**Java程序设计**

**实验指导、报告书**

**班级：\_\_\_\_\_1504202\_\_\_\_\_\_\_\_\_**

**学号：\_\_\_\_\_140420227\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_侯捷\_\_\_\_\_\_\_\_\_\_**

**哈尔滨工业大学(威海)计算机科学与技术学院**

**目录**

[实验1 熟悉Java开发环境 3](#_Toc456686770)

[实验2 Java结构化程序设计 5](#_Toc456686771)

[实验3 数组和向量的使用 6](#_Toc456686772)

[实验4 类和对象的练习 8](#_Toc456686773)

[实验5 Java图形用户界面设计 11](#_Toc456686774)

[实验6 Java数据流 14](#_Toc456686775)

[实验7 Java多线程 18](#_Toc456686776)

[实验8 Java网络编程 21](#_Toc456686777)

# 实验1 熟悉Java开发环境

1. **实验目的**
2. 了解JDK及API文档的安装。
3. 了解Java应用程序的调试过程。
4. 正确使用Java命令行。
5. 熟悉Eclipse集成开发环境。
6. **实验内容**
7. 在自己使用的机器上安装JDK及Eclipse，并正确设置系统环境。
8. 编写一个Java应用程序，在屏幕上显示如下信息：Hello <你的姓名>。

Main.java

package com.company;

//2016年7月9日

//Java 实验1\_1

//侯捷

public class Main {

public static void main(String[] args) {

// write your code here

//直接输出

//System.out.println("hello, world! "+"<侯捷 140420227>");

//调用其他类输出

Demo\_1\_1 t = new Demo\_1\_1();

System.out.println(t.Print());

}

}

# 实验2 Java结构化程序设计

1. **实验目的**
2. 熟悉和理解Java中的数据类型、运算符、表达式、程序结构等内容。
3. 掌握常用的输入输出方法，编写、调试简单的程序。
4. 掌握选择、循环等语句用法。
5. 学会使用Java提供的类及方法。
6. **实验内容**
7. 在控制台输入正整数n，计算1/1+1/2+…+1/n的值并在控制台中输出。
8. 在控制台输入正整数n，计算1/1+1/3+1/5+…的前n项值并在控制台中输出，例如，n=4，计算1/1+1/3+1/5+1/7的值。
9. 在控制台输入正整数n，计算n！的值并在控制台中输出。
10. 在控制台输入两个整数m和n，求出它们之间的最大值并在控制台中输出。
11. 输入两个整数m和n，求出它们的最小公倍数和最大公约数并在控制台中输出。
12. 求出所有的水仙数并在控制台中输出，水仙数是三位数，它的各位数字的立方和等于这个数本身，例如，371=3\*3\*3+7\*7\*7+1\*1\*1。
13. 在控制台输入正整数n，求出从1到n的所有完全数并在控制台中输出，所谓完全数是指一个数，该数恰好等于它的因子之和，例如，6是一个完全数，因为6的因子为1、2和3，而6=1+2+3。
14. 在控制台输入n个整数，然后按照从小到大的顺序输出这些数。
15. 在控制台输入正整数n，求出1到n之间的所有素数，在控制台中输出这些素数，要求每行显示5个素数。
16. 在控制台输入正整数n，生成{1,2,…,n}的所有排列，在控制台中输出这些排列，每个排列中相邻两个数之间加一个空格，不同排列之间用分号“；”隔开，每行显示两个排列。

Demo 2\_1.java

**package** Demo2;  
  
**import** java.util.Scanner;  
**import** java.util.concurrent.SynchronousQueue;  
**import** java.util.concurrent.ThreadFactory;  
  
*/\*\*  
 \* Created by ·13 on 7/9/2016.  
 \*/***public class** Demo\_1\_2  
{  
 **protected** Scanner **s** =**new** Scanner(System.***in***);  
  
 *//获取正整数* **protected int** GetInt()  
 {  
 System.***out***.print(**"请输入一个正整数："**);  
 **int** tmp;  
 tmp = **s**.nextInt();  
 *//System.out.println("你输入的数字是："+temp+"\n");* **return** tmp;  
 }  
  
 *//计算 1/n 的前n项和* **protected void** test\_1()  
 {  
 **int** i;  
 **int** tmp;  
 tmp = **this**.GetInt();  
 **double** sum = 0;  
  
 **for**(i=1;i<=tmp;++i)  
 {  
 sum += 1.0/i;  
 *//System.out.println(sum);* }  
 System.***out***.print(sum);  
 }  
  
 *//计算 1/2n+1 的前n项和* **protected void** test\_2()  
 {  
 **int** i;  
 **int** tmp;  
 tmp = **this**.GetInt();  
 **double** tmp2 ;  
 **double** sum = 0;  
  
 **for**(i=1;i<=tmp;++i)  
 {  
 tmp2 = i\*2-1;  
 sum += 1.0/tmp2;  
 *//System.out.println(sum);* }  
 System.***out***.print(sum);  
 }  
  
 *//计算 n！* **protected void** test\_3()  
 {  
 **int** i;  
 **int** tmp;  
 tmp = **this**.GetInt();  
 **int** sum = 1;  
  
 **for**(i=1;i<=tmp;++i)  
 {  
 sum \*= i;  
 *//System.out.println(sum);* }  
 System.***out***.print(sum);  
 }  
  
 *//MAX* **protected void** test\_4()  
 {  
 **int** num1,num2;  
 num1 = **this**.GetInt();  
 num2 = **this**.GetInt();  
 **if** (num1>num2)  
 System.***out***.println(num1);  
 **else** System.***out***.println(num2);  
 }  
  
 *//最小公倍数，最大公约数* **protected void** test\_5()  
 {  
 **int** num1,num2;  
 num1 = **this**.GetInt();  
 num2 = **this**.GetInt();  
  
 **int** min;  
 **if** (num1>num2)  
 min = num2;  
 **else** min = num1;  
  
 *//公约数* **int** i,tmp = -1;  
 **for**(i=min;i>0;--i)  
 {  
 **if**(num1 % i == 0)  
 **if**(num2 % i == 0)  
 {  
 tmp = i;  
 **break**;  
 }  
 }  
  
 System.***out***.println(**"两数的最大公约数是："**+tmp);  
  
 **int** j;  
 **for**(j=min;**true**;j++)  
 {  
 **if** (j%num1 == 0)  
 **if**(j%num2 == 0)  
 {  
 System.***out***.println(**"两数的最小公倍数是："**+j);  
 **break**;  
 }  
 }  
 }  
  
 *//水仙花数* **protected void** test\_6()  
 {  
 *//获取计算上限* **int** limit;  
 limit = **this**.GetInt();  
 System.***out***.println(**"此整数范围内的水仙花数如下："**);  
 **int** i;  
 **for**(i=1;i<=limit;++i)  
 {  
 **if** (**this**.daffodilsNum(i))  
 System.***out***.println(i);  
 }  
  
  
 }  
 **protected boolean** daffodilsNum(**int** testNum)  
 {  
 **int** sum = 0;  
 **int** num = -1,tmp1,tmp2;  
 num = testNum;  
 tmp2 = testNum;  
 **for**(;**true**;)  
 {  
 *//存储末尾数字* tmp1 = tmp2%10;  
 sum += (tmp1\*tmp1\*tmp1);  
 *//数字向右移动一位* tmp2 /= 10;  
  
 **if** (tmp2<10)  
 {  
 sum += (tmp2\*tmp2\*tmp2);  
 **break**;  
 }  
 }  
 **if** (sum == num)  
 **return true**;  
 **else  
 return false**;  
 }  
  
 *//完全数* **protected void** test\_7()  
 {  
 *//获取计算上限* **int** limit;  
 limit = **this**.GetInt();  
 System.***out***.println(**"此整数范围内的完全数如下："**);  
 **int** i;  
 **for**(i=1;i<=limit;++i)  
 {  
 **if** (**this**.PerfectNum(i))  
 System.***out***.println(i);  
 }  
 }  
 **protected boolean** PerfectNum(**int** testNum)  
 {  
 **int** i;  
 **int** sum = (0 + 1); *//每个数字都有1因子* **int** num = -1, tmp1,tmp2;  
 num = testNum;  
 tmp2 = testNum;  
 **for**(i=2;**true**;)  
 {  
 *//System.out.println("\_8\_0\_test");  
 //System.out.println(tmp2);* **if**(tmp2%i == 0)  
 {  
 sum += i;  
 tmp2 /= i;  
 i=2;  
 }  
 **else if** (tmp2 == 1)  
 **break**;  
 **else** ++i;  
 }  
 **if** (sum == num)  
 **return true**;  
 **else  
 return false**;  
 }  
  
 *//排序* **protected void** test\_8()  
 {  
 *//获取计算上限* **int** numLimit;  
 numLimit = **this**.GetInt();  
 System.***out***.println(**"请输入"**+numLimit+**"个整数"**);  
  
 **int** numArray[] =**new int**[numLimit];  
 **int** i;  
 **for**(i=0;i<numLimit;++i)  
 {  
 numArray[i] = **this**.GetInt();  
 }  
  
 *//冒泡排序* **int** tmp;  
 **int** l,m;  
 **for**(l=0;l<numArray.**length**-1;++l)  
 **for**(m=0;m<numArray.**length**-1;++m)  
 {  
 **if**(numArray[m]>numArray[m+1])  
 {  
 tmp = numArray[m];  
 numArray[m] = numArray[m+1];  
 numArray[m+1] = tmp;  
 }  
 }  
  
 **for**(i=0;i<numArray.**length**;++i)  
 {  
 System.***out***.print(numArray[i]);  
 System.***out***.print(**" "**);  
 }  
  
  
 }  
  
 *//输出素数* **protected void** test\_9()  
 {  
 {  
 *//获取计算上限* **int** limit;  
 limit = **this**.GetInt();  
 System.***out***.println(**"此整数范围内的素数如下："**);  
 **int** i;  
 **int** count=0;*//用于计数换行* **for**(i=1;i<=limit;++i)  
 {  
 **if** (**this**.primeNum(i))  
 {  
 **if** (count > 4) {  
 System.***out***.print(**"\n"**);  
 *//System.out.print("count=" + count+ " ");* count = 1;  
 } **else** count++;  
 System.***out***.print(i);  
 System.***out***.print(**" "**);  
 }  
 }  
 }  
 }  
 **protected boolean** primeNum(**int** testNum)  
 {  
 **int** i;  
 **for**(i=2;i<=testNum/2;i++)  
 {  
 **if** (testNum%i == 0)  
 **return false**;  
 }  
 **return true**;  
 }  
  
 *//输出排列* **protected void** test\_10()  
 {  
 *//获取计算上限* **int** limit;  
 limit = **this**.GetInt();  
 System.***out***.println(**"此整数范围内的所有排列如下："**);  
  
 *//数组初始化及其赋值* **int** arr[] = **new int**[limit];  
 **int** i;  
 **for**(i=1;i<=limit;++i)  
 {  
 arr[i-1] = i;  
 }  
  
 *//用递归的思路编写代码：  
 //N个数全排列 = N-1个数全排列  
 //再将取出来的那个数字放置于最前方* **this**.FullArray(arr,0,arr.**length**);  
 }  
 **public boolean sw** = **true**;  
 *//测试函数，用来输出数组* **protected void** printTest(**int** arr[])  
 {  
 **int** num = arr.**length**;  
 **int** i;  
 **for**(i=0;i<num;i++)  
 {  
 System.***out***.print(arr[i]+**" "**);  
 }  
 **if**(**sw**)  
 {  
 System.***out***.print(**";"**);  
 **sw** = **false**;  
 }  
 **else** {  
 System.***out***.print(**"\n"**);  
 **sw** = **true**;  
 }  
 *//System.out.println();* }  
 *//交换函数(数组，第一个索引，第二个索引)* **protected int**[] swap(**int** arr[],**int** index1,**int** index2)  
 {  
 **int** num = arr.**length**;  
 **int** tmp[] = **new int**[num];  
 tmp = arr;  
 **int** tmpNum;  
 tmpNum = tmp[index1];  
 tmp[index1] = tmp[index2];  
 tmp[index2] = tmpNum;  
 **return** tmp;  
 }  
  
 *//全排列函数* **protected void** FullArray(**int** arr[],**int** st,**int** len)  
 {  
 **int**[] tmp;  
 tmp = arr;  
 **if**(st == len-1)  
 {  
 **this**.printTest(tmp);  
 }  
 **else** {  
 **for**(**int** i=st;i<len;i++)  
 {  
 **this**.swap(tmp,st,i);  
 FullArray(tmp,st+1,len);  
 **this**.swap(tmp,st,i);  
 }  
 }  
 }  
}

# 实验3 数组和向量的使用

1. **实验目的**
2. 熟练掌握数组的定义、初始化和使用方法。
3. 掌握向量的使用方法，如插入、删除和查找等功能。
4. 利用数组和向量来实现一个简单的应用系统。
5. **实验内容**
6. 定义一个一维数组，其中存储随机生成的1000个1至100以内的整数，统计每个整数出现的次数。
7. 设计一个矩阵乘法程序，输入两个矩阵可以计算出它们的乘积。
8. 使用向量保存图书信息，包括书名、作者、出版社、刊号、出版日期、页数和摘要，并能够实现图书的插入、删除、查找功能。插入、删除时要显示操作是否正确与否的提示信息；查找时按关键字值进行查找，并显示查找结果。
9. 设计一个简单的学生成绩管理系统，要求能够录入并存储学生信息、课程信息以及每个学生每门课程的成绩。学生信息包括姓名、学号、性别和班级，课程信息包括课程号、课程名和学分。系统的功能包括：计算每个学生的总成绩、平均成绩和总学分，计算每个班级的平均成绩，按照学生平均成绩由高到低的顺序进行排序，统计所有学生每门课程的平均成绩。

实验1

**package** Demo2\_3;  
**import** java.util.Vector;  
**import** java.lang.Math;  
  
*/\*\*  
 \* Created by hj on 16-7-14.  
 \* 定义一个一维数组，其中存储随机生成的1000个1至100以内的整数，统计每个整数出现的次数。  
 \*/***public class** Demo\_2\_3\_1  
{  
 **private** Vector<Integer> **NumVector** = **new** Vector<>(100, 50);  
  
