块的功能面板，显示数据库表格数据及销售额柱状图、曲线图。作为用户的使用桌面。

* + 1. 访问模块说明

主要包括7大功能模块，分别是：

‘人事资料’

‘员工登陆’

‘菜谱价格’

‘报表统计’

‘成本库房’

‘帮 助’

‘系统设置’

* + 1. 主界面功能说明

点击主界面左功能按钮缩进纽后,左功能按钮区缩进,便于用户放大右区域的图标进行浏览查询。缩进后按钮变为功能面板弹出钮，点击后功能面板在左侧弹出。

主界面布局结构：一共有4个Jpanel，一个分屏面板，分屏面板加载了左侧的功能面板和右边的卡片式布局面板，然后下方框架CON加载了状态栏面板。

* 1. 人事管理

**模块功能**：对员工的信息进行添加、修改、删除、查询、显示详细信息的操作。

访问数据表说明：表名‘人事资料’

字段名及用途：

‘员工编号’存放员工的编号。（主键）

‘姓名’存放员工姓名。

‘照片’存放指定图片的路径。

‘性别’存放员工的性别。

‘住址’存放员工的家庭住址。

‘出生日期’存放员工的出生日期。

‘身份证号’存放员工的身份证号。

‘学历’存放员工的学历。

‘职位’存放员工现担任的职位。

‘婚否’存放员工的婚姻状况。

‘联系电话1’存放员工的坐机电话。

‘联系电话2’存放员工的手机号码。

‘邮箱’存放员工的电子邮箱。

‘注册时间’存放此条记录写入的时间。

‘备注’存放此员工的一些特殊说明。

**人事管理界面功能说明**：

**添加功能**：点击人事管理界面的‘添加’按钮，弹出‘新员工登记’对话框，填写此对话框的内容来说明员工信息，包括：姓名、性别、出生日期、身份证号、家庭住址、学历、婚否、联系电话、手机、职位、邮箱、备注、照片等信息。

**新员工登记**’对话框布局说明：采用空布局，共有15个JLabel、4个JRadioButton，5个JComboBox，6个JTextField，1个JTextArea，1个JScrollPane，3个JButton，1个JPanel组成。

‘新员工登记’对话框填写要求及功能说明：姓名、身份证号和职位是必添项，其他为选添项，姓名、地址、邮箱、备注为文本域且均小于50个字符，身份证号为文本域只能填写数字且在16－18位之间，性别是单选按钮默认是‘男’，婚否是单选按钮默认是‘未婚’，学历是下拉菜单默认是‘无’，职位是下拉菜单默认值是空白，手机和联系电话均为文本域，添加照片是由一个‘添加照片’按钮和一个Panel组成，点击‘添加照片’按钮可弹出文件选择对话框，在对话框中的筛选器只能选择图片文件（JPG、BMP、GIF），选中指定文件按打开后，可以把选中的图片显示到Panel中去。取消按钮，可不保存当前填写的信息而返回人事管理界面。添加按钮，可以把当前的信息存入数据库中的‘人事信息’表中，把选订的图片放到Image文件夹下，如果没有选择文件则按选择的性别分配默认图片文件，并根据身份证号码的后六位和此员工的职位信息自动产生一个编号为员工编号连同当前点击添加按钮的时间为注册时间一并显示在一个添出式对话框中，让用户确定，确定后将此信息一并写入数据库。

**修改功能**：在选中人事管理界面中的表格的一行时，‘修改’按钮会变为可选状态，如点击可修改这一行记录的信息，弹出‘修改员工记录’对话框，填写此对话框的内容来修改所选中的已经添加过的员工基本信息。如选中一行是再点击人事管理界面的其他无按钮位置，‘修改’按钮会变为不可选状态。

‘修改员工记录’对话框布局说明：采用空布局，共有15个JLabel、4个JRadioButton，5个JComboBox，6个JTextField，1个JTextArea，1个JScrollPane，3个JButton，1个JPanel组成。

‘修改员工记录’对话框填写要求及功能说明：从数据库中调出此员工注册的信息分别填写到‘修改员工记录’对话框的相应位置，从数据库中调出此员工的照片路径再从此位置把照片画到Panel中去，修改相应的要修改信息。其中：姓名、身份证号和职位是必添项，其他为选添项，姓名、地址、邮箱、备注为文本域且均小于50个字符，身份证号为文本域只能填写数字且在16－18位之间，性别是单选按钮默认是‘男’，婚否是单选按钮默认是‘未婚’，学历是下拉菜单默认是‘无’，职位是下拉菜单默认值是空白，手机和联系电话均为文本域，添加照片是由一个‘添加照片’按钮和一个Panel组成，点击‘修改照片’按钮可弹出文件选择对话框，在对话框中的筛选器只能选择图片文件（JPG、BMP、GIF），选中指定文件按打开后，可以把选中的图片显示到Panel中去。取消按钮，可不保存当前填写的信息而返回人事管理界面。修改按钮，可以把当前的信息替换数据库中‘人事信息’表中的当前要修改行，把修改的照片放到Image文件夹下，并把以前旧的照片删除，如果没有修改照片则无照片操作。

**删除功能**：点击人事管理界面表格中要删除的记录，‘删除’按钮会变为可选状态，点击‘删除’按钮，弹出‘删除员工’对话框，询问是否删除当前所选记录，选择‘是’则删除数据表中的相应记录，人事管理界面表格中的相应记录，和image文件夹下的照片文件，如是默认图片则不做删除照片操作。

‘删除员工’对话框布局说明：JOPtionPane中的Message选择框。

**查询功能**：在人事管理的上面有查询对话框，可以输入员工的姓名，员工号，和职位，均可以查昂到相应的信息。

**详细信息功能**：双击或点中一行按详细信息，都可以打开人事资料界面可以查看此人的详细信息。

* 1. 收款界面主界面

**模块功能**：调用各个模块函数，可根据数据库的相关信息改变桌子的颜色，并且右边按钮的状态可根据左边桌子按钮的相关状态进行

**改变**。而且右边标签内容可根据左边所按下的桌子按钮的相关

信息，桌子用蓝色表示空，黄色表示预定，红色表示正在用餐

**访问模块说明**：

主要包括6大功能模块，分别是：

‘预定位置’

‘退定位置’

‘点菜服务’

‘结帐服务’

‘客户服务’

‘结帐服务’

* 1. 预定位置(BookTable)

**模块功能**：可以输入预定时间，预定人数，预定人姓名，联系电话，

预定位置，接到电话时间是自动取系统当前时间添入 ，

如按下确定，则将相关信息加入数据库预定表中，并将桌子状态表中此桌子的状态改为预定，然后将所

预定的桌子颜色变为黄色，按取消则放弃。注意此界面支持

键盘回车Enter键及方向键

**界面结构：**由六个标签及六个文本框和两个按钮构成

2 退定位置(ExitTable)

**模块功能**：以画图的形式显示预定时间，预定人数，预定人姓名，联系电话， 预定位置，接到电话时间 ， 如按下确定，则将此桌子数据库预定表中相关信息删除，并将桌子状态表中此桌子的状态改为空，并将此退定桌子颜色变为蓝色，按取消则放弃。注意此界面支持键盘回车Enter键及方向键

3点菜服务(BookDish)

**模块功能**：完成点菜的相关操作，左边列表框显示菜的类别，其下面表格可根据列表框所选类别显示此类别的菜品，左边最下面两个列表框可选择服务员及厨师，双击左边表格中某行或选中后单击添加，可将相关信息加入到右边表格中，也可在右边表格 的文本框中输入菜的名称或编码，按确定键直接将相关菜的信息加入到右边表格中。如果选中右边表格某行后，可按修改按钮修改所选菜品的数量，按删除将删除该行。按确定则将所选的菜的相关信息加入到单号流水表中，用餐表中，并将桌子状态表中此桌子的状态改为用餐中，并将此桌子颜色变为红色，按取消则放弃。

**界面结构：**由类别列表框，服务员列表框，厨师列表框，两个表格，输入名称的文本框，输入数量的文本眶，确定按钮，删除按钮，修改按钮，添加按钮，退出按钮组成

4结帐服务(endBill)

**模块功能**：完成结帐的相关操作，左边表格显示所点的菜，右边总计中显示所点菜的总额，可在贵宾卡号中输入卡号，如果正确，连接数据库得到相应折扣信息，并计算出实际应交费用，结帐方式可选择三种，注意挂帐只能是经理，需要经理输入密码验证，如选择现金，则需在收款中输入所收款项，按结帐按钮，计算出应找的钱数，然后可选择是否打印帐单，如选择打印，则此界面消失，打印帐单，更改数据库的相关信息，并将此桌子颜色变为蓝色，按取消则放弃。

**界面结构：**由类别列表框，服务员列表框，厨师列表框，两个表格，输入名称的文本框，输入数量的文本眶，确定按钮，删除按钮，修改按钮，添加按钮，退出按钮组成

5 客户服务(cardSever)

**模块功能**：因为只有主管及经理才能进行添加客户，修改客户资料，删除客户资料，因此需主管验证

**模块功能**：客户服务主界面，可在查找眶中输入客户姓名，卡号，或者选择下拉列表眶中的数值按查找按钮进行查询操作。选则表格中的一行，启用修改，删除，详细资料按钮，按下相应按钮调用相应子界面。

**模块功能**：客户添加子界面，可输入姓名，性别，家庭住址，邮政编码，联系电话，注册日期及卡号自动生成

**模块功能**：双击某行或选中后单击详细按钮，以画图的形式显示客户的详细信息。

**6 结帐服务(BillSever)**

**模块功能**：主要完成日结帐，月结帐，年结帐，及相关查找，在查找框中可输入帐单号，或日期，查找相关帐单，将其信息加入到表格中。按下日结帐，将连接数据库以画图的形式显示当天的销售额总计，并显示每钟结帐方式的销售额及每种类别采的畅销菜的前五名，月结帐，年结帐同理。

1. 人性化设计分析
2. 总结
3. 备注

满汉楼视频同步源代码说明：

项目：满汉楼餐饮管理系统

开发平台：MyEclipse 2014

数据库：sql server 2012

数据库连接：使用的是微软提供的JDBC4.0驱动

软件项目名：MyRestaurant

Java包名：com.mhl.view(界面包)/com.mhl.model(模型包)/com.mhl.db(数据库包)/com.mhl.tools(工具包)

特别说明：由于视频教程中只是将满汉楼项目的界面搭出来和写了对数据库的连接和删除功能外，其它功能都没有完善。满汉楼视频教程的主要教学思想是以model2模式来开发项目。

Model2模式即：界面层/业务逻辑层/数据库操作层/数据库

com.mhl.view(界面包)的各个源代码

[Index.java]闪屏效果源代码

/\*\*

\* 用户登录的闪屏效果

\*/

package com.mhl.view;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.Toolkit;

import javax.swing.JPanel;

import javax.swing.JWindow;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

public class Index extends JWindow implements Runnable {

paint p;

public static void main(String[] args) {

try {

// 将当前窗体外观设置为所在操作系统的外观

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

Index index = new Index();

Thread t = new Thread(index);

t.start();

}

public Index() {

// 创建P

p = new paint();

this.add(p);

this.setSize(400, 250);

// 确定Jwindow初始位置

// int width = Toolkit.getDefaultToolkit().getScreenSize().width;

// int height = Toolkit.getDefaultToolkit().getScreenSize().height;

// this.setLocation(width / 2 - 200, height / 2 - 150);

this.setLocationRelativeTo(null);

this.setVisible(true);

}

public void run() {

while (true) {

// 根据计算等待闪屏效果结束后，再跳转到登陆界面

try {

Thread.sleep(30 \* 500);

} catch (InterruptedException e) {

e.printStackTrace();

}

//关闭闪屏

this.dispose();

// 跳转到登陆界面

new UserLogin();

break;

}

}

}

// 开发一个闪屏类

class paint extends JPanel implements Runnable{// 可以自动刷新Panel

Thread t;

int x = 10;

int i = 0, j = 40, u = 10;

String gg[] = { "系", "统", "正", "在", "加", "载", "请", "稍", "侯" };

int k = 0, tt = 0;

String shi[] = { "满", "汉", "楼", "融", "满", "汉", "精", "华", "做", "天", "下",

"美", "味", "招", "八", "方", "食", "客", "结", "四", "海", "良", "朋", "满", "汉", "楼", "程", "序", "演", "示" };

Font f = new Font("隶书", Font.PLAIN, 18);

boolean ifok = true;

int width = 180;

int height = 0;

int dian = 0;

paint() {

t = new Thread(this);

t.start();

}

public void run() {

while (true) {

if (x <= 380);

repaint();

try {

Thread.sleep(70);

i++;// i=0

j = j - 6;// j=40

u = u + 10;// u=10

if (tt == 3)

width = width - 20; // tt=0,width=180

if (i == 4) {

tt++;

if (ifok && x > 120 + k \* 20)

k++;// k=0

if (k >= gg.length - 1)

ifok = false;

x = x + 10;

i = 0;

j = 40;

u = 10;

dian++;

if (dian > 3)

dian = 0;

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public void paintComponent(Graphics g) {

// super.paintComponent(g);

Image image;

image = Toolkit.getDefaultToolkit().getImage("image/index/index.jpg");// 获得背景图片

g.drawImage(image, 0, 0, this.getWidth(), 200, this);

int r = (int) (Math.random() \* 255);

int b = (int) (Math.random() \* 255);

int y = (int) (Math.random() \* 255);

g.setColor(new Color(253, 250, 250));// White

g.fillRect(x, 210, 380 - x, 30);

g.setColor(new Color(253, 250, 250));// white

if (i > 1)

g.fillRect(x, 235 - (j + 20) / 2, 10, j + 10);// 领头的长方块儿

if (j > 25)

g.setColor(new Color(r, b, y));

else

g.setColor(new Color(123, 194, 252));// light blue

g.fillRect(x, 235 - (j + 20) / 2, 10, j);// 后面跟着的短方块儿,10代表闪动条的宽度，j代表闪动条的高度

g.setColor(new Color(123, 194, 252));// light blue

g.drawRect(10, 210, 380, 30);

if (x < 120) {

for (int l = 0; l < gg.length; l++) {

g.setColor(new Color(0, 0, 0));

g.drawString(gg[l], 120 + l \* 20, 230);

}

for (int l = 0; l < dian; l++) {

g.setColor(new Color(0, 0, 0));

g.drawString("\*", 300 + l \* 10, 235);

}

g.drawString("\*", 300 + dian \* 10, 235);

} else {

// 设置写字的颜色和位置

g.setColor(new Color(23, 23, 230));

g.drawString(gg[k], 120 + k \* 20, 230);

for (int l = k + 1; l < gg.length; l++) {

g.setColor(new Color(0, 0, 0));

g.drawString(gg[l], 120 + l \* 20, 230);

}

if (x > 300 + dian \* 10)

g.setColor(new Color(23, 23, 230));

for (int l = 0; l < dian; l++) {

g.drawString("\*", 300 + l \* 10, 235);

}

g.drawString("\*", 300 + dian \* 10, 235);

}

// ------------逐字写诗

if (tt < 3) {

for (int rr = 0; rr <= tt; rr++) {

g.setColor(new Color(250, 0, 0));

g.drawString(shi[rr], 180, 60 + rr \* 20);

}

g.drawString(shi[tt], 180, 60 + tt \* 20);

}

if (tt >= 3 && tt < 8) {

g.setColor(new Color(230, 0, 0));

for (int rr = 0; rr < 3; rr++)

g.drawString(shi[rr], 180, 60 + rr \* 20);

g.setColor(new Color(r, b, y));

if (tt < 8)

for (int rr = 3; rr <= tt; rr++)

g.drawString(shi[rr], 150, rr \* 20);

if (tt >= 7)

for (int rr = 3; rr < 8; rr++)

g.drawString(shi[rr], 150, rr \* 20);

}

if (tt >= 8 && tt < 13) {

g.setColor(new Color(230, 0, 0));

for (int rr = 0; rr < 3; rr++)

g.drawString(shi[rr], 180, 60 + rr \* 20);

for (int rr = 3; rr <= 7; rr++)

g.drawString(shi[rr], 150, rr \* 20);

g.setColor(new Color(r, b, y));

if (tt < 13)

for (int rr = 8; rr <= tt; rr++)

g.drawString(shi[rr], 120, rr \* 20 - 100);

if (tt >= 13)

for (int rr = 8; rr < 13; rr++)

g.drawString(shi[rr], 120, rr \* 20 - 100);

}

if (tt >= 13 && tt < 18) {

g.setColor(new Color(230, 0, 0));

for (int rr = 0; rr < 3; rr++)

g.drawString(shi[rr], 180, 60 + rr \* 20);

for (int rr = 3; rr <= 7; rr++)

g.drawString(shi[rr], 150, rr \* 20);

for (int rr = 8; rr < 13; rr++)

g.drawString(shi[rr], 120, rr \* 20 - 100);

g.setColor(new Color(r, b, y));

if (tt < 18)

for (int rr = 13; rr <= tt; rr++)

g.drawString(shi[rr], 90, rr \* 20 - 200);

if (tt >= 18)

for (int rr = 13; rr < 13; rr++)

g.drawString(shi[rr], 90, rr \* 20 - 200);

}

if (tt >= 18 && tt < 23) {

g.setColor(new Color(230, 0, 0));

for (int rr = 0; rr < 3; rr++)

g.drawString(shi[rr], 180, 60 + rr \* 20);

for (int rr = 3; rr <= 7; rr++)

g.drawString(shi[rr], 150, rr \* 20);

for (int rr = 8; rr < 13; rr++)

g.drawString(shi[rr], 120, rr \* 20 - 100);

for (int rr = 13; rr < 18; rr++)

g.drawString(shi[rr], 90, rr \* 20 - 200);

g.setColor(new Color(r, b, y));

if (tt < 23)

for (int rr = 18; rr <= tt; rr++)

g.drawString(shi[rr], 60, rr \* 20 - 300);

if (tt >= 23)

for (int rr = 18; rr < 23; rr++)

g.drawString(shi[rr], 60, rr \* 20 - 300);

}

if (tt >= 23 && tt < 30) {

g.setColor(new Color(230, 0, 0));

for (int rr = 0; rr < 3; rr++)

g.drawString(shi[rr], 180, 60 + rr \* 20);

for (int rr = 3; rr <= 7; rr++)

g.drawString(shi[rr], 150, rr \* 20);

for (int rr = 8; rr < 13; rr++)

g.drawString(shi[rr], 120, rr \* 20 - 100);

for (int rr = 13; rr < 18; rr++)

g.drawString(shi[rr], 90, rr \* 20 - 200);

for (int rr = 18; rr < 23; rr++)

g.drawString(shi[rr], 60, rr \* 20 - 300);

g.setColor(new Color(r, b, y));

if (tt < 30)

for (int rr = 23; rr <= tt; rr++)

g.drawString(shi[rr], 30, rr \* 20 - 400);

if (tt >= 30) {

tt = 0;

for (int rr = 18; rr < 23; rr++)

g.drawString(shi[rr], 30, rr \* 20 - 300);

}

}

if (tt > 30) {

g.setColor(new Color(230, 0, 0));

for (int rr = 0; rr < 3; rr++)

g.drawString(shi[rr], 180, 60 + rr \* 20);

for (int rr = 3; rr <= 7; rr++)

g.drawString(shi[rr], 150, rr \* 20);

for (int rr = 8; rr < 13; rr++)

g.drawString(shi[rr], 120, rr \* 20 - 100);

for (int rr = 13; rr < 18; rr++)

g.drawString(shi[rr], 90, rr \* 20 - 200);

for (int rr = 18; rr < 23; rr++)

g.drawString(shi[rr], 60, rr \* 20 - 300);

for (int rr = 23; rr < 30; rr++)

g.drawString(shi[rr], 30, rr \* 20 - 400);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[UserLogin.java]登录界面源代码

/\*\*

\* 用户登录界面

\*/

package com.mhl.view;

import com.mhl.model.UserModel;

import com.mhl.tools.\*;

import java.awt.Color;

import java.awt.Container;

import java.awt.Font;

import java.awt.GradientPaint;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class UserLogin extends JDialog implements ActionListener{

//定义 需要的组件

JLabel jl1,jl2,jl3;

JTextField jName;

JPasswordField jPasswd;

JButton jConf,jCancel;

public static void main(String []args){

UserLogin ul=new UserLogin();

}

public UserLogin(){

//把一个组件放入到JFrame或者JDialog可以直接放

//也可以先放到Container

Container ct=this.getContentPane();

//空布局

this.setLayout(null);

//创建各个组件

jl1=new JLabel("请输入用户名:");

jl1.setBounds(60, 190, 150, 30);

jl1.setFont(MyTools.f1);

ct.add(jl1);

jName=new JTextField(20);

jName.setFocusable(true);

jName.setBounds(180, 190, 120, 30);

jName.setBorder(BorderFactory.createLoweredBevelBorder());//下凹

ct.add(jName);

jl2=new JLabel("(或员工号)");

jl2.setForeground(Color.red);//前景色

jl2.setFont(MyTools.f2);

jl2.setBounds(100, 210, 100, 30);

ct.add(jl2);

jl3=new JLabel("请输入密 码:");

jl3.setBounds(60, 240, 150, 30);

jl3.setFont(MyTools.f1);

ct.add(jl3);

jPasswd=new JPasswordField(20);

jPasswd.setBounds(180, 240, 120, 30);

jPasswd.setBorder(BorderFactory.createLoweredBevelBorder());

ct.add(jPasswd);

jConf=new JButton("确定");

jConf.setFont(MyTools.f1);

jConf.setBounds(110, 300, 70, 30);

ct.add(jConf);

jConf.addActionListener(this);

jCancel=new JButton("取消");

jCancel.setFont(MyTools.f1);

jCancel.setBounds(210, 300, 70, 30);

ct.add(jCancel);

jCancel.addActionListener(this);

//创建一个BackImage对象

BackImage bi=new BackImage();

//把位置确定

bi.setBounds(0, 0, 360, 360);

//不使用上下框

ct.add(bi);

// this.add(bi);

this.setUndecorated(true);

this.setSize(360, 360);

this.setLocationRelativeTo(null);

this.setVisible(true);

}

//内部类

class BackImage extends JPanel{

Image im;

public BackImage(){

try {

im=ImageIO.read(new File("image/index/login.gif"));

} catch (IOException e) {

e.printStackTrace();

}

}

public void paintComponent(Graphics g){

g.drawImage(im, 0, 0, 360, 360, this);

}

}

//响应用户登录的请求

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jConf){

//取出员工号，密码

String u=this.jName.getText().trim();

String p=new String(this.jPasswd.getPassword());

UserModel um=new UserModel();

String res=um.checkUser(u, p);

if(res.equals("SystemAd")||res.equals("经理")||res.equals("主管")){

new Windows1();

this.dispose();

}else{

JOptionPane.showMessageDialog(this, "您无权限登录或用户名密码错误", "登录提示", JOptionPane.ERROR\_MESSAGE);

return;

}

}

else if(e.getSource()==jCancel){

this.dispose();

System.exit(0);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[Windows1.java]主界面源代码

/\*\*

\* 这是系统管理员、经理、主管可以进入的管理操作界面

\* 完成界面的顺序，从上到下，从左到右

\*/

package com.mhl.view;

import com.mhl.tools.\*;

import java.awt.BorderLayout;

import java.awt.CardLayout;

import java.awt.Container;

import java.awt.Cursor;

import java.awt.GridLayout;

import java.awt.Image;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.File;

import java.io.IOException;

import java.util.Calendar;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JPanel;

import javax.swing.JSplitPane;

import javax.swing.JToolBar;

import javax.swing.Timer;

public class Windows1 extends JFrame implements ActionListener,MouseListener{

//定义需要的组件

Image titleIcon,timeBg;

JMenuBar jmb;

//一级菜单

JMenu jm1,jm2,jm3,jm4,jm5,jm6;

//二级菜单

JMenuItem jmm1,jmm2,jmm3,jmm4,jmm5,jmm6,jmm7,jmm8,jmm9,jmm10,jmm11,jmm12;

//二级菜单图标

ImageIcon jmm1\_Icon,jmm2\_Icon,jmm3\_Icon,jmm4\_Icon,jmm5\_Icon,jmm6\_Icon,jmm7\_Icon,jmm8\_Icon,jmm9\_Icon,jmm10\_Icon,jmm11\_Icon,jmm12\_Icon;

//工具栏

JToolBar jtb;

JButton jb1,jb2,jb3,jb4,jb5,jb6,jb7,jb8,jb9,jb10;

//定义需要的五个JPanel

JPanel p1,p2,p3,p4,p5;

//显示当前时间的Label

JLabel timeNow;

JLabel p1\_lab1,p1\_lab2,p1\_lab3,p1\_lab4,p1\_lab5,p1\_lab6,p1\_lab7,p1\_lab8;