 **public** Demo\_2\_3\_1()  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **int** i,tmp;  
 **for**(i=0;i<1000;i++)  
 {  
 tmp = (**int**)(Math.*random*()\*100+1);  
 **NumVector**.addElement(tmp);  
 }  
 **this**.countVector();  
 }  
  
 **private void** countVector()  
 {  
 */\*\*  
 \* 统计向量中每个元素的个数  
 \*/* **int** i,j;  
 **int** count;  
 **for**(i=1;i<=100;i++)  
 {  
 count = 0;  
 **for**(j=0;j<1000;j++)  
 {  
 **if**(**NumVector**.elementAt(j) == i)  
 count ++;  
 }  
 System.***out***.println(i+**" : "**+count);  
 }  
 }  
   
}

实验2

**package** Demo2\_3;  
  
**import org.jetbrains.annotations.Nullable**;  
  
**import** java.util.Scanner;  
  
*/\*\*  
 \* Created by hj on 16-7-14.  
 \* 设计一个矩阵乘法程序，输入两个矩阵可以计算出它们的乘积。  
 \*/***public class** Demo\_2\_3\_2 {  
 **private int matrix1**[][];  
 **private int matrix2**[][];  
 **private int result\_matrix**[][];  
  
 **public** Demo\_2\_3\_2() {  
 */\*\*  
 \* 构造函数  
 \*/* **int** i, j, tmp;  
 **int** ColumnNum, RowNum;  
 Scanner in = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"请输入第一个矩阵的行数:"**);  
 RowNum = in.nextInt();  
 System.***out***.println(**"请输入第一个矩阵的列数:"**);  
 ColumnNum = in.nextInt();  
 **matrix1** = **new int**[RowNum][ColumnNum];  
 **for** (i = 1; i <= RowNum; i++)  
 **for** (j = 1; j <= ColumnNum; j++) {  
 System.***out***.println(**"请输入矩阵第"** + i + **"行第"** + j + **"列的数字"**);  
 tmp = in.nextInt();  
 **matrix1**[i - 1][j - 1] = tmp;  
 }  
  
 System.***out***.println(**"请输入第二个矩阵的行数:"**);  
 RowNum = in.nextInt();  
 System.***out***.println(**"请输入第二个矩阵的列数:"**);  
 ColumnNum = in.nextInt();  
 **matrix2** = **new int**[RowNum][ColumnNum];  
 **for** (i = 1; i <= RowNum; i++)  
 **for** (j = 1; j <= ColumnNum; j++) {  
 System.***out***.println(**"请输入矩阵第"** + i + **"行第"** + j + **"列的数字"**);  
 tmp = in.nextInt();  
 **matrix2**[i - 1][j - 1] = tmp;  
 }  
 *//this.printTest(matrix1);  
 //this.printTest(matrix2);* **result\_matrix** = **this**.matrixMultiplication(**matrix1**, **matrix2**);  
 **this**.printTest(**result\_matrix**);  
 }  
  
 **private void** printTest(**int** arry[][]) {  
 */\*\*  
 \* 测试输出一个矩阵  
 \*/* System.***out***.println(**"--------result--------"**);  
 **int** i, j;  
 **int** row, col;  
 row = arry.**length**;  
 col = arry[0].**length**;  
 **for** (i = 0; i < row; i++) {  
 **for** (j = 0; j < col; j++)  
 System.***out***.print(arry[i][j] + **" "**);  
 System.***out***.println();  
 }  
 System.***out***.println();  
 }  
  
 **@Nullable  
 private int**[][] matrixMultiplication(**int** arry1[][], **int** arry2[][]) {  
 */\*\*  
 \* 矩阵乘法  
 \** ***@parm1：矩阵1*** *\** ***@parm2：矩阵2*** *\** ***@return：矩阵1和2相乘的结果，若无法相乘返回null*** *\*/* **int** row1, col1, row2, col2;  
 row1 = arry1.**length**;  
 col1 = arry1[0].**length**;  
 row2 = arry2.**length**;  
 col2 = arry2[0].**length**;  
  
 **if** (col1 != row2) {  
 System.***out***.println(**"Error：输入的两个矩阵无法相乘"**);  
 **return null**;  
 }  
  
 **result\_matrix** = **new int**[row1][col2];  
 **int** tmp;  
 **int** i, j, l;  
 **for** (i = 0; i < row1; i++)  
 **for** (j = 0; j < col2; j++)  
 {  
 tmp = 0;  
 **for** (l = 0; l < col1; l++)  
 *//System.out.println(matrix1[i][l] + "\*" + matrix2[l][j]);* tmp += **matrix1**[i][l] \* **matrix2**[l][j];  
 **result\_matrix**[i][j] = tmp;  
 }  
 **return result\_matrix**;  
 }  
}

实验3

**package** Demo2\_3;  
  
**import** java.util.Scanner;  
  
*/\*\*  
 \* Created by hj on 16-7-14.  
 \* 图书信息类  
 \* 书名、作者、出版社、刊号、出版日期、页数和摘要  
 \*/***class** Book\_Info {  
 String **bookname**;  
 String **author**;  
 String **press**;  
 String **ISBN**;  
 String **publishData**;  
 String **pageNum**;  
 String **bookAbstrac**;  
  
 Book\_Info(String arg1,String arg2,String arg3,String arg4,String arg5,String arg6,String arg7)  
 {  
 */\*\*  
 \* /。， 0  
 \* 构造函数-1  
 \*/* **bookname** = arg1;  
 **author** = arg2;  
 **press** = arg3;  
 **ISBN** = arg4;  
 **publishData** = arg5;  
 **pageNum** = arg6;  
 **bookAbstrac** = arg7;  
 }  
  
 Book\_Info()  
 {  
 */\*\*  
 \* 构造函数-2  
 \*/* System.***out***.println(**"请按屏幕提示信息，输入书籍对应的信息"**);  
 System.***out***.print(**"请输入书籍的书名："**);  
 Scanner in = **new** Scanner(System.***in***);  
 **bookname** = in.nextLine();  
 *//System.out.println(bookname);* System.***out***.print(**"请输入书籍的作者："**);  
 **author** = in.nextLine();  
 System.***out***.print(**"请输入书籍的出版社："**);  
 **press** = in.nextLine();  
 System.***out***.print(**"请输入书籍的刊号："**);  
 **ISBN** = in.nextLine();  
 System.***out***.print(**"请输入书籍的出版日期："**);  
 **publishData** = in.nextLine();  
 System.***out***.print(**"请输入书籍的页数："**);  
 **pageNum** = in.nextLine();  
 System.***out***.print(**"请输入书籍的摘要："**);  
 **bookAbstrac** = in.nextLine();  
 }  
  
 **void** displayInfo()  
 {  
 System.***out***.println(**"---图书信息---"**);  
 System.***out***.println(**"书名\t："**+**this**.**bookname**);  
 System.***out***.println(**"作者\t："**+**this**.**author**);  
 System.***out***.println(**"出版社\t："**+**this**.**press**);  
 System.***out***.println(**"刊号\t："**+**this**.**ISBN**);  
 System.***out***.println(**"出版日期\t："**+**this**.**publishData**);  
 System.***out***.println(**"页数\t："**+**this**.**pageNum**);  
 System.***out***.println(**"摘要\t："**+**this**.**bookAbstrac**);  
 System.***out***.println();  
 }  
}

**package** Demo2\_3;  
  
**import** java.util.Scanner;  
**import** java.util.Vector;  
  
*/\*\*  
 \* Created by hj on 16-7-14.  
 \* 使用向量保存图书信息，包括书名、作者、出版社、刊号、出版日期、页数和摘要，  
 \* 并能够实现图书的插入、删除、查找功能。插入、删除时要显示操作是否正确与否的提示信息；  
 \* 查找时按关键字值进行查找，并显示查找结果。  
 \*/***class** Demo\_2\_3\_3 {  
 **private** Book\_Info **tmp**;  
 **private** Vector<Book\_Info> **BookVector**;  
 **private** Scanner **in** = **new** Scanner(System.***in***);  
  
 Demo\_2\_3\_3()  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **BookVector** = **new** Vector<>(20, 1);  
 System.***out***.println(**"请输入初始化时图书信息数目"**);  
 **int** booknum = **in**.nextInt();  
 **int** i;  
 **for**(i=0;i<booknum;i++)  
 **this**.insert();  
 }  
  
 **private void** insert()  
 {  
 */\*\*  
 \* 插入  
 \*/* **tmp** = **new** Book\_Info();  
 **BookVector**.addElement(**tmp**);  
 }  
  
 **private void** selete()  
 {  
 */\*\*  
 \* 查询  
 \*/* System.***out***.println(**"请输入要查询的书目信息(模糊搜索)："**);  
 String search\_info = **in**.next();  
 **int** VectorSize = **BookVector**.size();  
 **int** i;  
 **for**(i=0;i<VectorSize;i++)  
 {  
 *//从Vector中取出的是Object类型变量，需要转换成所需类型* **tmp** = **BookVector**.elementAt(i);  
 **if** (**this**.infoContain(**tmp**,search\_info))  
 **tmp**.displayInfo();  
 }  
 }  
  
 **private boolean** infoContain(Book\_Info book, String search) {  
 **return** book.**bookname**.contains(search) ||  
 book.**author**.contains(search) ||  
 book.**press**.contains(search) ||  
 book.**ISBN**.contains(search) ||  
 book.**publishData**.contains(search) ||  
 book.**pageNum**.contains(search) ||  
 book.**bookAbstrac**.contains(search);  
 }  
  
 **void** delete()  
 {  
 */\*\*  
 \* 删除  
 \*/* System.***out***.println(**"请输入要删除书籍的ISBN号（刊号）。"**);  
 String ISBN = **in**.next();  
 **boolean** opreationFlag = **true**;  
  
 *//=============可以重构的部分===============* **int** VectorSize = **BookVector**.size();  
 **int** i;  
 **for**(i=0;i<VectorSize;i++)  
 {  
 *//从Vector中取出的是Object类型变量，需要转换成所需类型* **tmp** = **BookVector**.elementAt(i);  
 **if**(**tmp**.**ISBN**.contains(ISBN))  
 {  
 opreationFlag = **false**;  
 System.***out***.println(**"您要删除的书籍信息为："**);  
 **tmp**.displayInfo();  
  
 System.***out***.println(**"确认删除吗：？（Y/N）"**);  
 String answer = **in**.next();  
 **switch** (answer) {  
 **case "Y"**:  
 **case "y"**:  
 **BookVector**.remove(i);  
 **break**;  
 **case "N"**:  
 **case "n"**:  
 System.***out***.println(**"取消删除"**);  
 **break**;  
 **default**:  
 System.***out***.println(**"输入了错误的信息，请重新操作。"**);  
 **break**;  
 }  
 }  
 }  
 **if**(opreationFlag)  
 System.***out***.println(**"未查询到对应的书籍信息"**);  
  
 }  
  
  
}

实验4

**package** Demo2\_3;  
  
**import** java.util.Scanner;  
  
*/\*\*  
 \* Created by hj on 16-7-19.  
 \* 课程信息包括 课程号、  
 \* 课程名 和  
 \* 学分。  
 \*/***class** Courses\_Info {  
 **protected** String **coursesNum**;  
 **protected** String **coursesName**;  
 **protected double grade**;  
 **protected double credit**;  
  
 Courses\_Info()  
 {  
 */\*\*  
 \* 构造函数  
 \*/* System.***out***.println(**"请按此输入这门课的相应信息："**);  
 Scanner in = **new** Scanner(System.***in***);  
 System.***out***.print(**"这门课的课程号："**);  
 **coursesNum** = in.nextLine();  
 System.***out***.print(**"这门课的课程名："**);  
 **coursesName** = in.nextLine();  
 System.***out***.print(**"这门课的成绩："**);  
 **grade** = in.nextDouble();  
 System.***out***.print(**"这门课的学分："**);  
 **credit** = in.nextDouble();  
 }  
  
 **void** show\_detail()  
 {  
 *//System.out.println("课程信息");* System.***out***.println(**" 课程名："**+**coursesName**+**"\t成绩："**+**grade**+**"\t学分："**+**credit**);  
 }  
  
}

**package** Demo2\_3;  
  
**import** com.sun.org.glassfish.external.statistics.AverageRangeStatistic;  
  
**import** java.util.Scanner;  
**import** java.util.Vector;  
  
*/\*\*  
 \* Created by hj on 16-7-19.  
 \* 学生信息包括 姓名、  
 \* 学号、  
 \* 性别 和  
 \* 班级，  
 \* 课程信息  
 \*/***class** Stu\_Info {  
 **protected** String **stuName**;  
 **protected** String **stuNum**;  
 **protected** String **sex**;  
 **protected** String **classNum**;  
 **protected** Vector<Courses\_Info> **coursesVector**;  
 **protected double averagGrade**;  
 **protected double totalGrade**;  
 **protected double totalCredit**;  
  
 Stu\_Info()  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **coursesVector** = **new** Vector<>(5,1);  
 System.***out***.println(**"请按此输入这个学生的相应信息："**);  
 Scanner in = **new** Scanner(System.***in***);  
 System.***out***.print(**"学生姓名："**);  
 **stuName** = in.nextLine();  
 System.***out***.print(**"学生学号："**);  
 **stuNum** = in.nextLine();  
 System.***out***.print(**"学生性别："**);  
 **sex** = in.nextLine();  
 System.***out***.print(**"学生班级："**);  
 **classNum** = in.nextLine();  
  
 System.***out***.println(**"请输入学生目前的课程数目："**);  
 **int** coursesNum = in.nextInt();  
 **int** i;  
 **for**(i=0;i<coursesNum;i++)  
 {  
 Courses\_Info tmp = **new** Courses\_Info();  
 **coursesVector**.addElement(tmp);  
 }  
 **totalCredit** = getTotalGrade(**coursesVector**);  
 **averagGrade** = getAveGrade(**coursesVector**);  
 **totalGrade** = getTotalGrade(**coursesVector**);  
 }  
 **private double** getTotalCredit(Vector<Courses\_Info> coursesVector)  
 {  
 */\*\*  
 \* 获得总学分  
 \** ***@parm:课程容器*** *\** ***@return:总学分*** *\*/* **int** VectorSize = coursesVector.size();  
 **int** i;  
 **double** totalCredit = 0;  
 Courses\_Info tmp;  
 **for** (i = 0; i < VectorSize; i++)  
 {  
 tmp = coursesVector.elementAt(i);  
 totalCredit += tmp.**credit**;  
 }  
 **return** totalCredit;  
 }  
 **private double** getAveGrade(Vector<Courses\_Info> coursesVector)  
 {  
 */\*\*  
 \* 获得平均成绩  
 \** ***@parm:课程容器*** *\** ***@return:平均成绩*** *（总成绩/总学分）  
 \*/* **double** totalGrade = getTotalGrade(coursesVector);  
 **double** totalCredit = getTotalCredit(coursesVector);  
  
 **return** totalGrade/totalCredit;  
 }  
  
 **private double** getTotalGrade(Vector<Courses\_Info> coursesVector)  
 {  
 */\*\*  
 \* 获得总成绩  
 \** ***@parm:课程容器*** *\** ***@return:总成绩*** *(成绩*\**学分)  
 \*/* **int** VectorSize = coursesVector.size();  
 **int** i;  
 **double** totalGrade = 0;  
 Courses\_Info tmp;  
 **for** (i = 0; i < VectorSize; i++)  
 {  
 tmp = coursesVector.elementAt(i);  
 totalGrade += tmp.**grade** \* tmp.**credit**;  
 }  
 **return** totalGrade;  
 }  
  
 **void** show\_detail()  
 {  
 */\*\*  
 \* 展示一个学生的详细信息  
 \*/* System.***out***.println(**"该学生的详细信息如下："**);  
 System.***out***.println(**"姓名："**+**stuName**+**"\t学号："**+**stuNum**+**"\t班级："**+**classNum**+**"\t性别："**+**sex**);  
 System.***out***.println(**"各项成绩："**);  
 **int** coursesVectorSize = **coursesVector**.size();  
 Courses\_Info tmp ;  
 **int** i;  
 **for**(i=0;i<coursesVectorSize;i++)  
 {  
 tmp = **coursesVector**.elementAt(i);  
 tmp.show\_detail();  
 }  
 }  
}

**package** Demo2\_3;  
  
**import** java.util.Scanner;  
**import** java.util.Vector;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Created by hj on 16-7-19.  
 \* 设计一个简单的学生成绩管理系统，  
 \* 要求能够录入并存储学生信息、课程信息以及每个学生每门课程的成绩。  
 \* 学生信息包括姓名、学号、性别和班级，课程信息包括课程号、课程名和学分。  
 \*  
 \* 系统的功能包括：  
 \* 计算每个学生的总成绩、  
 \* 平均成绩和总学分，  
 \* 计算每个班级的平均成绩，  
 \* 按照学生平均成绩由高到低的顺序进行排序，  
 \* 统计所有学生每门课程的平均成绩。  
 \*/***class** Demo\_2\_3\_4 {  
  
 **private** Vector<Stu\_Info> **StuVector**;  
  
 Demo\_2\_3\_4()  
 {  
 */\*\*  
 构造函数  
 \*/* **StuVector** = **new** Vector<>(10,1);  
 System.***out***.println(**"请输入初始化是需要输入的学生数目："**);  
 Scanner in = **new** Scanner(System.***in***);  
 **int** stusNum = in.nextInt();  
 **int** i;  
 **for**(i=0;i<stusNum;i++)  
 {  
 Stu\_Info tmp = **new** Stu\_Info();  
 **StuVector**.addElement(tmp);  
 }  
 }  
  
 **void** show\_averagGrade()  
 {  
 */\*\*  
 展示平均成绩  
 \*/* **int** StusNum = **StuVector**.size();  
 **int** i;  
 Stu\_Info tmp;  
 **for**(i=0;i<StusNum;++i)  
 {  
 tmp = **StuVector**.elementAt(i);  
 System.***out***.println(**"姓名："**+tmp.**stuNum**+**"\t平均成绩(加权)："**+tmp.**averagGrade**);  
 }  
 }  
  
 **void** show\_totalCredit()  
 {  
 */\*\*  
 展示总学分  
 \*/* **int** StusNum = **StuVector**.size();  
 **int** i;  
 Stu\_Info tmp;  
 **for**(i=0;i<StusNum;++i)  
 {  
 tmp = **StuVector**.elementAt(i);  
 System.***out***.println(**"姓名："**+tmp.**stuNum**+**"\t总学分："**+tmp.**totalCredit**);  
 }  
 }  
  
 **void** show\_totalGrade()  
 {  
 */\*\*  
 展示总成绩  
 \*/* **int** StusNum = **StuVector**.size();  
 **int** i;  
 Stu\_Info tmp;  
 **for**(i=0;i<StusNum;++i)  
 {  
 tmp = **StuVector**.elementAt(i);  
 System.***out***.println(**"姓名："**+tmp.**stuNum**+**"\t总成绩： "**+tmp.**totalGrade**);  
 }  
 }  
  
 **void** getClassAveragGrade(String classNum)  
 {  
 */\*\** ***@parm*** *班号  
 展示班级平均成绩  
 \*/* **int** stusNum = **StuVector**.size();  
 **int** i;  
 **double** classStuNum = 0,classToallGrade = 0,classAveragGrade;  
 Stu\_Info tmp;  
 **for**(i=0;i<stusNum;++i)  
 {  
 tmp = **StuVector**.elementAt(i);  
 **if** (tmp.**classNum**.equals(classNum))  
 {  
 classToallGrade += tmp.**averagGrade**;  
 classStuNum += 1;  
 }  
 }  
 **if** (classStuNum == 0)  
 {  
 System.***out***.println(**"未查询此班号"**);  
 **return** ;  
 }  
 classAveragGrade = classToallGrade/classStuNum;  
 System.***out***.println(**"班级"**+classNum+**"的平均成绩: "**+classAveragGrade);  
 }  
  
 **void** getCoursesGrade(String coursesNum)  
 {  
 */\*\** ***@parm*** *课序号  
 展示班级平均成绩  
 \*/* **int** stusNum = **StuVector**.size(),courNUM;  
 **int** i,j;  
 **double** coursesStuNum = 0,coursesToallGrade = 0,coursesAveragGrade = 0;  
 Stu\_Info tmp;  
 Courses\_Info tmp2;  
 **for**(i=0;i<stusNum;++i)  
 {  
 tmp = **StuVector**.elementAt(i);  
 courNUM = tmp.**coursesVector**.size();  
 **for**(j=0;j<courNUM;++j)  
 {  
 tmp2 = tmp.**coursesVector**.elementAt(j);  
 **if**(tmp2.**coursesNum**.equals(coursesNum))  
 {  
 coursesToallGrade += tmp2.**grade**;  
 coursesStuNum += 1;  
 **break**;  
 }  
 }  
 }  
 **if** (coursesStuNum == 0)  
 {  
 System.***out***.println(**"未查询此课序号"**);  
 **return** ;  
 }  
 coursesAveragGrade = coursesToallGrade/coursesStuNum;  
 System.***out***.println(**"课程"**+coursesNum+**"的平均成绩: "**+coursesAveragGrade);  
 }  
  
 **private** List<String> getAllCoursesNum()  
 {  
 */\*\** ***@Return*** *所有课序号构成的容器  
 获取不重复的所有课序号  
 \*/* List<String> coursList = **new** ArrayList<>();  
 *//Vector<String> coursVector = new Vector<>(10,5);* **int** stusNum = **StuVector**.size() , courNUM;  
 **int** i,j;  
 Stu\_Info tmp;  
 Courses\_Info tmp2;  
 **for**(i=0;i<stusNum;++i)  
 {  
 tmp = **StuVector**.elementAt(i);  
 courNUM = tmp.**coursesVector**.size();  
 **for**(j=0;j<courNUM;++j)  
 {  
 tmp2 = tmp.**coursesVector**.elementAt(j);  
 *//添加不重复的课程列表* **if** (!coursList.contains(tmp2.**coursesNum**))  
 coursList.add(tmp2.**coursesNum**);  
 }  
 }  
 **return** coursList;  
 }  
  
 **void** show\_all\_courses\_averageGrade()  
 {  
 List<String> coursList = **new** ArrayList<>();  
 coursList = getAllCoursesNum();  
 **int** size = coursList.size();  
 **int** i;  
 **for**(i=0;i<size;++i)  
 {  
 getCoursesGrade(coursList.get(i));  
 }  
 }  
  
 **void** sort\_by\_averaneGrade()  
 {  
 **int** vectorSize = **StuVector**.size();  
 **int** i,j;  
 Stu\_Info tmp;  
 *//简单冒泡排序* **for**(i=0;i<vectorSize-1;++i)  
 {  
 **for**(j=0;j<vectorSize-1;++j)  
 {  
 **if** ((**StuVector**.elementAt(j).**averagGrade**<**StuVector**.elementAt(j+1).**averagGrade**))  
 {  
 tmp = **StuVector**.elementAt(j);  
 **StuVector**.setElementAt(**StuVector**.elementAt(j+1),j);  
 **StuVector**.setElementAt(tmp,j+1);  
 }  
 }  
 }  
 System.***out***.println(**"按平均成绩排序如下："**);  
 show\_averagGrade();  
 }  
  
}

# 实验4 类和对象的练习

1. **实验目的**
2. 熟悉和理解Java中的类、方法等概念，掌握类的定义和使用，能够在程序中定义自己使用的类，编写构造方法及成员方法。
3. 能够创建类的实例，掌握对象的声明和不同访问属性的成员访问方式，正确调用类的方法。
4. 深入理解Java中的继承概念，掌握继承的使用方法。
5. **实验内容**
6. 编写一个Java应用程序，该程序中有三个类：Triangle、Trapezoid和Circle，分别用来刻画“三角形”、“梯形”和“圆形”。具体要求如下：
7. Triangle类具有类型为double的三个边，以及周长、面积属性，Triangle类具有返回周长、面积以及修改三个边的功能。此外，Triangle类还有一个boolean型的属性，该属性用来判断三个数能够构成一个三角形。完成上述操作后，给Triangle再增加3个方法，分别用来返回3个边sideA, sideB, sideC的值。
8. Lander类具有类型为double的上底、下底、高、面积属性，具有返回面积的功能。
9. Circle类具有类型为double的半径、周长和面积属性，具有返回周长、面积的功能。
10. 编写一个测试类Test，分别创建类Triangle、Trapezoid和Circle的对象，并计算三角形、梯形和圆形的周长和面积。
11. 画出该应用程序的类图。
12. 创建一个名为pay的类，该类包括每小时价格(price)、扣缴率(withhold\_rate)、工资率(pay\_rate)和应得工资 (payment) 等4个double型成员变量。创建3个重载方法computePayment()来计算应得工资，计算规则如下：
13. 当computePayment ()接收小时、扣缴率和工资率三个参数值时，计算应得工资=工作小时\*每小时工资\*(1−扣缴率)\*(1−工资率)；
14. 当computePayment()接收小时和工资率两个参数时，扣缴率为15%，计算应得工资=工作小时\*每小时工资\*(1−0.15)\*(1−工资率)；
15. 当computePayment ()接收一个参数时，扣缴率为15%，每小时工资率为4.65%，计算应得工资=工作小时\*每小时工资\*(1−0.15)\*(1−0.0465)。

编写一个测试类Test，该测试类的main方法测试所有3个重载的方法。

1. 创建一个名为Telephone的类，Telephone类中包含有电话品牌(brand)、电话号码(number)、通话时间(dialedTime)、费率(rate)等属性，以及计算话费(callCost)和显示话费信息(display)方法。再创建一个名为Mobilephone的类，它是Telephone的子类，除了具有Telephone类的属性外，它还有自己的属性如网络类型(network)、被叫时间(receivedTime)，同时它有自己的计算话费和显示信息的方法。另外，程序中还有一个主类PhoneCost来使用上述两个类并显示它们的信息。

电话话费= dialedTime\*rate；

电话费率rate=0.2元/分钟；

移动电话话费=(dialedTime+0.5\*receivedTime)\*rate；

移动电话费率rate=0.4元/分钟

**程序运行的效果如下：**

电话品牌：TCL

电话号码：5687506

通话时间：130.0分钟

费率： 0.2元/分钟

花费总计：36.0元

电话品牌：SAMSUNG

电话号码：15163155787

网络： 3G

主叫时间：80.0 分钟

被叫时间：120.0分钟

费率： 0.4元/分钟

花费总计：56.0元

**package** Demo4;  
  
**import** java.util.Scanner;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
 1)Triangle类具有类型为double的三个边，以及周长、面积属性，  
  
 Triangle类具有返回周长、面积以及修改三个边的功能。此外，  
 Triangle类还有一个boolean型的属性，该属性用来判断三个数能够构成一个三角形。完成上述操作后，  
 给Triangle再增加3个方法，分别用来返回3个边sideA, sideB, sideC的值。  
 \*/***public class** Triangle {  
 **private double sideA**;  
 **private double sideB**;  
 **private double sideC**;  
 **private double perimeter**;  
 **private double square**;  
 **private boolean isTriangle**;  
  
 Triangle(**double** A, **double** B, **double** C)  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **sideA** = A;  
 **sideB** = B;  
 **sideC** = C;  
 init();  
 }  
  
 **private void** init()  
 {  
 */\*\*  
 \* 成员变量的计算  
 \*/* **perimeter** = **sideA** + **sideB** + **sideC**;  
 *//半周长* **double** p = **perimeter**/2;  
 *// 海伦公式* **square** = Math.*sqrt*(p\*(p-**sideA**)\*(p-**sideB**)\*(p-**sideC**));  
  
 **double** max, mid, min, tmp;  
 tmp = Math.*max*(**sideA**, **sideB**);  
 max = Math.*max*(tmp, **sideC**);  
 tmp = Math.*min*(**sideA**, **sideB**);  
 min = Math.*min*(tmp, **sideC**);  
 mid = **perimeter** - max - min;  
 **isTriangle** = (max < mid + min) && (min > max - min);  
 }  
  