//给p2面板定义需要的JLabel

JLabel p2\_lab1,p2\_lab2;

//javax.swing包中的Timer可以定时的触发Action事件，我们可以利用它来完成一些事情

Timer t;

ImagePanel p1\_imgPanel;

JSplitPane jsp;

//卡片布局

CardLayout cardP2,cardP3;

public static void main(String[] args) {

new Windows1();

}

//初始化菜单

public void initMenu(){

//创建图标

jmm1\_Icon=new ImageIcon("image/Windows1/30/jb1.png");

jmm2\_Icon=new ImageIcon("image/Windows1/30/jb2.png");

jmm3\_Icon=new ImageIcon("image/Windows1/30/jb3.png");

jmm4\_Icon=new ImageIcon("image/Windows1/30/jb4.png");

jmm5\_Icon=new ImageIcon("image/Windows1/30/jb5.png");

jmm6\_Icon=new ImageIcon("image/Windows1/30/jb6.png");

jmm7\_Icon=new ImageIcon("image/Windows1/30/jb7.png");

jmm8\_Icon=new ImageIcon("image/Windows1/30/jb8.png");

jmm9\_Icon=new ImageIcon("image/Windows1/30/jb9.png");

jmm10\_Icon=new ImageIcon("image/Windows1/30/jb10.png");

jmm11\_Icon=new ImageIcon("image/Windows1/30/jb11.png");

jmm12\_Icon=new ImageIcon("image/Windows1/30/jb12.png");

//创建菜单栏

jmb=new JMenuBar();

//创建一级菜单

jm1=new JMenu("系统管理");

jm1.setFont(MyTools.f2);

//创建系统管理二级菜单

jmm1=new JMenuItem("切换用户", jmm1\_Icon);

jmm2=new JMenuItem("切换收款界面", jmm2\_Icon);

jmm3=new JMenuItem("登录管理", jmm3\_Icon);

jmm4=new JMenuItem("万年历", jmm4\_Icon);

jmm5=new JMenuItem("退出", jmm5\_Icon);

//将二级菜单加入到一级菜单

jm1.add(jmm1);

jm1.add(jmm2);

jm1.addSeparator();//分割符

jm1.add(jmm3);

jm1.addSeparator();

jm1.add(jmm4);

jm1.addSeparator();

jm1.add(jmm5);

jm2=new JMenu("人事管理");

jm2.setFont(MyTools.f2);

jmm6=new JMenuItem("人事登记", jmm6\_Icon);

jm2.add(jmm6);

jm3=new JMenu("菜单服务");

jm3.setFont(MyTools.f2);

jmm7=new JMenuItem("菜谱及价格录入", jmm7\_Icon);

jm3.add(jmm7);

jm4=new JMenu("报表统计");

jm4.setFont(MyTools.f2);

jmm8=new JMenuItem("报表统计", jmm8\_Icon);

jm4.add(jmm8);

jm5=new JMenu("成本及库房");

jm5.setFont(MyTools.f2);

jmm9=new JMenuItem("成本控制", jmm9\_Icon);

jm5.add(jmm9);

jm6=new JMenu("帮助");

jm6.setFont(MyTools.f2);

jmm10=new JMenuItem("动画帮助", jmm10\_Icon);

jmm11=new JMenuItem("文本帮助", jmm11\_Icon);

jmm12=new JMenuItem("关于", jmm12\_Icon);

jm6.add(jmm10);

jm6.add(jmm11);

jm6.add(jmm12);

//将一级菜单加入到菜单栏

jmb.add(jm1);

jmb.add(jm2);

jmb.add(jm3);

jmb.add(jm4);

jmb.add(jm5);

jmb.add(jm6);

//将jmb菜单栏加入到JFrame

this.setJMenuBar(jmb);

}

//初始化工具栏

public void initToolBar(){

//处理工具栏的组件

jtb=new JToolBar();

//设置工具栏JToolBar不浮动

jtb.setFloatable(false);

jb1=new JButton(new ImageIcon("image/Windows1/20/jb1.png"));

jb1.setToolTipText("切换用户");

jb2=new JButton(new ImageIcon("image/Windows1/20/jb2.png"));

jb2.setToolTipText("切换界面");

jb3=new JButton(new ImageIcon("image/Windows1/20/jb3.png"));

jb3.setToolTipText("登录管理");

jb4=new JButton(new ImageIcon("image/Windows1/20/jb4.png"));

jb4.setToolTipText("万年历");

jb6=new JButton(new ImageIcon("image/Windows1/20/jb6.png"));

jb6.setToolTipText("人事管理");

jb7=new JButton(new ImageIcon("image/Windows1/20/jb7.png"));

jb7.setToolTipText("菜谱及价格录入");

jb8=new JButton(new ImageIcon("image/Windows1/20/jb8.png"));

jb8.setToolTipText("报表统计");

jb9=new JButton(new ImageIcon("image/Windows1/20/jb9.png"));

jb9.setToolTipText("成本控制");

jb10=new JButton(new ImageIcon("image/Windows1/20/jb11.png"));

jb10.setToolTipText("帮助");

jb5=new JButton(new ImageIcon("image/Windows1/20/jb5.png"));

jb5.setToolTipText("退出系统");

//将按钮加入JToolBar中

jtb.add(jb1);

jtb.add(jb2);

jtb.add(jb3);

jtb.add(jb4);

jtb.add(jb6);

jtb.add(jb7);

jtb.add(jb8);

jtb.add(jb9);

jtb.add(jb10);

jtb.add(jb5);

}

//初始化中间的四个Panel

public void initAllPanels(){

//处理p1面板

p1=new JPanel(new BorderLayout());

Image p1\_bg=null;

try {

p1\_bg = ImageIO.read(new File("image/Windows1/jp1\_bg.jpg"));

} catch (IOException e1) {

e1.printStackTrace();

}

//定义手形光标

Cursor myCursor=new Cursor(Cursor.HAND\_CURSOR);

p1\_imgPanel=new ImagePanel(p1\_bg);

p1\_imgPanel.setLayout(new GridLayout(8,1));//网格布局

//p1的第一个Label

p1\_lab1=new JLabel(new ImageIcon("image/Windows1/p1\_mhl.jpg"));

p1\_imgPanel.add(p1\_lab1);

//加入第2个Label

p1\_lab2=new JLabel("人 事 管 理",new ImageIcon("image/Windows1/30/jb6.png"),0);

p1\_lab3=new JLabel("登 录 管 理",new ImageIcon("image/Windows1/30/jb3.png"),0);

p1\_lab4=new JLabel("菜 谱 价 格",new ImageIcon("image/Windows1/30/jb7.png"),0);

p1\_lab5=new JLabel("报 表 统 计",new ImageIcon("image/Windows1/30/jb8.png"),0);

p1\_lab6=new JLabel("成本及库房",new ImageIcon("image/Windows1/30/jb9.png"),0);

p1\_lab7=new JLabel("系 统 设 置",new ImageIcon("image/Windows1/30/sys.png"),0);

p1\_lab8=new JLabel("动 画 帮 助",new ImageIcon("image/Windows1/30/jb11.png"),0);

//初始化Label为不可用

p1\_lab2.setEnabled(false);

p1\_lab3.setEnabled(false);

p1\_lab4.setEnabled(false);

p1\_lab5.setEnabled(false);

p1\_lab6.setEnabled(false);

p1\_lab7.setEnabled(false);

p1\_lab8.setEnabled(false);

//鼠标移动到JLabel上变成手形提示

p1\_lab1.setCursor(myCursor);

p1\_lab2.setCursor(myCursor);

p1\_lab3.setCursor(myCursor);

p1\_lab4.setCursor(myCursor);

p1\_lab5.setCursor(myCursor);

p1\_lab6.setCursor(myCursor);

p1\_lab7.setCursor(myCursor);

p1\_lab8.setCursor(myCursor);

//注册鼠标事件监听

p1\_lab1.addMouseListener(this);

p1\_lab2.addMouseListener(this);

p1\_lab3.addMouseListener(this);

p1\_lab4.addMouseListener(this);

p1\_lab5.addMouseListener(this);

p1\_lab6.addMouseListener(this);

p1\_lab7.addMouseListener(this);

p1\_lab8.addMouseListener(this);

//将label加入到p1\_imgPanel中

p1\_imgPanel.add(p1\_lab2);

p1\_imgPanel.add(p1\_lab3);

p1\_imgPanel.add(p1\_lab4);

p1\_imgPanel.add(p1\_lab5);

p1\_imgPanel.add(p1\_lab6);

p1\_imgPanel.add(p1\_lab7);

p1\_imgPanel.add(p1\_lab8);

p1.add(p1\_imgPanel);

//处理p2,p3,p4面板

p4=new JPanel(new BorderLayout());//边界布局

cardP2=new CardLayout();

p2=new JPanel(cardP2);//卡片布局

p2\_lab1=new JLabel(new ImageIcon("image/Windows1/jp2\_left.jpg"));

p2\_lab2=new JLabel(new ImageIcon("image/Windows1/jp2\_right.jpg"));

//监听

p2\_lab1.addMouseListener(this);

p2\_lab2.addMouseListener(this);

//把p2\_lab1加入到p2中

p2.add(p2\_lab1,"0");

p2.add(p2\_lab2,"1");

//给P3创建卡片布局

cardP3=new CardLayout();

p3=new JPanel(cardP3);

//先给p3加入一个主界面的卡片(ImagePanel)

Image zhu\_image=null;

try {

zhu\_image = ImageIO.read(new File("image/Windows1/p3\_bj.jpg"));

} catch (IOException e1) {

e1.printStackTrace();

}

ImagePanel ip=new ImagePanel(zhu\_image);

p3.add(ip,"0");

//对p3做添加几个JLabel--测试，正式写要放Panel进去

//创建EmpInfo对象(JPanel)

EmpInfo eInfo=new EmpInfo();

p3.add(eInfo,"1");

//把显示报表的JPanel创建出来

Chart myChart=new Chart();

// p3.add(myChart,"4");

//把p2,p3加入到p4

p4.add(p2,"West");

p4.add(p3,"Center");

//做一个拆分窗口,分别存放p1和p4

jsp=new JSplitPane(JSplitPane.HORIZONTAL\_SPLIT,true,p1,p4);

//指定左边的面板占多大范围

jsp.setDividerLocation(120);

//把边界线设置为0

jsp.setDividerSize(0);

}

//构造函数

public Windows1(){

//创建组件

try {

titleIcon=ImageIO.read(new File("image/Windows1/jb.jpg"));

} catch (IOException e) {

e.printStackTrace();

}

//调用初始化菜单函数

this.initMenu();

//调用初始化工具栏函数

this.initToolBar();

//调用初始化中间的四个Panel函数

this.initAllPanels();

//处理p5面板

p5=new JPanel(new BorderLayout());

//创建Timer

t=new Timer(1000, this);//每隔1秒触发Action事件

//启动定时器

t.start();

timeNow=new JLabel("当前时间: "+Calendar.getInstance().getTime().toLocaleString()+" ");

timeNow.setFont(MyTools.f2);

try {

timeBg=ImageIO.read(new File("image/Windows1/time\_bg.jpg"));

} catch (IOException e) {

e.printStackTrace();

}

ImagePanel ip1=new ImagePanel(timeBg);

ip1.setLayout(new BorderLayout());

ip1.add(timeNow,"East");

p5.add(ip1);

//创建主容器

Container ct=this.getContentPane();

ct.add(jtb,"North");

ct.add(jsp,"Center");

ct.add(p5,"South");

//设置大小

int w=Toolkit.getDefaultToolkit().getScreenSize().width;

int h=Toolkit.getDefaultToolkit().getScreenSize().height;

this.setSize(w, h-30);

//关闭窗口时，退出系统

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//设置titel图标

this.setIconImage(titleIcon);

this.setTitle("满汉楼餐饮管理系统");

//窗口打开最大化

this.setExtendedState(JFrame.MAXIMIZED\_BOTH);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

this.timeNow.setText("当前时间: "+Calendar.getInstance().getTime().toLocaleString()+" ");

}

//鼠标点击事件

public void mouseClicked(MouseEvent e) {

//判断用户点击哪个操作JLabel

if(e.getSource()==p1\_lab1){

cardP3.show(p3,"0");

}else if(e.getSource()==p1\_lab2){

cardP3.show(p3,"1");

}else if(e.getSource()==p1\_lab3){

cardP3.show(p3,"2");

}else if(e.getSource()==p1\_lab4){

cardP3.show(p3,"3");

}else if(e.getSource()==p1\_lab5){

cardP3.show(p3,"4");

}else if(e.getSource()==p1\_lab6){

cardP3.show(p3,"5");

}else if(e.getSource()==p1\_lab7){

cardP3.show(p3,"6");

}else if(e.getSource()==p1\_lab8){

cardP3.show(p3,"7");

}

//p2左右伸缩功能

else if(e.getSource()==p2\_lab1){

//把显示各种操作的界面P1隐藏起来

cardP2.show(p2,"1");

jsp.setDividerLocation(0);

}else if(e.getSource()==p2\_lab2){

//把显示各种操作的界面P1显示出来

cardP2.show(p2,"0");

jsp.setDividerLocation(120);

}

}

//鼠标按下事件

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

//鼠标弹起事件

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

//鼠标移动到事件

public void mouseEntered(MouseEvent e) {

//如果用户选中了某个操作JLabel则高亮提示

if(e.getSource()==p1\_lab2){

p1\_lab2.setEnabled(true);

}else if(e.getSource()==p1\_lab3){

p1\_lab3.setEnabled(true);

}else if(e.getSource()==p1\_lab4){

p1\_lab4.setEnabled(true);

}else if(e.getSource()==p1\_lab5){

p1\_lab5.setEnabled(true);

}else if(e.getSource()==p1\_lab6){

p1\_lab6.setEnabled(true);

}else if(e.getSource()==p1\_lab7){

p1\_lab7.setEnabled(true);

}else if(e.getSource()==p1\_lab8){

p1\_lab8.setEnabled(true);

}

}

//鼠标离开事件

public void mouseExited(MouseEvent e) {

//如果用户鼠标离开选中了某个操作JLabel则高亮关闭提示

if(e.getSource()==p1\_lab2){

p1\_lab2.setEnabled(false);

}else if(e.getSource()==p1\_lab3){

p1\_lab3.setEnabled(false);

}else if(e.getSource()==p1\_lab4){

p1\_lab4.setEnabled(false);

}else if(e.getSource()==p1\_lab5){

p1\_lab5.setEnabled(false);

}else if(e.getSource()==p1\_lab6){

p1\_lab6.setEnabled(false);

}else if(e.getSource()==p1\_lab7){

p1\_lab7.setEnabled(false);

}else if(e.getSource()==p1\_lab8){

p1\_lab8.setEnabled(false);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[EmpInfo.java]人事资料界面源代码

/\*\*

\* 这是用于显示人事管理的界面

\*/

package com.mhl.view;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import com.mhl.model.EmpModel;

public class EmpInfo extends JPanel implements ActionListener{

//定义需要的各个组件

JPanel p1,p2,p3,p4,p5;

JLabel p1\_lab1,p3\_lab1;

JTextField p1\_jtf1;

JButton p1\_jb1,p4\_jb1,p4\_jb2,p4\_jb3,p4\_jb4;

//这个是用于显示人事信息的

JTable jtable;

JScrollPane jsp;

EmpModel em;

//构造函数

public EmpInfo(){

//创建需要的组件

//p1面板

p1=new JPanel(new FlowLayout(FlowLayout.CENTER));//流布局居中显示

p1\_lab1=new JLabel("请输入姓名(员工号或职位):");

p1\_jtf1=new JTextField(20);

p1\_jb1=new JButton("查询");

//把它们加入到p1

p1.add(p1\_lab1);

p1.add(p1\_jtf1);

p1.add(p1\_jb1);

//p2面板

em=new EmpModel();

String []paras={"1"};

em.query("select empid'员工号',empname'姓名',sex'性别',zhiwei'职位' from rszl where 1=?",paras);

jtable=new JTable(em);

jsp=new JScrollPane(jtable);

p2=new JPanel(new BorderLayout());//边界布局

p2.add(jsp);

//处理p3,p4,p5面板

p3=new JPanel(new FlowLayout(FlowLayout.LEFT));

p3\_lab1=new JLabel(" 共有"+jtable.getRowCount()+"条记录");

p3.add(p3\_lab1);

p4=new JPanel(new FlowLayout(FlowLayout.RIGHT));

p4\_jb1=new JButton("详细信息");

p4\_jb2=new JButton("添加");

p4\_jb3=new JButton("修改");

p4\_jb4=new JButton("删除");

//监听事件

p4\_jb1.addActionListener(this);

p4\_jb2.addActionListener(this);

p4\_jb3.addActionListener(this);

p4\_jb4.addActionListener(this);

//将按钮添加到p4面板

p4.add(p4\_jb1);

p4.add(p4\_jb2);

p4.add(p4\_jb3);

p4.add(p4\_jb4);

p5=new JPanel(new BorderLayout());

p5.add(p3,"West");

p5.add(p4,"East");

//把主面板设为BorderLayout

this.setLayout(new BorderLayout());

//把p1加入到JPanel中

this.add(p1,"North");

this.add(p2,"Center");

this.add(p5,"South");

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

//如果用户要删除某个员工

if(e.getSource()==p4\_jb1){//详细信息

}else if(e.getSource()==p4\_jb2){//添加

}else if(e.getSource()==p4\_jb3){//修改

}else if(e.getSource()==p4\_jb4){//删除

int selRowNum=jtable.getSelectedRow();//选中的行

if(selRowNum!=1){

JOptionPane.showMessageDialog(null, "请选中要删除的数据","删除提示",JOptionPane.INFORMATION\_MESSAGE);

}

String empNo=(String)em.getValueAt(selRowNum, 0);

if(em.delEmpById(empNo)){

JOptionPane.showMessageDialog(null, "删除数据成功","删除提示",JOptionPane.INFORMATION\_MESSAGE);

}else{

JOptionPane.showMessageDialog(null, "删除数据失败","删除提示",JOptionPane.ERROR\_MESSAGE);

}

//更新显示一下

em=new EmpModel();

String []paras={"1"};

em.query("select empid'员工号',empname'姓名',sex'性别',zhiwei'职位' from rszl where 1=?", paras);

jtable.setModel(em);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.mhl.medel(业务逻辑包)

[UserModel.java]用户登录业务逻辑源代码

/\*\*

\* 这是用户表数据模型，用它完成对用户的各种操作

\* [这里主要是编写项目需要的业务操作]

\*/

package com.mhl.model;

import java.sql.ResultSet;

import com.mhl.db.SqlHelper;

public class UserModel {

/\*\*

\*

\* @param userId 用户编号

\* @param passwd 用户密码

\* @return 该用户的职位,如果该用户不存在，则返回""

\*/

public String checkUser(String userId,String passwd){

String zhiwei="";

SqlHelper sp=null;

try {

//组织sql语句和参数列表

String sql="select r.zhiwei from login l,rszl r where l.empid=r.empid and l.empid=? and l.passwd=?";

String paras[]={userId,passwd};

sp=new SqlHelper();

ResultSet rs=sp.query(sql, paras);

if(rs.next()){

//取出职位zhiwei

zhiwei=rs.getString(1);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

sp.close();

}

return zhiwei;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[EmpModel.java]人事管理业务逻辑源代码

/\*\*

\* 这是人事的数据模型类，主要完成对人事表的各种操作(CRUD增、删、改、查)

\*/

package com.mhl.model;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.util.Vector;

import javax.swing.AbstractListModel;

import javax.swing.table.AbstractTableModel;

import com.mhl.db.SqlHelper;

public class EmpModel extends AbstractTableModel{

Vector<String> colums;

Vector<Vector> rows;

SqlHelper sh;

//提供一个通过员工号来删除员工的函数

public boolean delEmpById(String empId){

boolean b=true;

String sql="delete from rszl where empId=?";

String []paras={empId};

sh=new SqlHelper();

try {

b=sh.exeUpdate(sql, paras);

} catch (Exception e) {

e.printStackTrace();

}finally{

sh.close();

}

return b;

}

//写一个方法用于查询需要显示的人事信息

public void query(String sql,String []paras){

//初始化列

this.colums=new Vector<String>();

this.rows=new Vector<Vector>();

//创建SqlHelper对象

sh=new SqlHelper();

ResultSet rs=sh.query(sql, paras);

//把rs的结果放入到rows

try {

//从rs对象中可以得到一个ResultSetMetaData

//rsmt可以得到结果有多少列，而且可以知道每列的名字

ResultSetMetaData rsmt=rs.getMetaData();

for(int i=0;i<rsmt.getColumnCount();i++){

colums.add(rsmt.getColumnName(i+1));

}

//把rs的结果放到rows中

while(rs.next()){

Vector<String> temp=new Vector<String>();

for(int i=0;i<rsmt.getColumnCount();i++){

temp.add(rs.getString(i+1));

}

rows.add(temp);

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

sh.close();

}

}

@Override

public String getColumnName(int column) {

return this.colums.get(column).toString();

}

//返回多少行

public int getRowCount() {

return this.rows.size();

}

//返回多少列

public int getColumnCount() {

return this.colums.size();

}

//返回行列的值

public Object getValueAt(int rowIndex, int columnIndex) {

return ((Vector)rows.get(rowIndex)).get(columnIndex);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.mhl.db(数据库操作包)

[SqlHelper.java]数据库操作源代码

/\*\*

\* 对数据库操作的类

\* 功能：对数据库的操作就是(CRUD增删改查)

\* 调用存储过程[oracle]

\*/

package com.mhl.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.swing.JOptionPane;

public class SqlHelper {

//定义需要的对象

PreparedStatement ps=null;

ResultSet rs=null;

Connection ct=null;

String dirverName="com.microsoft.sqlserver.jdbc.SQLServerDriver";

String url="jdbc:sqlserver://127.0.0.1:1433;databaseName=Restaurant;user=sa;password=sa;";

//构造函数，初始化ct数据库连接

public SqlHelper(){

try {

//加载驱动

Class.forName(dirverName);

//得到连接

ct=DriverManager.getConnection(url);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "数据库连接失败!!"+"\r\n请检查数据源配置及数据库用户名密码是否正确!!", "数据库连接提示", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}

//把对数据库的增、删、改写一个函数

public boolean exeUpdate(String sql,String []paras){

boolean b=true;

try {

ps=ct.prepareStatement(sql);

//对sql的参数赋值

for(int i=0;i<paras.length;i++){

ps.setString(i+1, paras[i]);

}

//执行

ps.executeUpdate();

} catch (Exception e) {

b=false;

e.printStackTrace();

}

return b;

}

//查询结果集

public ResultSet query(String sql,String []paras){

try {

ps=ct.prepareStatement(sql);

//对sql的参数赋值

for(int i=0;i<paras.length;i++){

ps.setString(i+1, paras[i]);

}

//执行

rs=ps.executeQuery();

} catch (Exception e) {

e.printStackTrace();

}

return rs;

}

//关闭资源的方法

public void close(){

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.mhl.tools(工具包)

[MyTools.java]字体工具包源代码

package com.mhl.tools;

import java.awt.Font;

public class MyTools {

public static Font f1=new Font("宋体", Font.PLAIN, 16);

public static Font f2=new Font("宋体", Font.PLAIN, 14);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ImagePanel.java]动态加载图片做背景的Jpanel

/\*\*

\* 这是一个可以动态加载一个图片做背景的JPanel

\*/

package com.mhl.tools;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.Toolkit;

import javax.swing.JPanel;

public class ImagePanel extends JPanel{

Image im;

//构造函数去指定该Panel的大小

public ImagePanel(Image im){

this.im=im;

//希望它的大小是自适应的

int w=Toolkit.getDefaultToolkit().getScreenSize().width;

int h=Toolkit.getDefaultToolkit().getScreenSize().height;

this.setSize(w, h);

}

//画出背景

public void paintComponent(Graphics g){

//清屏

super.paintComponent(g);

g.drawImage(im, 0,0,this.getWidth(),this.getHeight(),this);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

建立数据库SQL代码分析器源代码

--创建Restaurant数据库

CREATE DATABASE Restaurant

--创建人事管理表

create table rszl(

empid nvarchar(30) primary key,

empname nvarchar(30) not null,

sex NCHAR(1) check (sex in('男','女')) default '男' ,

zhiwei nchar(8) not null,

)

go

--创建登录表

create table login(

empid nvarchar(30) foreign key references rszl(empid),

passwd nvarchar(30)

)

go

INSERT INTO rszl VALUES('admin','admin','男','SystemAd')

INSERT INTO rszl VALUES('xz001','admin1','男','经理')