  
 *// 返回周长* **protected double** getPerimeter(){**return perimeter**;}  
  
 *// 返回面积* **protected double** getSquare(){**return square**;}  
  
 **protected void** changeSideLength()  
 {  
 */\*\*  
 \* 修改三个边  
 \*/* **double** newSideA, newSideB, newSideC;  
 Scanner in = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入需要修改的三边长度：(双精度)"**);  
 System.***out***.print(**"A:"**);  
 newSideA = in.nextDouble();  
 System.***out***.print(**"B:"**);  
 newSideB = in.nextDouble();  
 System.***out***.print(**"C:"**);  
 newSideC = in.nextDouble();  
  
 **sideA** = newSideA;  
 **sideB** = newSideB;  
 **sideC** = newSideC;  
 init();  
 }  
  
}

**package** Demo4;  
  
**import** jdk.internal.dynalink.beans.StaticClass;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
 \*  
 Lander类具有类型为double的上底、下底、高、面积属性，  
 具有返回面积的功能。  
 \*/***public class** Lander {  
 **private double upperBottom**;  
 **private double lowerBottom**;  
 **private double height**;  
 **private double square**;  
  
  
 Lander(**double** ub, **double** lb, **double** h)  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **upperBottom** = ub;  
 **lowerBottom** = lb;  
 **height** = h;  
 *// 计算面积* **square** = (**upperBottom** + **lowerBottom**)\***height**/2;  
 }  
  
 *//返回面积* **protected double** getSquare(){**return square**;}  
}

**package** Demo4;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
  
 Circle类具有类型为double的半径、周长和面积属性，  
 具有返回周长、面积的功能。  
 \*/***public class** Circle {  
 *// 静态成员变量pi* **private static double** *pi* = 3.1415926;  
 **private double radii**;  
 **private double perimeter**;  
 **private double square**;  
  
 Circle(**double** r)  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **radii** = r;  
 **perimeter** = 2\**pi*\***radii**;  
 **square** = r\*r\**pi*;  
 }  
  
 *//返回周长* **protected double** getPerimeter() {**return perimeter**;}  
  
 *//返回面积* **protected double** getSquare() {**return square**;}  
}

**package** Demo2\_3;  
**import** java.util.Vector;  
**import** java.lang.Math;  
  
*/\*\*  
 \* Created by hj on 16-7-14.  
 \* 定义一个一维数组，其中存储随机生成的1000个1至100以内的整数，统计每个整数出现的次数。  
 \*/***public class** Demo\_2\_3\_1  
{  
 **private** Vector<Integer> **NumVector** = **new** Vector<>(100, 50);  
  
 **public** Demo\_2\_3\_1()  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **int** i,tmp;  
 **for**(i=0;i<1000;i++)  
 {  
 tmp = (**int**)(Math.*random*()\*100+1);  
 **NumVector**.addElement(tmp);  
 }  
 **this**.countVector();  
 }  
  
 **private void** countVector()  
 {  
 */\*\*  
 \* 统计向量中每个元素的个数  
 \*/* **int** i,j;  
 **int** count;  
 **for**(i=1;i<=100;i++)  
 {  
 count = 0;  
 **for**(j=0;j<1000;j++)  
 {  
 **if**(**NumVector**.elementAt(j) == i)  
 count ++;  
 }  
 System.***out***.println(i+**" : "**+count);  
 }  
 }  
  
}

实验2

**package** Demo4;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
 创建一个名为pay的类，该类包括每小时价格(price)、扣缴率(withhold\_rate)、工资率(pay\_rate)和应得工资 (payment)  
 等4个double型成员变量。创建3个重载方法computePayment()来计算应得工资，计算规则如下：  
  
 1)当computePayment ()接收小时、扣缴率和工资率三个参数值时，计算应得工资=工作小时\*每小时工资\*(1-扣缴率)\*(1-工资率)；  
 2)当computePayment()接收小时和工资率两个参数时，扣缴率为15%，计算应得工资=工作小时\*每小时工资\*(1-0.15)\*(1-工资率)；  
 3)当computePayment ()接收一个参数时，扣缴率为15%，每小时工资率为4.65%，计算应得工资=工作小时\*每小时工资\*(1-0.15)\*(1-0.0465)。  
 \*/***public class** pay {  
 **private double price**;  
 **private double withhold\_rate**;  
 **private double pay\_rate**;  
 **private double payment**;  
  
 pay(**double** p, **double** wr, **double** pr)  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **price** = p;  
 **withhold\_rate** = wr;  
 **pay\_rate** = pr ;  
 }  
  
 @overload  
 **protected void** computePayment(**double** workhour)  
 {  
 */\*\*  
 \* 计算工资的抽重载函数 - 1  
 \** ***@param workhour*** *工作时间  
 \*/* System.***out***.println(**"仅有工作时长的计算工资方法"**);  
 **payment** = **price**\*(1-0.15)\*(1-0.0465);  
 System.***out***.print(**"工资为："**);  
 System.***out***.print(**payment**);  
 System.***out***.println();  
 }  
   
 @overload  
 **protected void** computePayment(**double** workhour, **double** in\_pay\_rate)  
 {  
 */\*\*  
 \* 计算工资的抽重载函数 - 2  
 \** ***@param workhour*** *工作时间  
 \** ***@param in\_pay\_rate*** *传入的工费率  
 \*/* System.***out***.println(**"有工作时长和工费率的计算工资方法"**);  
 **payment** = **price**\*(1-0.15)\*(1-in\_pay\_rate);  
 System.***out***.print(**"工资为："**);  
 System.***out***.print(**payment**);  
 System.***out***.println();  
 }  
  
 @overload  
 **protected void** computePayment(**double** workhour, **double** in\_pay\_rate, **double** in\_withhold\_rate)  
 {  
 */\*\*  
 \* 计算工资的重载函数 - 3  
 \** ***@param workhour*** *工作时间  
 \** ***@param in\_pay\_rate*** *传入的工费率  
 \** ***@param in\_withhold\_rate*** *传入的扣缴率  
 \*/* System.***out***.println(**"有工作时长,工费率和扣缴率的计算工资方法"**);  
 **payment** = **price**\*(1-in\_withhold\_rate)\*(1-in\_pay\_rate);  
 System.***out***.print(**"工资为："**);  
 System.***out***.print(**payment**);  
 System.***out***.println();  
 }  
}

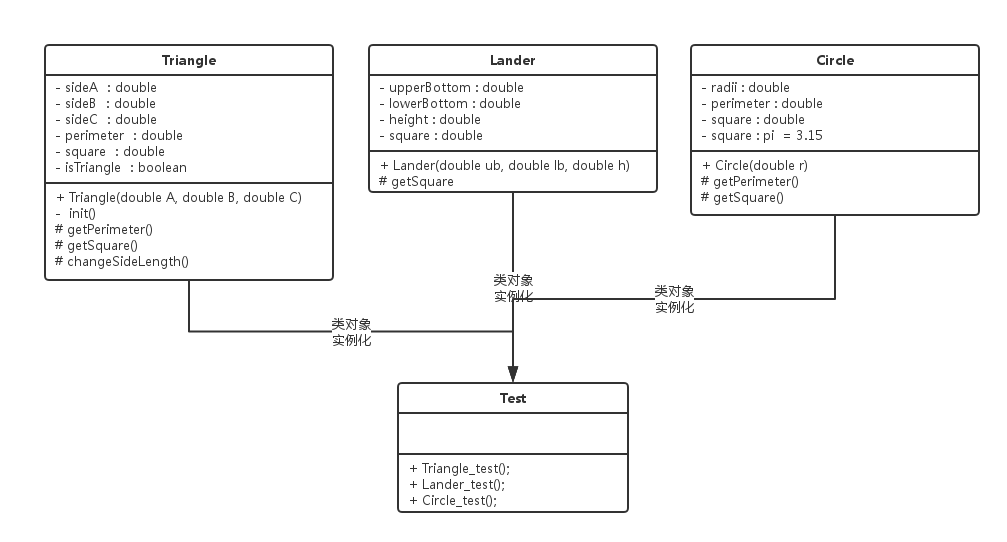
**package** Demo4;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
 创建一个名为pay的类，该类包括每小时价格(price)、扣缴率(withhold\_rate)、工资率(pay\_rate)和应得工资 (payment)  
 等4个double型成员变量。创建3个重载方法computePayment()来计算应得工资，计算规则如下：  
  
 1)当computePayment ()接收小时、扣缴率和工资率三个参数值时，计算应得工资=工作小时\*每小时工资\*(1-扣缴率)\*(1-工资率)；  
 2)当computePayment()接收小时和工资率两个参数时，扣缴率为15%，计算应得工资=工作小时\*每小时工资\*(1-0.15)\*(1-工资率)；  
 3)当computePayment ()接收一个参数时，扣缴率为15%，每小时工资率为4.65%，计算应得工资=工作小时\*每小时工资\*(1-0.15)\*(1-0.0465)。  
 编写一个测试类Test，该测试类的main方法测试所有3个重载的方法。  
 \*/***public class** Demo\_4\_2 {  
 Demo\_4\_2()  
 {  
 pay p = **new** pay(100,0.15,0.0465);  
 p.computePayment(100);  
 p.computePayment(1,0.0465);  
 p.computePayment(1,0.15,0.0465);  
 }  
}

实验3

**package** Demo4;  
  
**import** java.lang.String;  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
 名为Telephone的类，  
 Telephone类中包含有  
 电话品牌(brand)、  
 电话号码(number)、  
 通话时间(dialedTime)、  
 费率(rate) 等属性，  
  
 以及计算话费(callCost) 和  
 显示话费信息(display)方法。  
 \*/***public class** Telephone {  
 **protected** String **brand**;  
 **protected** String **number**;  
 **protected double** dialedTime;  
 **protected double rate**;  
 **protected double telephoneFare**;  
  
 Telephone(String in\_brand, String in\_number, **double** in\_dialedTime, **double** in\_rata)  
 {  
 */\*\*  
 \* 构造函数  
 \*/* **brand** = in\_brand;  
 **number** = in\_number;  
 dialedTime = in\_dialedTime;  
 **rate** = in\_rata;  
 }  
  
 *//计算费率* **protected void** callCost() {**telephoneFare** = dialedTime\***rate**;}  
  
 **protected void** display()  
 {  
 */\*\*  
 \* 展示信息函数  
 \*/* callCost();  
  
 System.***out***.println(**"电话品牌:"**+**brand**);  
 System.***out***.println(**"电话号码:"**+**number**);  
 System.***out***.println(**"通话时间:"**+dialedTime+**"分钟"**);  
 System.***out***.println(**"费率 :"**+**rate**+**"元/分钟"**);  
 System.***out***.println(**"花费总计:"**+**telephoneFare**+**"元"**);  
 }  
  
}

**package** Demo4;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
  
再创建一个名为Mobilephone的类，它是Telephone的子类，  
 除了具有Telephone类的属性外，  
 它还有自己的属性如网络类型(network)、  
 被叫时间(receivedTime)，  
  
 同时它有自己的计算话费和显示信息的方法。  
 \*/***public class** Mobilephone **extends** Demo4.Telephone {  
 **protected** String **network**;  
 **protected double receivedTime**;  
  
 Mobilephone(String in\_brand, String in\_number, **double** in\_dialedTime, **double** in\_rata,  
 String in\_network, **double** in\_receivedTime)  
 {  
 */\*\*  
 \* 构造函数-利用父类构造函数  
 \** ***@param in\_agr*** *传入的参数  
 \*/* **super**(in\_brand, in\_number, in\_dialedTime, in\_rata);  
 **network** = in\_network;  
 **receivedTime** = in\_receivedTime;  
 }  
  
 *//计算费率* **@Override  
 protected void** callCost() {**telephoneFare** = (**dialedTime**-**receivedTime**)\***rate**+**receivedTime**\***rate**/2;}  
  
 **@Override  
 protected void** display()  
 {  
 */\*\*  
 \* 展示信息函数-重写父类方法  
 \*/* **double** callingTime;  
 callingTime = **dialedTime**-**receivedTime**;  
 callCost();  
  
 System.***out***.println(**"电话品牌:"**+**brand**);  
 System.***out***.println(**"电话号码:"**+**number**);  
 System.***out***.println(**"网络 :"**+**network**);  
 System.***out***.println(**"主叫时间:"**+callingTime+**"分钟"**);  
 System.***out***.println(**"被叫时间:"**+**receivedTime**+**"分钟"**);  
 System.***out***.println(**"费率 :"**+**rate**+**"元/分钟"**);  
 System.***out***.println(**"花费总计:"**+**telephoneFare**+**"元"**);  
 }  
}  
*//网络： 3G  
// 主叫时间：80.0 分钟  
// 被叫时间：120.0分钟  
// 费率： 0.4元/分钟  
// 花费总计：56.0元*

**package** Demo4;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/22.  
 创建一个名为Telephone的类，  
 Telephone类中包含有 电话品牌(brand)、  
 电话号码(number)、  
 通话时间(dialedTime)、  
 费率(rate)等属性，  
  
 以及计算话费(callCost) 和  
 显示话费信息(display)方法。  
  
 再创建一个名为Mobilephone的类，它是Telephone的子类，  
 除了具有Telephone类的属性外，  
 它还有自己的属性如网络类型(network)、  
 被叫时间(receivedTime)，  
  
 同时它有自己的计算话费和显示信息的方法。  
  
 另外，程序中还有一个  
 主类PhoneCost来使用上述两个类并显示它们的信息。  
  
  
 电话话费= dialedTime\*rate；  
 电话费率rate=0.2元/分钟；  
 移动电话话费=(dialedTime+0.5\*receivedTime)\*rate；  
 移动电话费率rate=0.4元/分钟  
  
 ---DEMO----  
 程序运行的效果如下：  
  
 电话品牌：TCL  
 电话号码：5687506  
 通话时间：130.0分钟  
 费率： 0.2元/分钟  
 花费总计：26.0元  
  
 电话品牌：SAMSUNG  
 电话号码：15163155787  
 网络： 3G  
 主叫时间：80.0 分钟  
 被叫时间：120.0分钟  
 费率： 0.4元/分钟  
 花费总计：56.0元  
 \*/***public class** Demo\_4\_3 {  
  
 Demo\_4\_3()  
 {  
 Telephone t = **new** Telephone(**"TLC"**,**"5687506"**,130.0,0.2);  
 t.display();  
  