INSERT INTO login VALUES('admin','admin')

go

SELECT \* FROM login

SELECT \* FROM rszl

uml可以对系统进行设计

uml的几个重要的图

用例图--主要用于说明系统的功能

类图--主要说明该系统有多少个类，及类的结构

时序图--用于说明用户每个用例，它的内部流程(按照时间来看),一般说，针对每个用例都要有一个时序图对应

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# java网络编程

qq聊天项目介绍

涉及到java各个方面的技术

1、java面向对象编程；2、界面编程；3、网络编程；4、多线程；5、文件I/O操作；6、数据库操作

# java网络编程基础知识

学习java网络编程你必需要了解的几个概念：

## 1、协议(TCP/IP)

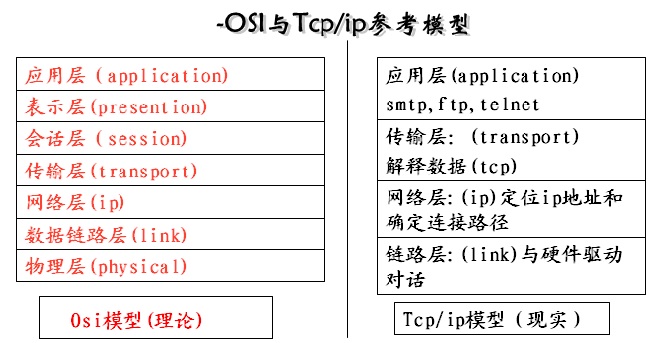
TCP/IP(Transmission Control Protocol/Internet Protocol)的简写，中文译名为传输控制协议/因特网互联协议，又叫网络通讯协议，这个协议是Internet最基本的协议、Internet国际互联网络的基础，简单地说，就是由网络层的IP协议和传输层的TCP协议组成的。

TCP/IP协议的由来

在阿帕网(ARPR)产生运作之初，通过接口信号处理机实现互联的电脑并不多，大部分电脑相互之间不兼容，在一台电脑上完成的工作，很难拿到另一台电脑上去用，想让硬件和软件都不一样的电脑联网，也有很多困难。当时美国的状况是，陆军用的电脑是DEC系列产品，海军用的电脑是Honeywell中标机器，空军用的是IBM公司中标的电脑，每一个军种的电脑在各自的系里都运行良好，但却有一个大弊病：不能共享资源。

互联网之父--瑟夫(Vinton G.Cerf)

1997年，为了褒奖对因特网发展作出突出贡献的科学家，并对TCP/IP协议作出充分肯定，美国授予为因特网发明和定义TCP/IP协议的文顿·瑟夫和卡恩“国家技术金奖”。这无疑使人们认识到TCP/IP协议的重要性。



tcp/ip基础--ip地址与包的路由传递

ip地址

概述：每个internet上的主机和路由器都有一个ip地址，它包括网络号和主机号，所有ip地址都是32位的，ip地址按照国际标准的划分为a,b,c,d,e五种类型。

## 端口(port)--概念

在网络技术中，端口(Port)有好几种意思。集线器、交换机、路由器的端口指的是连接其他网络设备的接口，如RJ-45端口、Serial端口等。我们这里所指的端口不是指物理意义上的端口，而是特指TCP/IP协议中的端口，是逻辑意义上的端口。

如果把IP地址比作一间房子，端口就是出入这间房子的门。真正的房子只有几个门，但是一个IP地址的端口可以有65536(即：256\*256)个之多!端口是通过端口号来标记的，端口号只有整数，范围是从0到65535(256\*256-1)。

可以使用netstat –an查看本机被使用的端口

端口(port)--分类

有65536个端口 0号是保留端口

1-1024是固定端口

又叫有名端口，即被某些程序固定使用，一般程序员不使用。

22：SSH远程登录协议 23：telnet使用 21：ftp使用

25：smtp服务使用 80：iis使用 7：echo服务

1025-65535是动态端口

这些端口，程序员可以使用

端口(port)--注意事项

1、在计算机(尤其是做服务器)要尽可能的少开端口；

2、一个端口只能被一个程序监听；

3、如果使用netstat -an可以查看本机有哪些端口在监听

4、可以使用netstat -anb来查看监听端口的pid，在结合任务管理器关闭不案例的端口。

## url--概念

统一资源定位符(URL，英语Uniform Resource Locator的缩写)也被称为网页地址，是因特网上标准的资源的地址。它最初是由蒂姆·伯纳斯-李发明用来作为万维网的地址的。现在它已经被万维网联盟编制为因特网标准RFC1738了。

Internet上的每一个网页都具有一个唯一的名称标识，通常称之为URL地址，这种地址可以是本地磁盘，也可以是局域网上的某一台计算机，更多的是Internet上的站点，简单地说，URL就是Web地址，俗称“网址”，是用于完整地描述Internet上网页和其他资源的地址的一种标识方法。

url--组成

http://www.sina.com:8080/index.html

1、协议；2、ip地址(32位)；3、端口号(16位)0-65535；4、资源名称。

单工、半双工和全双工

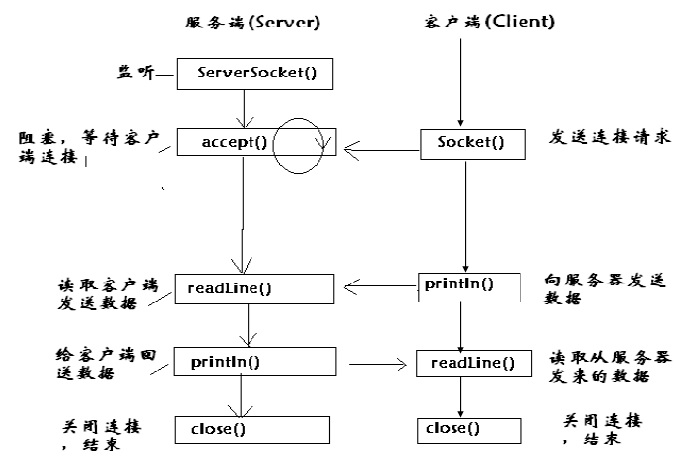
如果甲可以向乙发送数据，但是乙不能向甲发送数据，这样的通信就是单工通信(Simplex Communication)。

单工数据传输只支持数据在一个方向上传输，就和传呼机一样。

半双工数据传输允许数据在两个方向上传输，但是，在某一时刻，只允许数据在一个方向上传输，它实际上是一种切换方向的单工通信，就和对讲机(步话机)一样；

全双工数据通信允许数据同时在两个方向上传输，因此，全双工通信是两个单工通信方式的结合，它要求发送设备和接收设备都有独立的接收和发送能力，就和电话一样。

**JAVA网络编程**



半双工网络C/S演示

[MyServer.java]

/\*\*

\* 这是第一个服务器端程序，让它在9999端口监听

\* 可以接收客户端发来的数据

\*/

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.\*;

public class MyServer {

public static void main(String[] args) {

new MyServer();

}

//构造函数

public MyServer(){

try {

//在9999号端口上监听

ServerSocket ss=new ServerSocket(9999);

System.out.println("9999端口监听...");

//等待某个客户端来连接，该函数会返回一个Socket连接

Socket s=ss.accept();

//要读取s中传递的数据

InputStreamReader isr=new InputStreamReader(s.getInputStream());

BufferedReader br=new BufferedReader(isr);

String info=br.readLine();

System.out.println("服务器接收到:\t"+info);

PrintWriter pw=new PrintWriter(s.getOutputStream(),true);

pw.println("我是服务器，已收到你发送的信息!");

} catch (IOException e) {

e.printStackTrace();

}

}

}

-------------------------------------------------------------------------------

[MyClient.java]

/\*\*

\* 这是一个客户端程序，可以连接服务器端口9999

\*/

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.\*;

public class MyClient {

public static void main(String[] args) {

new MyClient();

}

public MyClient(){

try {

//Socket()就是去连接某个服务器端 127.0.0.1表示服务器的ip

//9999是服务器的端口号

Socket s=new Socket("127.0.0.1",9999);

//如果s连接成功，就可以发送数据到服务器端

//我们通过pw向s写数据,true表示即时刷新

PrintWriter pw=new PrintWriter(s.getOutputStream(),true);

pw.println("你好吗？我是客户端");

//要读取s中传递的数据

InputStreamReader isr=new InputStreamReader(s.getInputStream());

BufferedReader br=new BufferedReader(isr);

String info=br.readLine();

System.out.println("接收到服务器：\t"+info);

} catch (Exception e) {

e.printStackTrace();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

半双工，客户端与服务端进行控制台输入交流

[MyServer2.java]服务端源代码

/\*\*

\* 功能：是一个服务器端，在9999端口上监听

\* 可以通过控制台,输入回送给客户端的信息

\*/

package com.net.server2;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.\*;

public class MyServer2 {

InputStreamReader isr,isr2;

BufferedReader br,br2;

PrintWriter pw;

public static void main(String[] args) {

new MyServer2();

}

//构造函数

public MyServer2(){

try {

//在9999号端口上监听

ServerSocket ss=new ServerSocket(9999);

//等待客户端连接，该函数会返回一个Socket连接

Socket s=ss.accept();

//要读取s中传递的数据

isr=new InputStreamReader(s.getInputStream());

br=new BufferedReader(isr);

pw=new PrintWriter(s.getOutputStream(),true);

//接收从控制台输入的信息

isr2=new InputStreamReader(System.in);

br2=new BufferedReader(isr2);

while(true){

//接收客户端信息

String infoFromClient=br.readLine();

System.out.println("服务器接收到:\t"+infoFromClient);

//接收从控制台输入的信息

System.out.println("请输入要发送到客户端的信息：");

String response=br2.readLine();

//服务器从控制台上接收bye服务器端退出

if(response.equals("bye")){

System.out.println("退出对话");

s.close();

break;

}

//把从控制台输入的信息，回送给客户端

pw.println(response);

}

} catch (IOException e) {

e.printStackTrace();

}finally{

try {

if(br2!=null) br2.close();

if(isr2!=null) isr2.close();

if(pw!=null) pw.close();

if(br!=null) br.close();

if(isr!=null) isr.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[MyClient2.java]客户端源代码

/\*\*

\* 这是一个客户端程序，可以连接服务器端口9999

\*/

package com.net.client2;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.\*;

public class MyClient2 {

InputStreamReader isr,isr2;

BufferedReader br,br2;

PrintWriter pw;

public static void main(String[] args) {

new MyClient2();

}

public MyClient2(){

try {

//连接服务器端

Socket s=new Socket("127.0.0.1",9999);

pw=new PrintWriter(s.getOutputStream(),true);

//控制台接收信息

isr=new InputStreamReader(System.in);

br=new BufferedReader(isr);

//从服务器接收信息

isr2=new InputStreamReader(s.getInputStream());

br2=new BufferedReader(isr2);

while(true){

//客户端先从控制台接收

String info=br.readLine();

//然后发送给服务器

pw.println(info);

//客户端从控制台上输入byebye信息，客户端退出

if(info.equals("byebye")){

System.out.println("客户端退出对话");

s.close();

break;

}

//接收从服务器发来的信息

String res=br2.readLine();

System.out.println("客户端接收到的信息："+res);

//客户端接收服务端发送来的byebye信息，客户端退出

if(res.equals("byebye")){

System.out.println("服务器端强制客户端退出对话");

s.close();

break;

}

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(br2!=null) br2.close();

if(isr2!=null) isr2.close();

if(pw!=null) pw.close();

if(br!=null) br.close();

if(isr!=null) isr.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

全双工客户端与服务器进行简易界面聊天

[MyServer3.java]服务器端源代码

/\*\*

\* 全双工服务器端，让它在9999端口监听

\* 可以接收客户端发来的数据

\*/

package com.net.server3;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.\*;

import javax.swing.AbstractAction;

import javax.swing.JButton;

import javax.swing.JComponent;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.KeyStroke;

public class MyServer3 extends JFrame implements ActionListener{

JTextArea jta;

JTextField jtf;

JButton jb;

JPanel jp1;

JScrollPane jsp;

InputStreamReader isr;

BufferedReader br;

PrintWriter pw;

public static void main(String[] args) {

new MyServer3();

}

//构造函数

public MyServer3(){

jta=new JTextArea();

jtf=new JTextField(20);

jb=new JButton("发送");

jb.addActionListener(this);

//给jb发送按钮添加快捷键(回车)事件

jb.getInputMap(JComponent.WHEN\_IN\_FOCUSED\_WINDOW).put(KeyStroke.getKeyStroke(KeyEvent.VK\_ENTER, 0), "fasong");

jb.getActionMap().put("fasong", new AbstractAction() {

public void actionPerformed(ActionEvent e) {

String info=jtf.getText();

jta.append("客户端 对 服务端说："+info+"\r\n");

pw.println(info);

jtf.setText("");