 System.***out***.println();  
  
 Mobilephone mt = **new** Mobilephone(**"SAMSUNG"**,**"15163155787"**,200,0.4,**"3G"**,120);  
 mt.display();  
 }  
  
}



# 实验5 Java图形用户界面设计

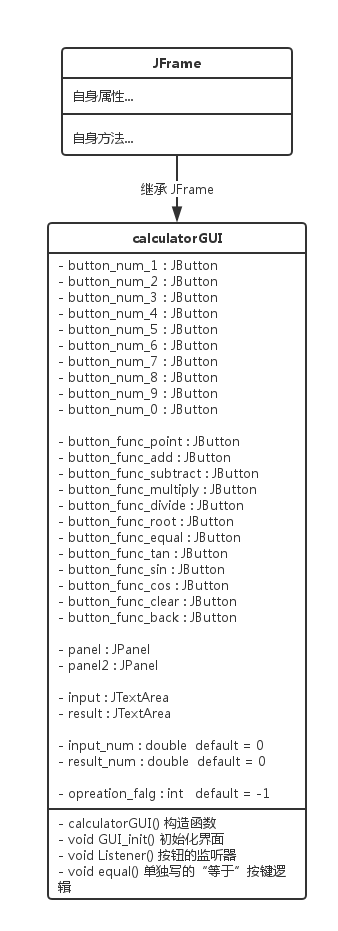
1. **实验目的**
2. 熟悉和理解Java中AWT和Swing，能够设计简单美观的用户界面。
3. 掌握使用Java中图形界面设计的基本元素与方法。
4. **实验内容**
5. 利用Frame设计一个简单计算器，该计算器可以实现三种常用的三角函数(sin、cos、tan)和5种常用运算(+、−、×、/、√)的功能。计算器窗口内的内容大多是按键，将这些按键全部定义为按钮，另外，还需要一个显示区，用来表示输入的值及最后结果。
6. 编写一个图书管理程序，该程序具有如下功能：
7. 录入图书信息(Book)，图书的属性包括书名(name)、作者(author)、出版社(press)、刊号(ISBN)、出版日期(pubDate)、页数(pages)等。
8. 录入用户信息(User)，用户属性包括用户名(account)，密码(password)、用户类型(type)，用户分为管理员和普通用户。
9. 记录每个用户的借书记录，借书记录(BorrowRecord)包括：用户(user)、借书日期(borrowDate)、图书(book)。
10. 记录每个用户的还书记录，还书记录(ReturnRecord)包括：用户(user)、还书日期(returnDate)、图书(book)。

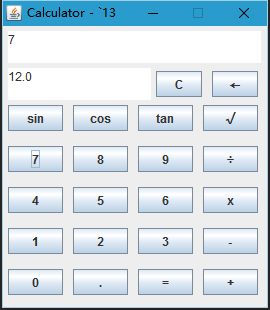
**要求：**

1. 根据上述描述画出类图。
2. 定义所有的类以及每个类中所包含的属性和方法，需要通过注释来说明每个属性和方法的含义。
3. 采用图形界面来实现用户登陆、录入图书信息、录入用户信息、录入借书记录、录入还书记录等功能。

实验1

**package** Demo5;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** javax.swing.JButton;  
**import** javax.swing.JFrame;  
**import** javax.swing.JTextArea;  
**import** javax.swing.JPanel;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/25.  
 利用Frame设计一个简单计算器，  
  
 该计算器可以实现三种常用的三角函数(sin、cos、tan) 和  
 5种常用运算(+,-,\*,/,sqr)的功能。  
  
 计算器窗口内的内容大多是按键，将这些按键全部定义为按钮，  
 另外，还需要一个显示区，用来表示输入的值及最后结果。  
 \*/***public class** calculatorGUI **extends** JFrame  
{  
 *// 数字按钮* **private** JButton **button\_num\_1** = **new** JButton(**"1"**);  
 **private** JButton **button\_num\_2** = **new** JButton(**"2"**);  
 **private** JButton **button\_num\_3** = **new** JButton(**"3"**);  
 **private** JButton **button\_num\_4** = **new** JButton(**"4"**);  
 **private** JButton **button\_num\_5** = **new** JButton(**"5"**);  
 **private** JButton **button\_num\_6** = **new** JButton(**"6"**);  
 **private** JButton **button\_num\_7** = **new** JButton(**"7"**);  
 **private** JButton **button\_num\_8** = **new** JButton(**"8"**);  
 **private** JButton **button\_num\_9** = **new** JButton(**"9"**);  
 **private** JButton **button\_num\_0** = **new** JButton(**"0"**);  
  
 *//功能按钮* **private** JButton **button\_func\_point** = **new** JButton(**"."**);  
 **private** JButton **button\_func\_add** = **new** JButton(**"+"**);  
 **private** JButton **button\_func\_subtract** = **new** JButton(**"-"**);  
 **private** JButton **button\_func\_multiply** = **new** JButton(**"x"**);  
 **private** JButton **button\_func\_divide** = **new** JButton(**"÷"**);  
 **private** JButton **button\_func\_root** = **new** JButton(**"√"**);  
 **private** JButton **button\_func\_equal** = **new** JButton(**"="**);  
 **private** JButton **button\_func\_tan** = **new** JButton(**"tan"**);  
 **private** JButton **button\_func\_sin** = **new** JButton(**"sin"**);  
 **private** JButton **button\_func\_cos** = **new** JButton(**"cos"**);  
 **private** JButton **button\_func\_clear** = **new** JButton(**"C"**);  
 **private** JButton **button\_func\_back** = **new** JButton(**"←"**);  
  
 **private** JPanel **panel2** = **new** JPanel(**new** GridLayout(1,2,10,15)); *// 功能按钮面板容器* **private** JPanel **panel** = **new** JPanel(**new** GridLayout(5,4,10,15)); *// 按钮面板容器* **private** JTextArea **input** = **new** JTextArea(2,23); *// 输入文本条* **private** JTextArea **result** = **new** JTextArea(2,13); *// 输出文本条* **private double input\_num** = 0;  
 **private double result\_num** = 0;  
 **private int opreation\_falg** = -1;  
 *// 操作表示 -1 = 默认无操作。0-等 1-加,2-减,3-乘,4-除。* calculatorGUI()  
 {  
 GUI\_init();  
 Listener();  
 }  
  
 **private void** GUI\_init()  
 {  
 */\*\*  
 \* GUI界面初始化  
 \*/* setTitle(**"Calculator - `13"**); *//标题* setLayout(**new** FlowLayout(FlowLayout.***LEADING***)); *//设置为流窗格* setSize(270, 310); *//大小* setResizable(**false**); *//不能缩放* setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***); *//关闭操作* setLocationRelativeTo(**null**); *//设置显示在屏幕中央* **input**.setEditable(**false**); *//无法修改文本条* **result**.setEditable(**false**); *//无法修改文本条* add(**input**); *//增加输入文本条* add(**result**); *//增加输入文本条  
 //第一行  
 //panel2.add(result); //增加结果文本条* **panel2**.add(**button\_func\_clear**);  
 **panel2**.add(**button\_func\_back**);  
 add(**panel2**); *//功能及结果面板  
  
 // 第一行* **panel**.add(**button\_func\_sin**);  
 **panel**.add(**button\_func\_cos**);  
 **panel**.add(**button\_func\_tan**);  
 **panel**.add(**button\_func\_root**);  
 *// 第二行* **panel**.add(**button\_num\_7**);  
 **panel**.add(**button\_num\_8**);  
 **panel**.add(**button\_num\_9**);  
 **panel**.add(**button\_func\_divide**);  
 *// 第三行* **panel**.add(**button\_num\_4**);  
 **panel**.add(**button\_num\_5**);  
 **panel**.add(**button\_num\_6**);  
 **panel**.add(**button\_func\_multiply**);  
 *// 第四行* **panel**.add(**button\_num\_1**);  
 **panel**.add(**button\_num\_2**);  
 **panel**.add(**button\_num\_3**);  
 **panel**.add(**button\_func\_subtract**);  
 *// 第五行* **panel**.add(**button\_num\_0**);  
 **panel**.add(**button\_func\_point**);  
 **panel**.add(**button\_func\_equal**);  
 **panel**.add(**button\_func\_add**);  
 add(**panel**);*// 增加面板* setVisible(**true**); *// 设为可见* **input**.setText(**"0"**); *// 输入框设为 0* **result**.setText(**"0"**); *// 输出框设为 0* }  
  
 **private void** Listener()  
 {  
 */\*\*  
 \* 注册按钮的事件监听器  
 \*/  
  
 //给数字按钮添加事件监听器* **button\_num\_0**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"0"**);  
 **else  
 input**.setText(**input**.getText()+**"0"**);  
 }  
 }  
 );  
 **button\_num\_1**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"1"**);  
 **else  
 input**.setText(**input**.getText()+**"1"**);  
 }  
 }  
 );  
 **button\_num\_2**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"2"**);  
 **else  
 input**.setText(**input**.getText()+**"2"**);  
 }  
 }  
 );  
 **button\_num\_3**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"3"**);  
 **else  
 input**.setText(**input**.getText()+**"3"**);  
 }  
 }  
 );  
 **button\_num\_4**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"4"**);  
 **else  
 input**.setText(**input**.getText()+**"4"**);  
 }  
 }  
 );  
 **button\_num\_5**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"5"**);  
 **else  
 input**.setText(**input**.getText()+**"5"**);  
 }  
 }  
 );  
 **button\_num\_6**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"6"**);  
 **else  
 input**.setText(**input**.getText()+**"6"**);  
 }  
 }  
 );  
 **button\_num\_7**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"7"**);  
 **else  
 input**.setText(**input**.getText()+**"7"**);  
 }  
 }  
 );  
 **button\_num\_8**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"8"**);  
 **else  
 input**.setText(**input**.getText()+**"8"**);  
 }  
 }  
 );  
 **button\_num\_9**.addActionListener(  
 **new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **if** (**input\_num** == 0 && **input**.getText().length() == 1)  
 **input**.setText(**"9"**);  
 **else  
 input**.setText(**input**.getText()+**"9"**);  
 }  
 }  
 );  
  
 *// 给小数点添加事件监听器* **button\_func\_point**.addActionListener(**new** ActionListener()  
 {**@Override public void** actionPerformed(ActionEvent e) {**input**.setText(**input**.getText()+**"."**);}});  
  
 *// 给Clear添加事件监听器* **button\_func\_clear**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **opreation\_falg** = -1;  
 **input**.setText(**"0"**);  
 **result**.setText(**"0"**);  
 }  
 });  
  
 *// 给Back添加事件监听器* **button\_func\_back**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **if** (**input**.getText().length() != 1)  
 **input**.setText(**input**.getText().substring(0,**input**.getText().length()-1));  
 **else  
 input**.setText(**"0"**);  
 }  
 });  
  
 *// 给三角函数添加事件监听器* **button\_func\_sin**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Math.*sin*(**input\_num**);  
 **result**.setText(Double.*toString*(**result\_num**));  
 }  
 });  
 **button\_func\_cos**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Math.*cos*(**input\_num**);  
 **result**.setText(Double.*toString*(**result\_num**));  
 }  
 });  
 **button\_func\_tan**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Math.*tan*(**input\_num**);  
 **result**.setText(Double.*toString*(**result\_num**));  
 }  
 });  
  
 *// 给开方添加事件监听器* **button\_func\_root**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Math.*sqrt*(**input\_num**);  
 **result**.setText(Double.*toString*(**result\_num**));  
 }  
 });  
  
  
 *// 给基本操作添加事件监听器* **button\_func\_add**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Double.*parseDouble*(**result**.getText());  
  
 **if** (**opreation\_falg** == -1)  
 {  
 **input**.setText(**"0"**);  
 **result**.setText(Double.*toString*(**input\_num**));  
 }  
 **else** {  
 equal();  
  
 }  
 **opreation\_falg** = 1;  
 }  
 });  
 **button\_func\_divide**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Double.*parseDouble*(**result**.getText());  
  
 **if** (**opreation\_falg** == -1)  
 {  
 **input**.setText(**"0"**);  
 **result**.setText(Double.*toString*(**input\_num**));  
 }  
 **else** {  
 equal();  
  
 }  
 **opreation\_falg** = 4;  
 }  
 });  
 **button\_func\_subtract**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Double.*parseDouble*(**result**.getText());  
  
 **if** (**opreation\_falg** == -1)  
 {  
 **input**.setText(**"0"**);  
 **result**.setText(Double.*toString*(**input\_num**));  
 }  
 **else** {  
 equal();  
  
 }  
 **opreation\_falg** = 2;  
 }  
 });  
 **button\_func\_multiply**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Double.*parseDouble*(**result**.getText());  
  
 **if** (**opreation\_falg** == -1)  
 {  
 **input**.setText(**"0"**);  
 **result**.setText(Double.*toString*(**input\_num**));  
 }  
 **else** {  
 equal();  
  
 }  
 **opreation\_falg** = 3;  
 }  
 });  
  
 *//添加等号监听器* **button\_func\_equal**.addActionListener(**new** ActionListener()  
 {  
 **@Override  
 public void** actionPerformed(ActionEvent e)  
 {  
 equal();  
 **opreation\_falg** = 0;  
 }  
 });  
  
  
 }  
  
 *// 等号的监听器单独写成方法,供其它监听器调用* **private void** equal()  
 {  
 **input\_num** = Double.*parseDouble*(**input**.getText());  
 **result\_num** = Double.*parseDouble*(**result**.getText());  
  
 **switch** (**opreation\_falg**)  
 {  
 **case** -1:  
 **result\_num** = **input\_num**;  
 **break**;  
 **case** 0:  
 **break**;  
 **case** 1:  
 **result\_num** = **result\_num** + **input\_num**;  
 **break**;  
 **case** 2:  
 **result\_num** = **result\_num** - **input\_num**;  
 **break**;  
 **case** 3:  
 **result\_num** = **result\_num** \* **input\_num**;  
 **break**;  
 **case** 4:  
 **if** (**input\_num** != 0)  
 **result\_num** = **result\_num** / **input\_num**;  
 **else  
 result\_num** = 88888888;  
 **break**;  
 }  
 **input**.setText(**"0"**);  
 **result**.setText(Double.*toString*(**result\_num**));  
  