}

});

jp1=new JPanel();

jp1.add(jtf);

jp1.add(jb);

jsp=new JScrollPane(jta);

this.add(jsp,"Center");

this.add(jp1,"South");

this.setTitle("qq简易聊天--服务端");

this.setSize(350, 250);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

try {

//在9999号端口上监听

ServerSocket ss=new ServerSocket(9999);

//等待客户端连接

Socket s=ss.accept();

//读取从客户端发来的信息

isr=new InputStreamReader(s.getInputStream());

br=new BufferedReader(isr);

pw=new PrintWriter(s.getOutputStream(),true);

while(true){

//读取从客户端发来的信息

String info=br.readLine();

jta.append("客户端 对 服务端说："+info+"\r\n");

}

} catch (IOException e) {

e.printStackTrace();

}finally{

try {

if(pw!=null) pw.close();

if(br!=null) br.close();

if(isr!=null) isr.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

//如果用户按下发送信息的按钮

if(e.getSource()==jb){

//把服务器在jtf写的 内容发送给客户端

String info=jtf.getText();

//向jta中不断追加内容

jta.append("服务端 对 客户端说："+info+"\r\n");

pw.println(info);

//清空jtf内容

jtf.setText("");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[MyClient3.java]客户端源代码

/\*\*

\* 这是一个客户端程序，可以连接服务器端口9999

\*/

package com.net.client3;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.\*;

import javax.swing.AbstractAction;

import javax.swing.JButton;

import javax.swing.JComponent;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.KeyStroke;

public class MyClient3 extends JFrame implements ActionListener{

JTextArea jta;

JTextField jtf;

JButton jb;

JPanel jp1;

JScrollPane jsp;

InputStreamReader isr;

BufferedReader br;

PrintWriter pw;

public static void main(String[] args) {

new MyClient3();

}

public MyClient3(){

jta=new JTextArea();

jtf=new JTextField(20);

jtf.setFocusable(true);

jb=new JButton("发送");

jb.addActionListener(this);

//给jb发送按钮添加快捷键(回车)事件

jb.getInputMap(JComponent.WHEN\_IN\_FOCUSED\_WINDOW).put(KeyStroke.getKeyStroke(KeyEvent.VK\_ENTER, 0), "fasong");

jb.getActionMap().put("fasong", new AbstractAction() {

public void actionPerformed(ActionEvent e) {

String info=jtf.getText();

jta.append("客户端 对 服务端说："+info+"\r\n");

pw.println(info);

jtf.setText("");

}

});

jp1=new JPanel();

jp1.add(jtf);

jp1.add(jb);

jsp=new JScrollPane(jta);

this.add(jsp,"Center");

this.add(jp1,"South");

this.setTitle("qq简易聊天--客户端");

this.setSize(350, 250);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

try {

//Socket()就是去连接某个服务器端 127.0.0.1表示服务器的ip

Socket s=new Socket("127.0.0.1",9999);

InputStreamReader isr=new InputStreamReader(s.getInputStream());

BufferedReader br=new BufferedReader(isr);

pw=new PrintWriter(s.getOutputStream(),true);

while(true){

//读取服务器端发来的信息

String info=br.readLine();

jta.append("服务端 对 客户端说："+info+"\r\n");

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(pw!=null) pw.close();

if(br!=null) br.close();

if(isr!=null) isr.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jb){

String info=jtf.getText();

jta.append("客户端 对 服务端说："+info+"\r\n");

pw.println(info);

jtf.setText("");

}

}

}

补充讲解:

如何在网络间传递对象？

[MyServer.java]服务器源代码

package com.test.server;

import java.io.ObjectInputStream;

import java.net.ServerSocket;

import java.net.Socket;

import com.common.User;

public class MyServer {

ServerSocket ss;

Socket s;

ObjectInputStream ois;

public static void main(String[] args) {

new MyServer();

}

//构造函数

public MyServer(){

try {

ss=new ServerSocket(3456);

s=ss.accept();

//以对象流的方式读取(假设客户端发送的是User的一个对象)

ois=new ObjectInputStream(s.getInputStream());

User u=(User)ois.readObject();

//输出

System.out.println(u.getName()+"\t"+u.getPasswd());

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(ois!=null) ois.close();

if(s!=null) s.close();

if(ss!=null) ss.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

}

[User.java]服务端的公共服务

package com.common;

import java.io.Serializable;

public class User implements Serializable{

private String name;

private String passwd;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPasswd() {

return passwd;

}

public void setPasswd(String passwd) {

this.passwd = passwd;

}

}

[MyClient.java]客户端源代码

package com.test.client;

import java.io.ObjectOutputStream;

import java.net.Socket;

import com.common.User;

public class MyClient {

Socket s;

ObjectOutputStream oos;

public static void main(String[] args) {

new MyClient();

}

//构造函数

public MyClient(){

try {

s=new Socket("127.0.0.1",3456);

//通过ObjectOutputStream给服务器传送对象

oos=new ObjectOutputStream(s.getOutputStream());

User u=new User();

u.setName("张三");

u.setPasswd("123");

oos.writeObject(u);

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(oos!=null) oos.close();

if(s!=null) s.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

}

[User.java]客户端的公共服务

package com.common;

import java.io.Serializable;

//通过io在网络上传送对象需要进行序列化Serializable

public class User implements Serializable{

private String name;

private String passwd;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPasswd() {

return passwd;

}

public void setPasswd(String passwd) {

this.passwd = passwd;

}

}

特别说明：

在java网络编程需要通过IO流实现对象的传递需要对，服务端和客户端定义相同的传递对对向方法，以达到实现对象的传递。同时需要对方法进行Serializable序列化，否则报错。

**QQ案例**

说明：该案例主讲内容：java网络编程、多线程、C/S数据交互

附源代码

注：代码分为Server部分与Client部分,由于此案例仅为对NET操作未使用数据库

**Server部分包含的包有：**

1. com.qq.server.view (服务端界面包)

1.1、QqServerFrame.java (服务端主页面)

1. com.qq.server.model (业务逻辑包)

2.1、MyQqServer.java (服务器验证客户信息)

2.2、QqServerUser.java (用于管理QQ用户信息[此包功能应为对数据库逻辑操作])(空)

2.3、SerConClientThread.java (服务器连接客户端线程)

2.4、ManageClientThrea.java (管理客户端连接线程)

3、com.qq.server.db (数据库操作包)

3.1、SqlHelper.java (对数据库的CURD操作)(空)

4、com.qq.server.tools (服务端工具包)(空)

5、com.qq.common (通用包)

5.1、Message.java (消息类)

5.2、MessageType.java (消息类型接口类)

5.3、User.java (用户信息类)

**Client部分包含的包有：**

1. com.qq.client.view (客户端界面包)
   1. QqClientLogin.java (QQ登录界面)
   2. QqFriendList.java (好友列表界面)
   3. QqChat.java (聊天界面)
2. com.qq.client.model (客户端业务逻辑包)

2.1、QqClienConServer.java (客户端连接服务端的后台用于与服务端数据交互)

2.2、QqClientUser.java (客户端用户操作逻辑类)

3、com.qq.client.tools (原应为客户端工具包，现为处理与服务端的线程包)

3.1、ClientConServerThread.java (客户端连接服务器线程类)

3.2、ManageClientConServerThread.java (管理C/S线程类)

3.3、ManageQqChat.java (管理聊天界面类)

3.4、ManageQqFriendList.java (管理好友列表类)

4、com.qq.common (通用包)

4.1、Message.java (消息类)

4.2、MessageType.java (消息类型类)

4.3、User.java (用户信息类)

注意：Server与Client中的com.qq.common通用包主要功能是统一保持客户端与服务端的用户及消息一致。

**下见源代码:**

**Server端源代码:**

com.qq.server.view

[QqServerFrame.java]

/\*\*

\* 服务器端的控制界面

\* 功能：

\* 1、完成启动、关闭服务器

\* 2、管理及监控用户

\*/

package com.qq.server.view;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import com.qq.server.model.MyQqServer;

public class QqServerFrame extends JFrame implements ActionListener{

JPanel jp1;

JButton jb1,jb2;

public static void main(String []args){

new QqServerFrame();

}

//构造函数

public QqServerFrame(){

jp1=new JPanel();

jb1=new JButton("启动服务器");

jb1.addActionListener(this);

jb2=new JButton("关闭服务器");

jb2.addActionListener(this);

jp1.add(jb1);

jp1.add(jb2);

this.add(jp1);

this.setSize(500, 400);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jb1){

new MyQqServer();

}else if(e.getSource()==jb2){

System.exit(0);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.qq.server.model

[MyQqServer.java]

/\*\*

\* 这是QQ服务器，它在监听等待某个QQ客户端连接服务器

\*/

package com.qq.server.model;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import com.qq.common.Message;

import com.qq.common.User;

public class MyQqServer {

BufferedReader br;

public MyQqServer(){

try {

System.out.println("服务器在监听9999端口...");

//在9999端口监听

ServerSocket ss=new ServerSocket(9999);

//等待连接

while(true){

Socket s=ss.accept();

//接收客户端发来的信息

ObjectInputStream ois=new ObjectInputStream(s.getInputStream());

User u=(User)ois.readObject();

Message m=new Message();

ObjectOutputStream oos=new ObjectOutputStream(s.getOutputStream());

//此处代替数据库密码验证

if(u.getPasswd().equals("12345")){

//返回一个成功登录的信息包

m.setMesType("1");

oos.writeObject(m);

//这里就单开一个线程，让该线程与该客户端保持通讯

SerConClientThread scct=new SerConClientThread(s);

ManageClientThrea.addClientThrea(u.getUserId(), scct);

//启动与该客户端通讯的线程

scct.start();

//并通知其它在线用户

scct.notifyOther(u.getUserId());

}else{

m.setMesType("2");

oos.writeObject(m);

//关闭连接

s.close();

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[SerConClientThread.java]

/\*\*

\* 功能：是服务器与某个客户端的通讯线程

\*/

package com.qq.server.model;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.util.HashMap;

import java.util.Iterator;

import com.qq.common.Message;

import com.qq.common.MessageType;

public class SerConClientThread extends Thread{

Socket s;

public SerConClientThread(Socket s){

//把服务器与该客户端的连接赋给s

this.s=s;

}

//让该线程去通知其它在线用户

public void notifyOther(String iam){

//得到所有在线的用户的线程

HashMap hm=ManageClientThrea.hm;

Iterator it=hm.keySet().iterator();

while(it.hasNext()){

Message m=new Message();

m.setCon(iam);

m.setMesType(MessageType.message\_ret\_onLineFriend);

//取出在线用户的id

String onLineUserId=it.next().toString();

try {

ObjectOutputStream oos=new ObjectOutputStream(ManageClientThrea.getClientThrea(onLineUserId).s.getOutputStream());

m.setGetder(onLineUserId);

oos.writeObject(m);

} catch (IOException e) {

e.printStackTrace();

}

}

}

public void run(){

while(true){

//这里该线程就可以接收客户端发来的信息

try {

ObjectInputStream ois=new ObjectInputStream(s.getInputStream());

Message m=(Message)ois.readObject();

System.out.println(m.getSender()+" 给 "+m.getGetder()+" 说:"+m.getCon());

//对从客户端取得的消息进行类型判断，然后做相应的处理

if(m.getMesType().equals(MessageType.message\_comm\_mes)){

//服务器完成转发功能

//取得接收人的通讯线程

SerConClientThread sc=ManageClientThrea.getClientThrea(m.getGetder());

ObjectOutputStream oos=new ObjectOutputStream(sc.s.getOutputStream());

oos.writeObject(m);

}else if(m.getMesType().equals(MessageType.message\_get\_onLineFriend)){

//把在服务器的好友给该客户端返回

String res=ManageClientThrea.getAllOnLineUserid();

Message m2=new Message();

m2.setMesType(MessageType.message\_ret\_onLineFriend);

m2.setCon(res);

m2.setGetder(m.getSender());

ObjectOutputStream oos=new ObjectOutputStream(s.getOutputStream());

oos.writeObject(m2);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ManageClientThrea.java]

/\*\*

\* 这是一个管理客户端线程的类

\*/

package com.qq.server.model;

import java.util.HashMap;

import java.util.Iterator;

public class ManageClientThrea {

public static HashMap hm=new HashMap<String,SerConClientThread>();

//向hashmap中添加一个客户端通讯线程

public static void addClientThrea(String uid,SerConClientThread scct){

hm.put(uid,scct);