 }  
}





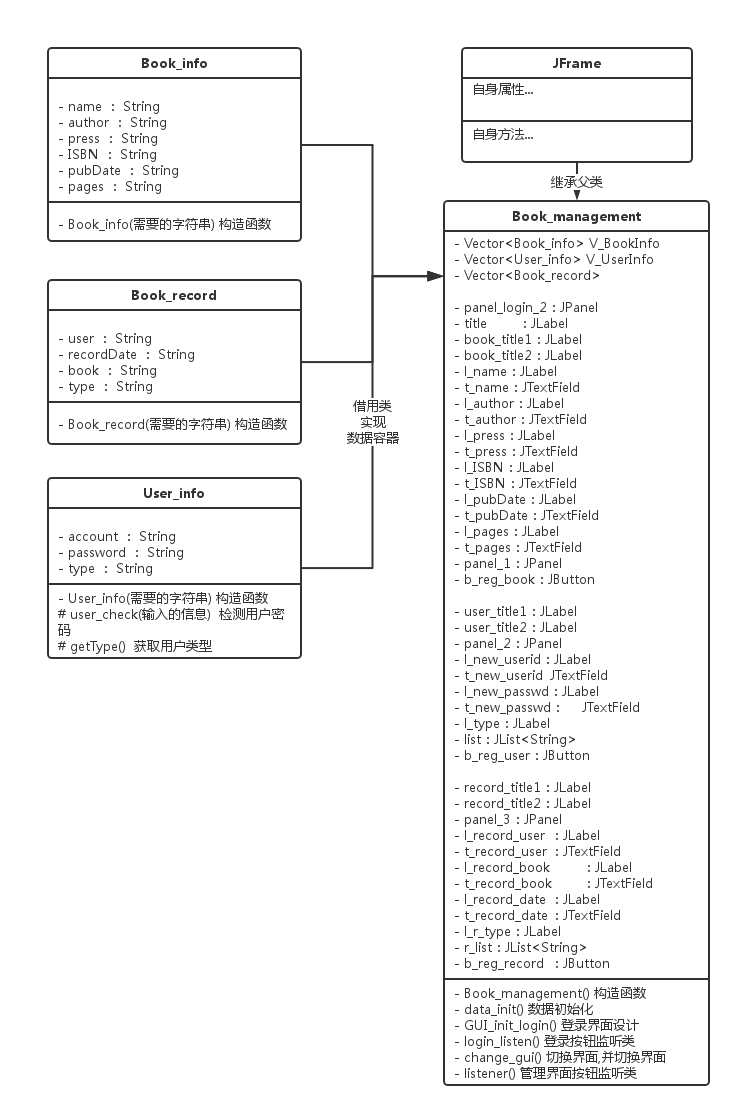
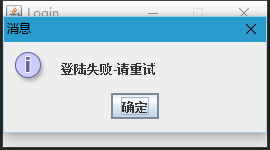
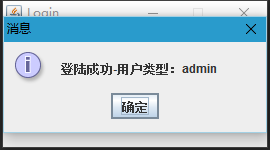
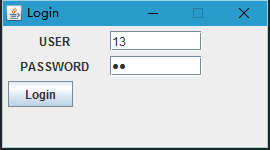
实验2

**package** Demo5;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/25.  
  
 图书信息(Book)，图书的属性包括  
 书名(name)、  
 作者(author)、  
 出版社(press)、  
 刊号(ISBN)、  
 出版日期(pubDate)、  
 页数(pages)等。  
 \*/***public class** Book\_info  
{  
 **private** String **name**;  
 **private** String **author**;  
 **private** String **press**;  
 **private** String **ISBN**;  
 **private** String **pubDate**;  
 **private** String **pages**;  
  
 Book\_info(String str1, String str2, String str3, String str4, String str5, String str6)  
 {  
 */\*\* 构造函数 \*/* **name** = str1;  
 **author** = str2;  
 **press** = str3;  
 **ISBN** = str4;  
 **pubDate** = str5;  
 **pages** = str6;  
 }  
  
}

**package** Demo5;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/25.  
 用户信息(User)，用户属性包括  
 用户名(account)，  
 密码(password)，  
 用户类型(type)，  
  
 用户分为管理员和普通用户。  
 \*/***public class** User\_info  
{  
 **private** String **account**;  
 **private** String **password**;  
 **private** String **type**;  
  
 User\_info(String str1, String str2, String str3)  
 {  
 */\*\* 构造函数 \*/* **account** = str1;  
 **password** = str2;  
 **type** = str3;  
 }  
  
 **protected boolean** user\_check(String user,String pswd)  
 {  
 */\*\* 检查输入的用户名和密码 \*/* **return**(**account**.equals(user)&& **password**.equals(pswd));  
 }  
  
 **protected** String getType()  
 {  
 */\*\* 获取账户属性 \*/* **return type**;  
 }  
}

**package** Demo5;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/25.  
 借书记录(BorrowRecord)包括：  
 用户(user)、  
 借书日期(borrowDate)、  
 图书(book)。  
  
 还书记录(ReturnRecord)包括：  
 用户(user)、  
 还书日期(returnDate)、  
 图书(book)。  
 \*/***public class** Book\_record  
{  
 **private** String **user**;  
 **private** String **recordDate**;  
 **private** String **book**;  
 **private** String type;  
  
 Book\_record(String str1, String str2, String str3, String str4)  
 {  
 */\*\* 构造函数 \*/* **user** = str1;  
 **recordDate** = str2;  
 **book** = str3;  
 type = str4;  
 }  
  
}

**package** Demo5;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.util.Vector;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/25.  
 编写一个图书管理程序，该程序具有如下功能：  
 1) 录入图书信息(Book)，  
 图书的属性包括书名(name)、作者(author)、  
 出版社(press)、刊号(ISBN)、出版日期(pubDate)、页数(pages)等。  
 2) 录入用户信息(User)，  
 用户属性包括用户名(account)，密码(password)、用户类型(type)，  
 用户分为管理员和普通用户。  
 3) 记录每个用户的借书记录，借书记录(BorrowRecord)包括：  
 用户(user)、借书日期(borrowDate)、图书(book)。  
 4) 记录每个用户的还书记录，还书记录(ReturnRecord)包括：  
 用户(user)、还书日期(returnDate)、图书(book)。  
 \*/***public class** Book\_management **extends** JFrame  
{  
 *// 数据容器* **private** Vector<Book\_info> **V\_BookInfo** = **new** Vector<>(10,5);  
 **private** Vector<User\_info> **V\_UserInfo** = **new** Vector<>(2,1);  
 **private** Vector<Book\_record> **V\_BookRecord** = **new** Vector<>(10,5);  
  
 *// 登录界面GUI* **private** JButton **b\_login** = **new** JButton(**"Login"**);  
 **private** JLabel **l\_passwd** = **new** JLabel(**"PASSWORD"**,JLabel.***CENTER***);  
 **private** JPasswordField **t\_passwd** = **new** JPasswordField(8);  
 **private** JLabel **l\_userid** = **new** JLabel(**"USER"**,JLabel.***CENTER***);  
 **private** JTextField **t\_user** = **new** JTextField(8);  
 **private** JPanel **panel\_login\_1** = **new** JPanel(**new** GridLayout(1,2,10,15));  
 **private** JPanel **panel\_login\_2** = **new** JPanel(**new** GridLayout(1,2,10,15));  
  
 *// 管理界面GUI  
 // 录入图书信息* **private** JLabel **title** = **new** JLabel(**" Book management - `13 "**,JLabel.***CENTER***);  
 **private** JLabel **book\_title1** = **new** JLabel(**" 录入书籍信息： "**);  
 **private** JLabel **book\_title2** = **new** JLabel(**"------------------------------------------------------------------------------------------------------------------------------------------------------------- "**);  
 **private** JLabel **l\_name** = **new** JLabel(**"name"**,JLabel.***CENTER***);  
 **private** JTextField **t\_name** = **new** JTextField(8);  
 **private** JLabel **l\_author** = **new** JLabel(**"author"**,JLabel.***CENTER***);  
 **private** JTextField **t\_author** = **new** JTextField(8);  
 **private** JLabel **l\_press** = **new** JLabel(**"press"**,JLabel.***CENTER***);  
 **private** JTextField **t\_press** = **new** JTextField(8);  
 **private** JLabel **l\_ISBN** = **new** JLabel(**"ISBN"**,JLabel.***CENTER***);  
 **private** JTextField **t\_ISBN** = **new** JTextField(8);  
 **private** JLabel **l\_pubDate** = **new** JLabel(**"pubDate"**,JLabel.***CENTER***);  
 **private** JTextField **t\_pubDate** = **new** JTextField(8);  
 **private** JLabel **l\_pages** = **new** JLabel(**"pages"**,JLabel.***CENTER***);  
 **private** JTextField **t\_pages** = **new** JTextField(8);  
 **private** JPanel **panel\_1** = **new** JPanel(**new** GridLayout(2,6,10,15));  
 **private** JButton **b\_reg\_book** = **new** JButton(**" 登录书籍信息"**);  
  
 *// 录入用户信息* **private** JLabel **user\_title1** = **new** JLabel(**" 录入用户信息： "**);  
 **private** JLabel **user\_title2** = **new** JLabel(**"------------------------------------------------------------------------------------------------------------------------------------------------------------- "**);  
 **private** JPanel **panel\_2** = **new** JPanel(**new** GridLayout(1,6,10,15));  
 **private** JLabel **l\_new\_userid** = **new** JLabel(**"user\_id"**,JLabel.***CENTER***);  
 **private** JTextField **t\_new\_userid** = **new** JTextField(8);  
 **private** JLabel **l\_new\_passwd** = **new** JLabel(**"passwd"**,JLabel.***CENTER***);  
 **private** JTextField **t\_new\_passwd** = **new** JTextField(8);  
 **private** JLabel **l\_type** = **new** JLabel(**"user\_type"**,JLabel.***CENTER***);  
 String **s**[] = {**"admin"**, **"user"**};  
 **private** JList<String> **list** = **new** JList<>(**s**);  
 **private** JButton **b\_reg\_user** = **new** JButton(**"注册用户"**);  
  
 *// 借/还书信息* **private** JLabel **record\_title1** = **new** JLabel(**" 录入书籍借还信息： "**);  
 **private** JLabel **record\_title2** = **new** JLabel(**"------------------------------------------------------------------------------------------------------------------------------------------------------------- "**);  
 **private** JPanel **panel\_3** = **new** JPanel(**new** GridLayout(1,6,10,15));  
 **private** JLabel **l\_record\_user** = **new** JLabel(**"user"**,JLabel.***CENTER***);  
 **private** JTextField **t\_record\_user** = **new** JTextField(8);  
 **private** JLabel **l\_record\_book** = **new** JLabel(**"book"**,JLabel.***CENTER***);  
 **private** JTextField **t\_record\_book** = **new** JTextField(8);  
 **private** JLabel **l\_record\_date** = **new** JLabel(**"date"**,JLabel.***CENTER***);  
 **private** JTextField **t\_record\_date** = **new** JTextField(8);  
 **private** JLabel **l\_r\_type** = **new** JLabel(**"user\_type"**,JLabel.***CENTER***);  
 String **r**[] = {**"borrow"**, **"return"**};  
 **private** JList<String> **r\_list** = **new** JList<>(**r**);  
 **private** JButton **b\_reg\_record** = **new** JButton(**"记录注册"**);  
  
  
 Book\_management()  
 {  
 data\_init();  
 GUI\_init\_login();  
 login\_listen();  
 }  
 **private void** data\_init()  
 {  
 */\*\* 数据初始化 \*/  
 // 初始化图书信息* Book\_info b1 = **new** Book\_info(**"1"**,**"2"**,**"3"**,**"4"**,**"5"**,**"6"**);  
 **V\_BookInfo**.add(b1);  
 Book\_info b2 = **new** Book\_info(**"01"**,**"02"**,**"03"**,**"04"**,**"05"**,**"06"**);  
 **V\_BookInfo**.add(b2);  
 Book\_info b3 = **new** Book\_info(**"-1"**,**"-2"**,**"-3"**,**"-4"**,**"-5"**,**"-6"**);  
 **V\_BookInfo**.add(b3);  
 Book\_info b4 = **new** Book\_info(**"/1"**,**"/2"**,**"/3"**,**"/4"**,**"/5"**,**"/6"**);  
 **V\_BookInfo**.add(b4);  
 *// 初始化用户信息* User\_info u1 = **new** User\_info(**"13"**,**"13"**,**"admin"**);  
 **V\_UserInfo**.add(u1);  
 User\_info u2 = **new** User\_info(**"hj"**,**"hj"**,**"user"**);  
 **V\_UserInfo**.add(u2);  
 }  
  
 **private void** GUI\_init\_login()  
 {  
 */\*\*  
 \* GUI登录界面初始化  
 \*/* setTitle(**"Login"**); *// 标题* setLayout(**new** FlowLayout(FlowLayout.***LEADING***)); *// 设置为流窗格-竖排* setSize(270, 150); *// 大小* setResizable(**false**); *// 不能缩放* setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***); *// 关闭操作* setLocationRelativeTo(**null**); *// 设置显示在屏幕中央* **t\_passwd**.setEditable(**true**); *// 修改文本条* **t\_user**.setEditable(**true**); *// 修改文本条  
 // 设计GUI界面* **panel\_login\_2**.add(**l\_passwd**);  
 **panel\_login\_2**.add(**t\_passwd**);  
 **panel\_login\_1**.add(**l\_userid**);  
 **panel\_login\_1**.add(**t\_user**);  
 add(**panel\_login\_1**);  
 add(**panel\_login\_2**);  
 add(**b\_login**);  
  
 setVisible(**true**); *// 设为可见* }  
  
 **private void** login\_listen()  
 {  
 */\*\* 登录按钮监听器 \*/* **b\_login**.addActionListener(e ->  
 {  
  
 **int** V\_UserInfo\_length = **V\_UserInfo**.size();  
 **int** i;  
 **for** (i = 0; i < V\_UserInfo\_length; i++)  
 {  
 User\_info tmp = **V\_UserInfo**.elementAt(i);  
 **if** (tmp.user\_check(**t\_user**.getText(), **t\_passwd**.getText())) {  
 String dialog\_str = **"登陆成功-用户类型："** + tmp.getType();  
 JOptionPane.*showMessageDialog*(**null**, dialog\_str);  
 change\_gui();  
 **return**;  
 }  
 }  
 String dialog\_str = **"登陆失败-请重试"**;  
 JOptionPane.*showMessageDialog*(**null**, dialog\_str);  
 }  
 );  
 }  
  