}

public static SerConClientThread getClientThrea(String uid){

return (SerConClientThread)hm.get(uid);

}

//返回当前在线的人的好友情况

public static String getAllOnLineUserid(){

//使用迭代器完成

Iterator it=hm.keySet().iterator();

String res="";

while(it.hasNext()){

res+=it.next().toString()+" ";

}

return res;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.qq.common

[Message.java]

/\*\*

\* 为Message规定规则

\* mesType1-->表明登录成功

\* mesType2-->表明登录失败

\* mesType3-->表明普通消息包

\*/

package com.qq.common;

import java.io.Serializable;

public class Message implements Serializable{

private String mesType;

private String sender;

private String getder;

private String con;

private String sendTime;

public String getGetder() {

return getder;

}

public void setGetder(String getder) {

this.getder = getder;

}

public String getSender() {

return sender;

}

public void setSender(String sender) {

this.sender = sender;

}

public String getCon() {

return con;

}

public void setCon(String con) {

this.con = con;

}

public String getSendTime() {

return sendTime;

}

public void setSendTime(String sendTime) {

this.sendTime = sendTime;

}

public String getMesType() {

return mesType;

}

public void setMesType(String mesType) {

this.mesType = mesType;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[MessageType.java]

/\*\*

\* 定义消息包的种类的接口

\*/

package com.qq.common;

public interface MessageType {

String message\_succeed="1";//登录成功包

String message\_login\_fail="2";//登录失败包

String message\_comm\_mes="3";//普通消息包

String message\_get\_onLineFriend="4";//要求在线的好友包

String message\_ret\_onLineFriend="5";//返回在线好友包

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[User.java]

/\*\*

\* 这是用户信息类

\*/

package com.qq.common;

import java.io.Serializable;

public class User implements Serializable{

private String userId;

private String passwd;

public String getUserId() {

return userId;

}

public void setUserId(String userId) {

this.userId = userId;

}

public String getPasswd() {

return passwd;

}

public void setPasswd(String passwd) {

this.passwd = passwd;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Client端源代码**

com.qq.client.view

[QqClientLogin.java]

/\*\*

\* 功能：QQ客户端登陆界面

\*/

package com.qq.client.view;

import java.awt.Color;

import java.awt.Cursor;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import java.io.ObjectOutputStream;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTabbedPane;

import javax.swing.JTextField;

import javax.swing.UIManager;

import com.qq.client.model.QqClientUser;

import com.qq.client.tools.ManageClientConServerThread;

import com.qq.client.tools.ManageQqFriendList;

import com.qq.common.Message;

import com.qq.common.MessageType;

import com.qq.common.User;

public class QqClientLogin extends JFrame implements ActionListener{

//定义北部需要的组件

JLabel jbl1;

//定义中部需要的组件

//中部有三个JPanel,有一个叫选项卡窗口管理

JTabbedPane jtp;

JPanel jp2,jp3,jp4;

JLabel jp2\_jbl1,jp2\_jbl2,jp2\_jbl3,jp2\_jbl4;

JButton jp2\_jb1;

JTextField jp2\_jtf;

JPasswordField jp2\_jpf;

JCheckBox jp2\_jcb1,jp2\_jcb2;

//定义南部需要的组件

JPanel jp1;

JButton jp1\_jb1,jp1\_jb2,jp1\_jb3;

public static void main(String[] args) {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e) {

e.printStackTrace();

}

new QqClientLogin();

}

public void initNorthLayout(){

//处理北部

jbl1=new JLabel(new ImageIcon("image/qqview.jpg"));

}

public void initCenterLayout(){

//处理中部

jp2=new JPanel();

jp2.setLayout(null);

jp2\_jbl1=new JLabel("QQ号码:");

jp2\_jbl1.setBounds(30, 15, 60, 25);

jp2\_jbl2=new JLabel("QQ密码:");

jp2\_jbl2.setBounds(30, 50, 60, 25);

jp2\_jbl3=new JLabel("忘记密码");

jp2\_jbl3.setForeground(Color.blue);

jp2\_jbl3.setBounds(260, 50, 80, 25);

jp2\_jbl3.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));

jp2\_jbl4=new JLabel("<html><u>申请密码保护</u><html>");

jp2\_jbl4.setForeground(Color.blue);

jp2\_jbl4.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));

jp2\_jbl4.setBounds(250, 85, 100, 25);

jp2\_jb1=new JButton("清除号码");

jp2\_jb1.setBounds(245, 15, 80, 25);

jp2\_jtf=new JTextField();

jp2\_jtf.setBounds(80, 15, 150, 25);

jp2\_jpf=new JPasswordField();

jp2\_jpf.setBounds(80, 50, 150, 25);

jp2\_jcb1=new JCheckBox("隐身登录");

jp2\_jcb1.setBounds(75, 85, 80, 25);

jp2\_jcb2=new JCheckBox("记住密码");

jp2\_jcb2.setBounds(155, 85, 80, 25);

//把控制按顺序加入jp2

jp2.add(jp2\_jbl1);

jp2.add(jp2\_jtf);

jp2.add(jp2\_jb1);

jp2.add(jp2\_jbl2);

jp2.add(jp2\_jpf);

jp2.add(jp2\_jbl3);

jp2.add(jp2\_jcb1);

jp2.add(jp2\_jcb2);

jp2.add(jp2\_jbl4);

//jp3,jp4

jp3=new JPanel();

jp4=new JPanel();

//创建选项卡窗口

jtp=new JTabbedPane();

jtp.add(jp2,"QQ号码");

jtp.add(jp3,"手机号码");

jtp.add(jp4,"电子邮件");

}

public void initSouthLayout(){

//处理南部

jp1=new JPanel();

jp1\_jb1=new JButton("登录");

jp1\_jb1.addActionListener(this);

jp1\_jb2=new JButton("取消");

jp1\_jb2.addActionListener(this);

jp1\_jb3=new JButton("注册");

//把三个按钮放到jp1

jp1.add(jp1\_jb1);

jp1.add(jp1\_jb2);

jp1.add(jp1\_jb3);

}

//构造函数

public QqClientLogin(){

//布局北部

this.initNorthLayout();

//布局中部

this.initCenterLayout();

//布局南部

this.initSouthLayout();

this.add(jbl1,"North");

this.add(jtp,"Center");

this.add(jp1,"South");

this.setTitle("QQ登录界面");

this.setResizable(false);

this.setSize(350, 300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==jp1\_jb1){

QqClientUser qqClientUser=new QqClientUser();

User u=new User();

u.setUserId(jp2\_jtf.getText().trim());

u.setPasswd(new String(jp2\_jpf.getPassword()));

if(qqClientUser.checkUser(u)){

//发送一个要求返回在线好友的请求包

try {

this.dispose();

//把创建好友列表的语句提前，否则是不能成功的

QqFriendList qqList=new QqFriendList(u.getUserId());

ManageQqFriendList.addQqFriendList(u.getUserId(), qqList);

ObjectOutputStream oos=new ObjectOutputStream(ManageClientConServerThread.getClientConServerThread(u.getUserId()).getS().getOutputStream());

//做一个Message

Message m=new Message();

//指明我的是这个QQ号的好友在线情况

m.setSender(u.getUserId());

m.setMesType(MessageType.message\_get\_onLineFriend);

oos.writeObject(m);

} catch (Exception e1) {

e1.printStackTrace();

}

}else{

JOptionPane.showMessageDialog(this, "帐号或密码错误，请核实后重新登录!","登录提示", JOptionPane.ERROR\_MESSAGE);

return;

}

}else if(e.getSource()==jp1\_jb2){

System.exit(0);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[QqFriendList.java]

/\*\*

\* 功能：我的好友列表界面(也包括陌生人、黑名单)

\*/

package com.qq.client.view;

import java.awt.BorderLayout;

import java.awt.CardLayout;

import java.awt.Color;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import com.qq.client.tools.ManageQqChat;

import com.qq.common.Message;

public class QqFriendList extends JFrame implements ActionListener,MouseListener{

//处理第一张卡片(我的好友)

JPanel jphy1,jphy2,jphy3;

JButton jphy1\_jb1,jphy1\_jb2,jphy1\_jb3;

JScrollPane jsp1;

String ownerId;

JLabel []jbls;

//处理第二张卡片(陌生人)

JPanel jpmsr1,jpmsr2,jpmsr3;

JButton jpmsr\_jb1,jpmsr\_jb2,jpmsr\_jb3;

JScrollPane jsp2;

//处理第三张卡片(黑名单)

JPanel jphmd1,jphmd2,jphmd3;

JButton jphmd\_jb1,jphmd\_jb2,jphmd\_jb3;

JScrollPane jsp3;

//把整个JFrame设置成CardLayout

CardLayout cl;

// public static void main(String[] args) {

// new QqFriendList();

// }

public void initCard1(){

//处理第一张卡片(显示好友列表)

jphy1\_jb1=new JButton("我的好友");

jphy1\_jb2=new JButton("陌生人");

jphy1\_jb2.addActionListener(this);

jphy1\_jb3=new JButton("黑名单");

jphy1\_jb3.addActionListener(this);

jphy1=new JPanel(new BorderLayout());

//假定有50个好友

jphy2=new JPanel(new GridLayout(50, 1, 4, 4));

//给jp\_hy2初始化50个好友

jbls=new JLabel[50];

for(int i=0;i<jbls.length;i++){

jbls[i]=new JLabel(i+1+"",new ImageIcon("image/tou1\_1.jpg"),JLabel.LEFT);

jbls[i].setEnabled(false);

//判断在线

if(jbls[i].getText().equals(ownerId)){

jbls[i].setEnabled(true);

}

jbls[i].addMouseListener(this);

jphy2.add(jbls[i]);

}

jphy3=new JPanel(new GridLayout(2, 1));

//把两个按钮加入到jp\_hy3中

jphy3.add(jphy1\_jb2);

jphy3.add(jphy1\_jb3);

jsp1=new JScrollPane(jphy2);

//对jp\_hy1初始化

jphy1.add(jphy1\_jb1,"North");

jphy1.add(jsp1,"Center");

jphy1.add(jphy3,"South");

}

public void initCard2(){

//处理第二张卡片

jpmsr\_jb1=new JButton("我的好友");

jpmsr\_jb1.addActionListener(this);

jpmsr\_jb2=new JButton("陌生人");

jpmsr\_jb3=new JButton("黑名单");

jpmsr\_jb3.addActionListener(this);

jpmsr1=new JPanel(new BorderLayout());

//假定有20个陌生人

jpmsr2=new JPanel(new GridLayout(20, 1, 4, 4));

//给jp\_hy2初始化20个陌生人

JLabel []jbls2=new JLabel[20];

for(int i=0;i<jbls2.length;i++){

jbls2[i]=new JLabel(i+1+"",new ImageIcon("image/tou1\_1.jpg"),JLabel.LEFT);

jbls2[i].setEnabled(false);

if(jbls2[i].getText().equals(ownerId)){

jbls2[i].setEnabled(true);

}

jbls2[i].addMouseListener(this);

jpmsr2.add(jbls2[i]);

}

jpmsr3=new JPanel(new GridLayout(2, 1));

//把两个按钮加入到jp\_hy3中

jpmsr3.add(jpmsr\_jb1);

jpmsr3.add(jpmsr\_jb2);

jsp2=new JScrollPane(jpmsr2);

//对jp\_hy1初始化

jpmsr1.add(jpmsr3,"North");

jpmsr1.add(jsp2,"Center");

jpmsr1.add(jpmsr\_jb3,"South");

}

public void initCard3(){

//处理第三张卡片(黑名单)

jphmd\_jb1=new JButton("我的好友");

jphmd\_jb1.addActionListener(this);

jphmd\_jb2=new JButton("陌生人");

jphmd\_jb2.addActionListener(this);

jphmd\_jb3=new JButton("黑名单");

jphmd1=new JPanel(new BorderLayout());

//假定有5个黑名单

jphmd2=new JPanel(new GridLayout(5, 1, 4, 4));

//给jp\_hy2初始化5个黑名单

JLabel []jbls3=new JLabel[5];

for(int i=0;i<jbls3.length;i++){

jbls3[i]=new JLabel(i+1+"",new ImageIcon("image/tou1\_1.jpg"),JLabel.LEFT);

jbls3[i].setEnabled(false);

if(jbls3[i].getText().equals(ownerId)){

jbls3[i].setEnabled(true);

}

jbls3[i].addMouseListener(this);

jphmd2.add(jbls3[i]);

}

jphmd3=new JPanel(new GridLayout(3, 1));