 **private void** change\_gui()  
 {  
 */\*\*  
 \* GUI管理界面初始化  
 \*/* setTitle(**"Book Management - `13"**); *// 标题* setLayout(**new** FlowLayout(FlowLayout.***LEADING***)); *// 设置为流窗格-竖排* setSize(800, 600); *// 大小* setResizable(**true**); *// 可以缩放* setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***); *// 关闭操作* setLocationRelativeTo(**null**); *// 设置显示在屏幕中央  
 // 去除登录界面GUI的元素* remove(**panel\_login\_1**);  
 remove(**panel\_login\_2**);  
 remove(**b\_login**);  
  
 *// 重写生成GUI界面  
 // 图书信息录入界面* add(**title**);  
 add(**book\_title1**);  
 add(**book\_title2**);  
 **panel\_1**.add(**l\_name**);  
 **panel\_1**.add(**t\_name**);  
 **panel\_1**.add(**l\_author**);  
 **panel\_1**.add(**t\_author**);  
 **panel\_1**.add(**l\_press**);  
 **panel\_1**.add(**t\_press**);  
 **panel\_1**.add(**l\_ISBN**);  
 **panel\_1**.add(**t\_ISBN**);  
 **panel\_1**.add(**l\_pubDate**);  
 **panel\_1**.add(**t\_pubDate**);  
 **panel\_1**.add(**l\_pages**);  
 **panel\_1**.add(**t\_pages**);  
 add(**panel\_1**);  
 add(**b\_reg\_book**);  
  
 *// 用户录入界面* **list**.setVisibleRowCount(2);  
 **list**.setBorder(BorderFactory.*createTitledBorder*(**"用户类型"**));  
 add(**user\_title1**);  
 add(**record\_title2**);  
 **panel\_2**.add(**l\_new\_userid**);  
 **panel\_2**.add(**t\_new\_userid**);  
 **panel\_2**.add(**l\_new\_passwd**);  
 **panel\_2**.add(**t\_new\_passwd**);  
 **panel\_2**.add(**l\_type**);  
 **panel\_2**.add(**list**);  
 add(**panel\_2**);  
 add(**b\_reg\_user**);  
  
 *// 记录登入界面* **r\_list**.setVisibleRowCount(2);  
 **r\_list**.setBorder(BorderFactory.*createTitledBorder*(**"记录类型"**));  
 add(**record\_title1**);  
 add(**user\_title2**);  
 **panel\_3**.add(**l\_record\_user**);  
 **panel\_3**.add(**t\_record\_user**);  
 **panel\_3**.add(**l\_record\_book**);  
 **panel\_3**.add(**t\_record\_book**);  
 **panel\_3**.add(**l\_record\_date**);  
 **panel\_3**.add(**t\_record\_date**);  
 **panel\_3**.add(**l\_r\_type**);  
 **panel\_3**.add(**r\_list**);  
 add(**panel\_3**);  
 add(**b\_reg\_record**);  
 setVisible(**true**); *// 设为可见* listener();  
 }  
  
 **private void** listener()  
 {  
 */\*\* 按钮的监听类 \*/* **b\_reg\_book**.addActionListener(e ->  
 {  
 Book\_info tmp = **new** Book\_info(**t\_name**.getText(),**t\_author**.getText(),**t\_press**.getText(),**t\_ISBN**.getText(),**t\_pubDate**.getText(),**t\_pages**.getText());  
 **V\_BookInfo**.add(tmp);  
 **t\_name**.setText(**""**);  
 **t\_author**.setText(**""**);  
 **t\_press**.setText(**""**);  
 **t\_ISBN**.setText(**""**);  
 **t\_pubDate**.setText(**""**);  
 **t\_pages**.setText(**""**);  
 }  
 );  
  
 **b\_reg\_user**.addActionListener(e ->  
 {  
 User\_info tmp = **new** User\_info(**t\_new\_userid**.getText(),**t\_new\_passwd**.getText(),**list**.getSelectedValue());  
 **V\_UserInfo**.add(tmp);  
 **t\_new\_userid**.setText(**""**);  
 **t\_new\_passwd**.setText(**""**);  
 }  
 );  
  
 **b\_reg\_record**.addActionListener(e ->  
 {  
 Book\_record tmp = **new** Book\_record(**t\_record\_user**.getText(),**t\_record\_book**.getText(),**t\_record\_date**.getText(),**r\_list**.getSelectedValue());  
 **V\_BookRecord**.add(tmp);  
 **t\_record\_user**.setText(**""**);  
 **t\_record\_book**.setText(**""**);  
 **t\_record\_date**.setText(**""**);  
 }  
 );  
 }  
}



# 实验6 Java数据流

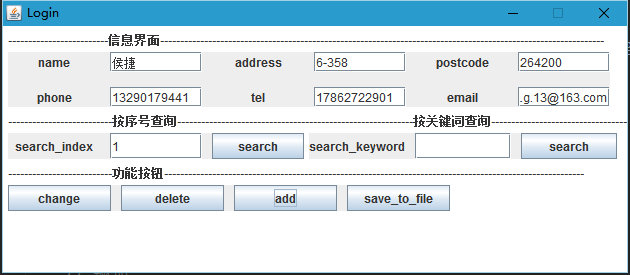
1. **实验目的**

本实验要求学生能够理解Java数据流概念，理解Java流类的层次结构，能够访问文件，并且将文件中的内容在图形界面上显示。

1. **必须学习和掌握的知识点**
2. 能够创建、读写和更新文件。
3. 能够使用各种输入/输出数据流。
4. 熟悉顺序存取文件和随机存取文件的处理。
5. 熟悉各种Swing组件的使用方法。
6. **实验内容**

编写一个个人通信录程序，具有如下功能：

* 定义一个通信录类，包括姓名、地址、邮政编码、电话、手机、Email等成员变量；
* 将通信录类的对象存储在文件中。
* 设计一个用户界面，在该界面中可以实现通信录增加、修改、删除和查询的功能。



**package** Demo6;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/26.  
  定义一个通信录类，  
 包括 姓名、  
 地址、  
 邮政编码、  
 电话、  
 手机、  
 Email  
 等成员变量；  
 \*/***public class** AddressListData {  
 **protected** String **name**;  
 **protected** String **adress**;  
 **protected** String **post\_code**;  
 **protected** String **phone**;  
 **protected** String **tel**;  
 **protected** String **email**;  
  
 AddressListData(String N,String AD,String Pc,String p, String t, String e)  
 {  
 */\*\* 构造函数 \*/* **name** = N;  
 **adress** = AD;  
 **post\_code** = Pc;  
 **phone** = p;  
 **tel** = t;  
 **email** = e;  
 }  
  
 **protected boolean** search\_by\_keyword(String input)  
 {  
 **return** (**name**.equals(input)||**adress**.equals(input)||**post\_code**.equals(input)||**phone**.equals(input)||**tel**.equals(input)||**email**.equals(input));  
 }  
  
}

**package** Demo6;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.io.\*;  
**import** java.util.Vector;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/26.  
 编写一个个人通信录程序，具有如下功能：  
  定义一个通信录类，包括姓名、地址、邮政编码、电话、手机、Email等成员变量；  
  
  将通信录类的对象存储在文件中。  
  
  设计一个用户界面，在该界面中可以实现  
 通信录增加、  
 修改、  
 删除和  
 查询的功能。  
  
 \*/***public class** AddressListGUI **extends** Frame  
{  
 *// 数据容器* **private** Vector<AddressListData> **V\_ALD** = **new** Vector<>(10,5);  
  
 *// GUI元素* **private** JLabel **title\_1** = **new** JLabel(**"-------------------------信息界面---------------------------------------------------------------------------------------------------------------"**,JLabel.***CENTER***);  
 **private** JLabel **l\_name** = **new** JLabel(**"name"**,JLabel.***CENTER***);  
 **private** JTextField **t\_name** = **new** JTextField(8);  
 **private** JLabel **l\_address** = **new** JLabel(**"address"**,JLabel.***CENTER***);  
 **private** JTextField **t\_address** = **new** JTextField(8);  
 **private** JLabel **l\_postcode** = **new** JLabel(**"postcode"**,JLabel.***CENTER***);  
 **private** JTextField **t\_postcode** = **new** JTextField(8);  
 **private** JLabel **l\_phone** = **new** JLabel(**"phone"**,JLabel.***CENTER***);  
 **private** JTextField **t\_phone** = **new** JTextField(8);  
 **private** JLabel **l\_tel** = **new** JLabel(**"tel"**,JLabel.***CENTER***);  
 **private** JTextField **t\_tel** = **new** JTextField(8);  
 **private** JLabel **l\_email** = **new** JLabel(**"email"**,JLabel.***CENTER***);  
 **private** JTextField **t\_email** = **new** JTextField(8);  
 **private** JPanel **panel\_1** = **new** JPanel(**new** GridLayout(2,6,10,15));  
  
 **private** JLabel **title\_2** = **new** JLabel(**"--------------------------按序号查询-----------------------------------------------------------按关键词查询-----------------------------------"**,JLabel.***CENTER***);  
 **private** JLabel **l\_index** = **new** JLabel(**"search\_index"**,JLabel.***CENTER***);  
 **private** JTextField **t\_index** = **new** JTextField(8);  
 **private** JButton **b\_index** = **new** JButton(**"search"**);  
 **private** JPanel **panel\_2** = **new** JPanel(**new** GridLayout(1,3,10,15));  
  
 **private** JLabel **l\_keyword** = **new** JLabel(**"search\_keyword"**,JLabel.***CENTER***);  
 **private** JTextField **t\_keyword** = **new** JTextField(8);  
 **private** JButton **b\_keyword** = **new** JButton(**"search"**);  
 **private** JPanel **panel\_3** = **new** JPanel(**new** GridLayout(1,3,10,15));  
  
 **private** JLabel **title\_3** = **new** JLabel(**"--------------------------功能按钮---------------------------------------------------------------------------------------------------------"**,JLabel.***CENTER***);  
 **private** JButton **b\_change** = **new** JButton(**"change"**);  
 **private** JButton **b\_delete** = **new** JButton(**"delete"**);  
 **private** JButton **b\_add** = **new** JButton(**"add"**);  
 **private** JButton **b\_save** = **new** JButton(**"save\_to\_file"**);  
 **private** JPanel **panel\_4** = **new** JPanel(**new** GridLayout(1,4,10,15));  
  
 AddressListGUI()  
 {  
 */\*\* 构造函数 \*/* DATA\_init();  
 GUI\_init();  
 Listen();  
 }  
  
 **private void** DATA\_init()  
 {  
 */\*\* 数据初始化 \*/* readTxtFile(**"DATA.txt"**);  
 *//writeTxtFile("DATA.txt");* }  
  
 **private void** readTxtFile(String filePath)  
 {  
 */\*\* 读取文件的数据到数组中 \*/* **try** {  
 String encoding=**"GBK"**;  
 File file=**new** File(filePath);  
 **if**(file.isFile() && file.exists())  
 { *//判断文件是否存在* InputStreamReader read = **new** InputStreamReader(  
 **new** FileInputStream(file),encoding);*//考虑到编码格式* BufferedReader bufferedReader = **new** BufferedReader(read);  
 String lineTxt = **null**;  
 **while**((lineTxt = bufferedReader.readLine()) != **null**)  
 {  
  
 AddressListData tmp = **new** AddressListData(lineTxt.split(**"-"**)[0], lineTxt.split(**"-"**)[1], lineTxt.split(**"-"**)[2],  
 lineTxt.split(**"-"**)[3], lineTxt.split(**"-"**)[4], lineTxt.split(**"-"**)[5]);  
 **V\_ALD**.add(tmp);  
 }  
 read.close();  
 }  
 **else** {  
 System.***out***.println(**"找不到指定的文件"**);  
 }  
 }  
 **catch** (Exception e)  
 {  
 System.***out***.println(**"读取文件内容出错"**);  
 e.printStackTrace();  
 }  
 }  
  
 **private void** writeTxtFile(String filePath)  
 {  
 */\*\* 把当前的数据写入到文件中 \*/* **try** {  
 *// 缓存写入文件* File file = **new** File(filePath);  
 **if** (!file.exists())  
 file.createNewFile();  
 FileWriter fw = **new** FileWriter(file.getAbsoluteFile());  
 BufferedWriter bw = **new** BufferedWriter(fw);  
  
 *//把当前数据数组写入文件* **int** Vl = **V\_ALD**.size();  
 **int** i;  
 **for**(i=0;i<Vl;i++)  
 {  
 AddressListData tmp;  
 tmp = **V\_ALD**.elementAt(i);  
 String str = **""**;  
 str = tmp.**name** + **"-"** + tmp.**adress** + **"-"** + tmp.**post\_code** + **"-"** + tmp.**phone** + **"-"** + tmp.**tel** + **"-"** + tmp.**email**;  
 *//System.out.println(str);* bw.write(str);  
 bw.newLine();  
 }  
 bw.close();  
 }  
 **catch** (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 **private void** GUI\_init()  
 {  
 */\*\*  
 \* GUI登录界面初始化  
 \*/* setTitle(**"Login"**); *// 标题* setLayout(**new** FlowLayout(FlowLayout.***LEADING***)); *// 设置为流窗格-竖排* setSize(630, 275); *// 大小* setResizable(**false**); *// 不能缩放* setLocationRelativeTo(**null**); *// 设置显示在屏幕中央* setVisible(**true**); *// 设为可见  
 // GUI设计-1* add(**title\_1**);  
 **panel\_1**.add(**l\_name**);  
 **panel\_1**.add(**t\_name**);  
 **panel\_1**.add(**l\_address**);  
 **panel\_1**.add(**t\_address**);  
 **panel\_1**.add(**l\_postcode**);  
 **panel\_1**.add(**t\_postcode**);  
 **panel\_1**.add(**l\_phone**);  
 **panel\_1**.add(**t\_phone**);  
 **panel\_1**.add(**l\_tel**);  
 **panel\_1**.add(**t\_tel**);  
 **panel\_1**.add(**l\_email**);  
 **panel\_1**.add(**t\_email**);  
 add(**panel\_1**);  
 *// GUI设计-2* add(**title\_2**);  
 **panel\_2**.add(**l\_index**);  
 **t\_index**.setText(**"0"**);  
 **panel\_2**.add(**t\_index**);  
 **panel\_2**.add(**b\_index**);  
 add(**panel\_2**);  
 **panel\_3**.add(**l\_keyword**);  
 **panel\_3**.add(**t\_keyword**);  
 **panel\_3**.add(**b\_keyword**);  
 add(**panel\_3**);  
 *// GUI设计-3* add(**title\_3**);  
 **panel\_4**.add(**b\_change**);  
 **panel\_4**.add(**b\_delete**);  
 **panel\_4**.add(**b\_add**);  
 **panel\_4**.add(**b\_save**);  
 add(**panel\_4**);  
 }  
  
 **private void** Listen()  
 {  
 */\*\* 按钮的事件监听类 \*/* **b\_index**.addActionListener(e ->  
 {  
 *// 按照索引查找* **int** Vsize = **V\_ALD**.size();  
 **int** index = Integer.*valueOf*(**t\_index**.getText()).intValue();  
 AddressListData tmp;  
 **if** (index<Vsize&&index>0)  
 {  
 tmp = **V\_ALD**.elementAt(index - 1);  
 **t\_name**.setText(tmp.**name**);  
 **t\_address**.setText(tmp.**adress**);  
 **t\_postcode**.setText(tmp.**post\_code**);  
 **t\_phone**.setText(tmp.**phone**);  
 **t\_tel**.setText(tmp.**tel**);  
 **t\_email**.setText(tmp.**email**);  
 }  
 }  
 );  
  
 **b\_keyword**.addActionListener(e ->  
 {  
 *// 按照关键词查找* **int** Vsize = **V\_ALD**.size();  
 **int** i;  
 AddressListData tmp;  
 **for**(i=0;i<Vsize;i++)  
 {  
 tmp = **V\_ALD**.elementAt(i);  
 **if**(tmp.search\_by\_keyword(**t\_keyword**.getText()))  
 {  
 **t\_name**.setText(tmp.**name**);  
 **t\_address**.setText(tmp.**adress**);  
 **t\_postcode**.setText(tmp.**post\_code**);  
 **t\_phone**.setText(tmp.**phone**);  
 **t\_tel**.setText(tmp.**tel**);  
 **t\_email**.setText(tmp.**email**);  
 **t\_index**.setText(String.*valueOf*(i+1));  
 }  
 }  
 }  
 );  
  
 **b\_change**.addActionListener(e ->  
 {  
 *// 修改* AddressListData tmp;  
 **int** index = Integer.*valueOf*(**t\_index**.getText()).intValue();  
 **int** Vsize = **V\_ALD**.size();  
 **if** (index<Vsize&&index>0)  
 {  
 **V\_ALD**.remove(index-1);  
 tmp= **new** AddressListData(**t\_name**.getText(),**t\_address**.getText(),**t\_postcode**.getText(),**t\_phone**.getText(),**t\_tel**.getText(),**t\_email**.getText());  
 **V\_ALD**.add(index - 1, tmp);  
 }  
 }  
 );  
  
 **b\_add**.addActionListener(e ->  
 {  
 *// 增加* AddressListData tmp;  
 tmp = **new** AddressListData(**t\_name**.getText(),**t\_address**.getText(),**t\_postcode**.getText(),**t\_phone**.getText(),**t\_tel**.getText(),**t\_email**.getText());  
 **V\_ALD**.add(tmp);  
 });  
  
 **b\_delete**.addActionListener(e ->  
 {  
 *// 删除* **int** index = Integer.*valueOf*(**t\_index**.getText()).intValue();  
 **int** Vsize = **V\_ALD**.size();  
 **if** (index<Vsize&&index>0)  
 **V\_ALD**.remove(index-1);  
 });  
  
 **b\_save**.addActionListener(e ->  
 {  
 *// 储存到文件* writeTxtFile(**"DATA2.txt"**);  
 }  
 );  
 }  
  
}

# 实验7 Java多线程

1. **实验目的**

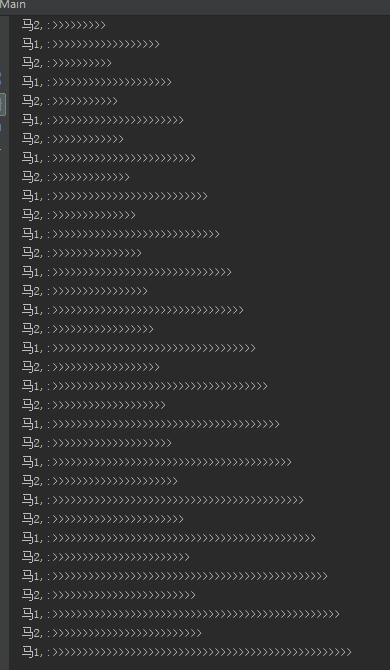
本实验要求学生能够理解Java多线程的概念，包括线程的生命周期、优先级和调度等，能够编写多线程的程序，

1. **必须学习和掌握的知识点**
2. 能够创建、管理、撤销线程。
3. 能够设置线程的优先级。
4. 能够控制线程执行过程。
5. **实验内容**

编写一个多线程的控制程序，称为赛马程序。创建分别代表两匹马的两个线程，并将它们设置为高低不同的优先级，并以进度条的形式显示赛马过程。

**package** Demo7;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/26.  
 \* 马类  
 \*/***public class** house **extends** Thread  
{  
 **private** String **name**; *// 马名* **private int velocity**; *// 速度* **private int distance**=0; *// 路程* **private static int** *journey* = 50; *// 全长* **private static long** *intervals* = 1; *// 间隔* house(String input\_name,**int** index)  
 {  
 */\*\* 构造函数 \*/* **name** = input\_name;  
 **velocity** = getSpeed();  
 setPriority(index); *//设置优先级* }  
 **private int** getSpeed()  
 {  
 */\*\* 获取随机速度  
 \** ***@return*** *返回一个1-3的随机整数作为速度  
 \*/* **return** (**int**)(Math.*random*()\*4+1);  
 }  
  
 **@Override  
 public void** run()  
 {  
 System.***out***.println();  
 */\*\* 重写父类run方法 \*/* **while**(**distance**<=*journey*)  
 {  
 **try** {  
 **this**.*sleep*(1000);  
 } **catch** (InterruptedException e)  
 {  
 e.printStackTrace();  
 }  
 **distance** += **velocity**;  
 **int** i;  
 System.***out***.print(**name**+**":"**);  
 **for**(i=0;i<**distance**;i++)  
 {  
 System.***out***.print(**">"**);  
 }  
 System.***out***.println();  
 }  
 }  
}

**package** Demo7;  
  
**public class** Main {  
 */\*\*  
 \* 编写一个多线程的控制程序，称为赛马程序。  
 \* 创建分别代表两匹马的两个线程，  
 \* 并将它们设置为高低不同的优先级，并以进度条的形式显示赛马过程。  
 \* \*/* **public static void** main(String[] args) {  
 *// write your code here* house h = **new** house(**"马1,"**,1);  
 house h2 = **new** house(**"马2,"**,1);  
 h.start();  
 h2.start();  
 }  
}



# 实验8 Java网络编程

1. **实验目的**

本实验要求学生掌握Java的网络编程方法，理解针对网络通信的不通层次(应用层、传输层和网络层)，Java所能提供的网络功能及方法。

1. **必须学习和掌握的知识点**
2. 掌握面向网络层的类InetAddress的使用方法。
3. 掌握面向应用层的类URL、URLConnection的使用方法。
4. 掌握面向网络层与TCP协议相关的类Socket和ServerSocket的使用方法。
5. **实验内容**

编写一个远程计算机监控程序，该程序可以监控网络上的多台其它计算机的资源使用情况，例如，CPU利用率、内存利用率以及硬盘利用率等。被监控的计算机称为计算节点，监控计算机称为管理节点。计算节点可以看作是客户端，管理节点可以看作是服务器端，客户端定时地向服务器端发送本机器的资源使用情况，服务器端接收客户端的数据并在监控界面上实时显示监控信息。

服务器和客户端要分开运行

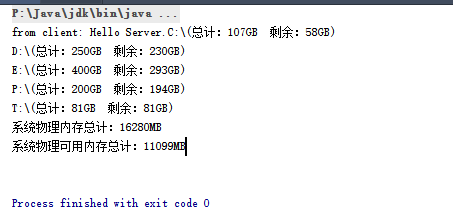
**package** Demo8;  
  
**import** java.io.IOException;  
  
**public class** Main {  
  
 **public static void** main(String[] args) **throws** Exception {  
 *// write your code here* Service s = **new** Service();  
 Client c = **new** Client();  
 }  
}

服务器

**package** Demo8;  
  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.io.Reader;  
**import** java.net.ServerSocket;  
**import** java.net.Socket;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/26.  
 //服务器端  
 \*/***public class** Service  
{  
 **private static int** *port* = 6013;  
 **private** ServerSocket **server** ;*//服务器socket* Service() **throws** IOException *//异常信息都往外抛* {  
 **server** = **new** ServerSocket(*port*);  
 Socket socket = **server**.accept(); *//阻塞式appect 直到有客户端连接* Reader reader = **new** InputStreamReader(socket.getInputStream());  
 **char** chars[] = **new char**[64];  
 **int** len;  
 StringBuilder sb = **new** StringBuilder();  
 **while** ((len=reader.read(chars)) != -1)  
 {  
 sb.append(**new** String(chars, 0, len));  
 }  
 System.***out***.println(**"from client: "** + sb);  
  
 *// 客户端退出后断开连接* reader.close();  
 socket.close();  
 **server**.close();  
 }  
}

客户端：

**package** Demo8;  
  
**import** com.sun.management.OperatingSystemMXBean;  
**import** com.sun.org.apache.xml.internal.serializer.OutputPropertiesFactory;  
**import** java.io.File;  
**import** java.io.OutputStreamWriter;  
**import** java.io.Writer;  
**import** java.lang.management.ManagementFactory;  
**import** java.net.Socket;  
**import** java.io.InputStreamReader;  
**import** java.io.LineNumberReader;  
**import** java.lang.management.ManagementFactory;  
**import** java.io.InputStreamReader;  
**import** java.io.LineNumberReader;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** com.sun.management.OperatingSystemMXBean;  
  
*/\*\*  
 \* Created by z.g.13@163.com on 2016/7/26.  
 \* 客户端  
 \*/***public class** Client  
{  
 **private static int** *port* = 6013;*//Port* **private static** String *host* = **"127.0.0.1"**; *//IP* Client() **throws** Exception *//异常都直接往外抛* {  
 *// 建立连接* Socket client = **new** Socket(*host*, *port*);  
 *// 发送数据* Writer writer = **new** OutputStreamWriter(client.getOutputStream());  
  
 writer.write(**"Hello Server."**);  
 writer.flush(); *// flush* **int** num\_D = *getDiskInfo*().split(**"<br>"**).**length**;  
 **int** i;  
 **for**(i=0;i<num\_D;i++)  
 {  
 writer.write(*getDiskInfo*().split(**"<br>"**)[i]);  
 writer.write(**"\n"**);  
 writer.flush();  
 *//System.out.println(getDiskInfo().split("<br>")[i]);* }  
 **int** num\_M = *getEMS*().split(**"<br>"**).**length**;  
 **int** j;  
 **for**(j=0;j<num\_M;j++)  
 {  
 writer.write(*getEMS*().split(**"<br>"**)[j]);  
 writer.write(**"\n"**);  
 writer.flush();  
 *//System.out.println(getEMS().split("<br>")[j]);* }  
 writer.close();  
 client.close();  
 }  
  
 **public static** String getDiskInfo()  
 {  
 */\*\*  
 \* 获取本地磁盘信息  
 \** ***@return*** *返回每隔硬盘的信息的字符串  
 \*/* StringBuffer sb=**new** StringBuffer();  
 File[] roots = File.*listRoots*();*// 获取磁盘分区列表* **for** (File file : roots)  
 {  
 **long** totalSpace=file.getTotalSpace();  
 **long** freeSpace=file.getFreeSpace();  
 **long** usableSpace=file.getUsableSpace();  
 **if**(totalSpace>0){  
 sb.append(file.getPath() + **"(总计："**);  
 sb.append(Math.*round*(((**double**)totalSpace/ (1024\*1024\*1024))\*100/100.0) + **"GB "**);  
 **if**(Math.*round*((((**double**)usableSpace/ (1024\*1024\*1024))\*100)/100.0)<=1){  
 sb.append(**"剩余："** + Math.*round*((((**double**)usableSpace/ (1024\*1024))\*100)/100.0) + **"MB)<br>"**);  
 }**else**{  
 sb.append(**"剩余："** + Math.*round*((((**double**)usableSpace/ (1024\*1024\*1024))\*100)/100.0) + **"GB)<br>"**);  
 }  
 *// sb.append("已使用" + Math.round((((double)(totalSpace-usableSpace)/(1024\*1024\*1024))\*100)/100.0) + "G<br>");* }  
 }  
 **return** sb.toString();  
 }  
  
 **public static** String getEMS()  
 {  
 */\*\*  
 \* 获取内存情况  
 \** ***@return*** *还有内存目前情况的字符串  
 \*/* StringBuffer sb=**new** StringBuffer();  
 OperatingSystemMXBean osmb = (OperatingSystemMXBean) ManagementFactory  
 .*getOperatingSystemMXBean*();  
 sb.append(**"系统物理内存总计："** + osmb.getTotalPhysicalMemorySize()  
 / 1024 / 1024 + **"MB<br>"**);  
 sb.append(**"系统物理可用内存总计："** + osmb.getFreePhysicalMemorySize()  
 / 1024 / 1024 + **"MB"**);  
 **return** sb.toString();  
 }  
}



客户端通信一次后断开时，服务器的截图