//把两个按钮加入到jp\_hy3中

jphmd3.add(jphmd\_jb1);

jphmd3.add(jphmd\_jb2);

jphmd3.add(jphmd\_jb3);

jsp3=new JScrollPane(jphmd2);

//对jp\_hy1初始化

jphmd1.add(jphmd3,"North");

jphmd1.add(jsp3,"Center");

}

//更新在线的好友情况

public void updateFriend(Message m){

String onLineFriend[]=m.getCon().split(" ");

for(int i=0;i<onLineFriend.length;i++){

jbls[Integer.parseInt(onLineFriend[i])-1].setEnabled(true);

}

}

//构造函数

public QqFriendList(String ownerId){

this.ownerId=ownerId;

this.initCard1();

this.initCard2();

this.initCard3();

cl=new CardLayout();

this.setLayout(cl);

this.add(jphy1,"1");

this.add(jpmsr1,"2");

this.add(jphmd1,"3");

//在窗口显示自己的编号

this.setTitle(ownerId);

this.setSize(150, 450);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

//如果点击了我的好友中的陌生人按钮，就显示第2张卡片

if(e.getSource()==jphy1\_jb2){

cl.show(this.getContentPane(),"2");

}

//如果点击了我的好友中的黑名单按钮，就显示第3张卡片

else if(e.getSource()==jphy1\_jb3){

cl.show(this.getContentPane(),"3");

}

//如果点击了陌生人中的我的好友按钮，就显示第1张卡片

else if(e.getSource()==jpmsr\_jb1){

cl.show(this.getContentPane(), "1");

}

//如果点击了陌生人中的黑名单按钮，就显示第3张卡片

else if(e.getSource()==jpmsr\_jb3){

cl.show(this.getContentPane(),"3");

}

//如果点击了黑名单中的我的好友按钮，就显示第1张卡片

else if(e.getSource()==jphmd\_jb1){

cl.show(this.getContentPane(),"1");

}

//如果点击了黑名单中的陌生人按钮，就显示第2张卡片

else if(e.getSource()==jphmd\_jb2){

cl.show(this.getContentPane(),"2");

}

}

@Override

public void mouseClicked(MouseEvent e) {

//响应用户双击鼠标左键的事件，并得到好友的编号

if(e.getButton()==1&&e.getClickCount()==2){

//得到该好友的编号

String friendNo=((JLabel)e.getSource()).getText();

QqChat qqChat=new QqChat(friendNo,this.ownerId);

//把聊天界面加入到管理类中

ManageQqChat.addQqChat(this.ownerId+" "+friendNo, qqChat);

}

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseEntered(MouseEvent e) {

JLabel jl=(JLabel)e.getSource();

jl.setForeground(Color.red);

}

@Override

public void mouseExited(MouseEvent e) {

JLabel jl=(JLabel)e.getSource();

jl.setForeground(Color.black);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[QqChat.java]

/\*\*

\* 这是与好友聊天的界面

\* 因为客户端，要处于读取的状态，因此我们把它做成一个线程

\*/

package com.qq.client.view;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Date;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import com.qq.client.model.QqClientConServer;

import com.qq.client.tools.ManageClientConServerThread;

import com.qq.common.Message;

import com.qq.common.MessageType;

public class QqChat extends JFrame implements ActionListener {

JTextArea jta;

JTextField jtf;

JButton jb;

JPanel jp;

JScrollPane jsp;

String ownerId;

String friendId;

// public static void main(String[] args) {

// new QqChat("小四");

// }

// 构造函数

public QqChat(String friendId, String ownerId) {

this.ownerId = ownerId;

this.friendId = friendId;

jta = new JTextArea();

jtf = new JTextField(15);

jb = new JButton("发送");

jb.addActionListener(this);

jp = new JPanel();

jp.add(jtf);

jp.add(jb);

jsp = new JScrollPane(jta);

this.add(jsp, "Center");

this.add(jp, "South");

this.setTitle(ownerId + " 正在与 " + friendId + " 聊天");

this.setSize(350, 300);

this.setIconImage((new ImageIcon("image/qie.jpg")).getImage());

this.setLocationRelativeTo(null);

this.setVisible(true);

}

// 写一个方法，让它显示消息

public void showMessage(Message m){

String info=m.getSender()+" 对 "+m.getGetder()+" 说:"+m.getCon()+"\r\n";

jta.append(info);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == jb) {

// 如果用户点击了发送按钮

Message m = new Message();

m.setMesType(MessageType.message\_comm\_mes);

m.setSender(this.ownerId);

m.setGetder(this.friendId);

m.setCon(this.jtf.getText());

m.setSendTime(new Date().toString());

jta.append(ownerId + " 对 " + friendId + " 说:" + jtf.getText()

+ "\r\n");

jtf.setText("");

// 发送给服务器

try {

ObjectOutputStream oos = new ObjectOutputStream(

ManageClientConServerThread

.getClientConServerThread(ownerId).getS()

.getOutputStream());

oos.writeObject(m);

} catch (Exception e1) {

e1.printStackTrace();

}

}

}

// @Override

// public void run() {

// while(true){

// //读取[如果读不到就等待]

// try {

// ObjectInputStream ois=new

// ObjectInputStream(QqClientConServer.s.getInputStream());

// Message m=(Message)ois.readObject();

// //显示出来

// String info=m.getSender()+" 对 "+m.getGetder()+" 说:"+m.getCon()+"\r\n";

// jta.append(info);

// } catch (Exception e) {

// e.printStackTrace();

// }

// }

// }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.qq.client.model

[QqClienConServer.java]

/\*\*

\* 这是客户端连接服务器的后台

\* 功能：客户端与服务器进行数据交互

\*/

package com.qq.client.model;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import com.qq.client.tools.ClientConServerThread;

import com.qq.client.tools.ManageClientConServerThread;

import com.qq.common.Message;

import com.qq.common.User;

public class QqClientConServer {

public Socket s;

//发送第一次请求

public boolean sendLoginInfoToServer(Object o){

boolean b=false;

try {

s=new Socket("127.0.0.1",9999);

ObjectOutputStream oos=new ObjectOutputStream(s.getOutputStream());

oos.writeObject(o);

ObjectInputStream ois=new ObjectInputStream(s.getInputStream());

Message ms=(Message)ois.readObject();

//这里就是验证用户登录的地方

if(ms.getMesType().equals("1")){

b=true;

//就创建一个该qq号和服务器端保持通讯连接的线程

ClientConServerThread ccst=new ClientConServerThread(s);

//启动该通讯线程

ccst.start();

ManageClientConServerThread.addClientConServerThread(((User)o).getUserId(), ccst);

}

} catch (Exception e) {

e.printStackTrace();

} finally{

}

return b;

}

//向服务器发送一个对象

public void SendInfoToServer(Object o){

try {

Socket s=new Socket("127.0.0.1",9999);

} catch (Exception e) {

e.printStackTrace();

} finally{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[QqClientUser.java]

/\*\*

\* 用户操作逻辑类

\* QqClientUser会调用QqClientConServer的

\* 方法sendLoginInfoToServer向服务器发送

\*/

package com.qq.client.model;

import com.qq.common.User;

public class QqClientUser {

//验证用户是否合法

public boolean checkUser(User u){

return new QqClientConServer().sendLoginInfoToServer(u);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.qq.client.tools

[ClientConServerThread.java]

/\*\*

\* 这是客户端和服务器端保持通讯的线程

\*/

package com.qq.client.tools;

import java.io.ObjectInputStream;

import java.net.Socket;

import com.qq.client.view.QqChat;

import com.qq.client.view.QqFriendList;

import com.qq.common.Message;

import com.qq.common.MessageType;

public class ClientConServerThread extends Thread{

private Socket s;

public Socket getS() {

return s;

}

public void setS(Socket s) {

this.s = s;

}

//构造函数

public ClientConServerThread(Socket s){

this.s=s;

}

public void run(){

while(true){

//不停的读取从服务器端发来的消息

try {

ObjectInputStream ois=new ObjectInputStream(s.getInputStream());

Message m=(Message)ois.readObject();

//按消息包的类型来处理不同的包

if(m.getMesType().equals(MessageType.message\_comm\_mes)){

//把从服务器获得的消息，显示到该显示的聊天界面上

QqChat qqChat=ManageQqChat.getQqChat(m.getGetder()+" "+m.getSender());

//显示

qqChat.showMessage(m);

}else if(m.getMesType().equals(MessageType.message\_ret\_onLineFriend)){

String con=m.getCon();

String []friends=con.split(" ");

String getter=m.getGetder();

//修改相应的好友列表

QqFriendList qqFrindList=ManageQqFriendList.getQqFriendList(getter);

//更新在线好友

if(qqFrindList!=null){

qqFrindList.updateFriend(m);

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ManageClientConServerThread.java]

/\*\*

\* 这是一个管理客户端与服务器保持通讯的线程类

\*/

package com.qq.client.tools;

import java.util.HashMap;

public class ManageClientConServerThread {

private static HashMap hm=new HashMap<String, ClientConServerThread>();

//把创建好的ClientConServerThread放入到hm中

public static void addClientConServerThread(String qqId,ClientConServerThread ccst){

hm.put(qqId, ccst);

}

//可以通过qqId取得该线程

public static ClientConServerThread getClientConServerThread(String qqId){

return (ClientConServerThread)hm.get(qqId);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ManageQqChat.java]

/\*\*

\* 这是一个管理用户聊天界面的类

\*/

package com.qq.client.tools;

import java.util.HashMap;

import com.qq.client.view.QqChat;

public class ManageQqChat {

private static HashMap hm=new HashMap<String,QqChat>();

//加入

public static void addQqChat(String loginIdAndFriendId,QqChat qqChat){

hm.put(loginIdAndFriendId, qqChat);

}

//取出

public static QqChat getQqChat(String loginIdAndFriendId){

return (QqChat)hm.get(loginIdAndFriendId);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ManageQqFriendList.java]

/\*\*

\* 管理qq好友、陌生人、黑名单界面类

\*/

package com.qq.client.tools;

import java.util.HashMap;

import com.qq.client.view.QqFriendList;

public class ManageQqFriendList {

private static HashMap hm=new HashMap<String,QqFriendList>();

//加入

public static void addQqFriendList(String qqid,QqFriendList qqFriendList){

hm.put(qqid, qqFriendList);

}

//取出

public static QqFriendList getQqFriendList(String qqId){

return (QqFriendList)hm.get(qqId);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.qq.common

[Message.java]

/\*\*

\* 为Message规定规则

\*/

package com.qq.common;

import java.io.Serializable;

public class Message implements Serializable{

private String mesType;

private String sender;

private String getder;

private String con;

private String sendTime;

public String getGetder() {

return getder;

}

public void setGetder(String getder) {

this.getder = getder;

}

public String getSender() {

return sender;

}

public void setSender(String sender) {

this.sender = sender;

}

public String getCon() {

return con;

}

public void setCon(String con) {

this.con = con;

}

public String getSendTime() {

return sendTime;

}

public void setSendTime(String sendTime) {

this.sendTime = sendTime;

}

public String getMesType() {

return mesType;

}

public void setMesType(String mesType) {

this.mesType = mesType;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[MessageType.java]

/\*\*

\* 定义消息包的种类的接口

\*/

package com.qq.common;

public interface MessageType {

String message\_succeed="1";//登录成功包

String message\_login\_fail="2";//登录失败包

String message\_comm\_mes="3";//普通消息包

String message\_get\_onLineFriend="4";//要求在线的好友包

String message\_ret\_onLineFriend="5";//返回在线好友包

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[User.java]

/\*\*

\* 这是用户信息类

\*/

package com.qq.common;

import java.io.Serializable;

public class User implements Serializable{

private String userId;

private String passwd;

public String getUserId() {

return userId;

}

public void setUserId(String userId) {

this.userId = userId;

}

public String getPasswd() {

return passwd;

}

public void setPasswd(String passwd) {

this.passwd = passwd;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

至此，韩顺平JAVA入门到精通(全94讲)视频到此完结。

笔记包含完整的视频PPT内容，及所有演示示例代码，和视频中所有作业答题。

学习注意：

视频教学内容简单易学，容易上手。对于JAVASE的内容讲解通俗易懂，不过深度不高，讲的较为浅显。主要是让学员对JAVASE有所了解，本人对视频印象深刻的为在企业开发中的注意事项，及开发软件的规则。对于需要全面了解学习JAVA的朋友来说，还需研读更深、更全面的JAVA书籍，及JAVA编程思想。从而对JAVA有全面的认识。

韩顺平讲师的后续视频笔记，将陆续放出。