21世纪 高等学校计算机语言课 进阶辅导

权威推荐

Java语言程序设计

↘知识点全面覆盖

↘国内外精典实例剖析

↘典型习题拓展训练

《进阶辅导》编委会 组编

超值享受

- 全国计算机等级考试上机考试练习系统
  - 模拟考试，完全贴近真实上机考试环境
  - 仿真练习，全面熟悉考试真题
- 全国计算机等级考试笔试练习系统
  - 模拟考试，充分体验考试氛围
  - 仿真练习，通过练习历年真题掌握考试内容
- 书中涉及的源代码

大连理工大学出版社
大连理工大学电子音像出版社

《进阶辅导》编委会

（按笔画顺序排列）

王民生　任建娅　刘玉霞　刘晓红　李正光
李　昕　汪　静　谷晓琳　郎大维　金　花
闻丽华　梁　旭　梁　霞　黄　明　程智勇

Java 语言程序设计

《进阶辅导》编委会　组编

大连理工大学出版社
大连理工大学电子音像出版社　出版

地址：大连市凌水河　邮政编码：116024

电话：0411-84708842　传真：0411-84701466　邮购：0411-84707961

E-mail：dzcb@dutp.cn　URL：http://www.dutp.cn

大连理工印刷有限公司印刷　　大连理工大学出版社发行

幅面尺寸：185mm × 260mm　　印张：13.25　　字数：266 千字

2005 年 7 月第 1 版　　2005 年 7 月第 1 次印刷

责任编辑：刘　剑　高智银　　责任校对：达　理

封面设计：宋　蕾

ISBN 7-900670-47-5　　定价：21.80 元

前　言

在多年的教学实践中，我们总是听到学生反映学完“计算机语言课”后，实际编程还有很大的困难。我们认为，主要的问题就在于课内教材偏重于语句、语法的教学，实际应用方法的介绍、应用技能的培养非常有限，而练习题较简单，无法满足学生提高技能并达到实际应用的需求。为此，我们编写了这套《21世纪高等学校计算机语言课进阶辅导》丛书。

1.内容结构

共分为两个部分：

第一部分是程序设计学习指导。按内容分章节讲述，每章中都包含知识点归纳、例题精讲、拓展训练及其参考答案。概念清晰、讲解透彻、重点突出、适用性强，知识点全面覆盖。

第二部分是程序设计上机指导。包括编译环境介绍、编程技巧、应用实例、经典游戏等内容。

2.特点

(1)以内容难度适中、讲解深入浅出为基础，以切实帮助学生提高能力为出发点和根本目的。

(2)参加编写的人员都是多年从事计算机语言课教学、毕业设计指导、计算机语言培训的教师，他们既有丰富的实际开发经验，又真切地了解大多数学生在日常学习中的不足，所以本套丛书可以帮助学生解决学习中遇到的普遍性问题。

(3)在精选国内外实例的同时，对每个实例的解题思路均进行详细分析，对程序中的重点语句也通过注释的形式加以特殊强调。本套丛书力争做到指导性与可操作性相结合。

(4)书中的程序通过调试，可以直接在实际项目中使用。

3.适用范围

适用范围广泛,既可以作为计算机语言课的课程设计、毕业设计辅导用书,又可以作为学习软件开发的辅助用书,还可以作为参加计算机等级考试、软件水平考试的参考用书,同时还可以作为学员培训、程序员入门用书。

4.配套光盘

与其配套的光盘中附有书中涉及的源代码和资源文件,可供读者调试、运行。另外,还为读者提供了最新的"全国计算机等级考试练习系统",全真的模拟环境、历年的真题练习可以满足广大全国计算机等级考试考生的要求。

本套丛书由《进阶辅导》编委会组织编写。由于水平有限,书中错误和不妥之处在所难免,请读者和专家批评指正。

读者在使用过程中如有问题,可与编者联系:dllhm@263.net。

编　者

2005年7月

目　录

Note:

第一部分 Java语言程序设计学习指导

第1章

Java常用数据类型及其应用

1.1 知识点归纳

一、在归纳第一章的知识点之前，先简单介绍一下Java这种语言的历史与特点

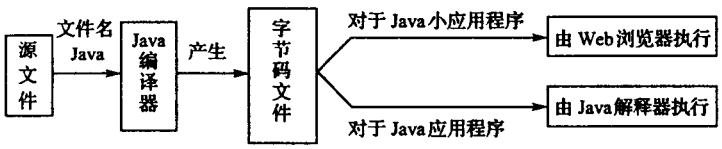
1.Java语言的历史

Java是1995年6月由美国的SUN公司开发出的面向对象的语言，同年它被美国著名杂志《PC Magazine》评为1995年十大优秀科技产品。由于其自身具有面向对象、与平台无关、安全、稳定与多线程等优点，Java语言日益受到业内外人士的关注。如今，很多大型的系统都是使用Java语言开发的。Java语言已经是开发Web应用程序、企业级系统的首选工具。

2.Java语言的特点

它具有简单、面向对象、稳定、与平台无关、解释型、多线程、动态等特点。

3.Java语言的开发过程



注:字节码文件是与平台无关的二进制码，再由解释器解释成本地机器码(解释一句，执行一句)。

① 编写源文件：使用一个文本编辑器，一般用记事本来编写源文件(不可使用Word编辑器，因它含有不可见字符)。将编好的源文件保存起来，被保存的源文件的扩展名必须是Java(你马上就会知道给源文件起名的规定)。

Note:

② 使用 Java 编译器(javac.exe):编译源文件得到字节码文件(你马上就会知道怎样使用 javac.exe)。

③ Java 程序分为两类——Java 应用程序和 Java 小应用程序:Java 应用程序必须通过 Java 解释器(java.exe)来解释执行其字节码文件;Java 小应用程序必须通过支持 Java 标准的浏览器来解释执行(你马上就会知道怎样使用解释器和浏览器来运行程序,普遍使用的 Netscape Navigator 和 Microsoft Internet Explorer 都完全支持 Java)。

每一种语言都有它自己的数据类型、运算符、表达式和语句,本章主要是对 Java 的基本数据类型进行介绍,数据类型是一门语言的基础,必须予以足够的重视。

二、Java 的基本数据类型有逻辑类型,字符类型与整数类型

1.逻辑类型

它包含两个常量:true 和 false。

逻辑类型的变量使用 Java 关键字 boolean 来定义。

例如:boolean isRight ;

```
boolean isRight , isOK ;
boolean isRight = true , isOK = false ;
```

2.字符类型

它使用 Unicode 字符集作为它的常量,也就是说它有 65535 个常量。

例如:'A' , 't' , '\n' , '@' , '通'。

字符类型的变量使用 Java 的关键字 char 来定义。

例如:char firstName ;

```
char firstName , middleName ;
char firstName = 'C' , middleName = 'K' ;
```

3.整数类型

它的常量包括十进制、八进制和十六进制。

例如:442(十进制)、064(八进制)、0xBA2(十六进制)。

整数类型的变量使用 Java 的关键字 int、byte、short、long 来定义。

① int 型

int 型变量,计算机内存将分配 4 个字节,共占 32 位,它的取值范围是 -2 147 483 648 ~ 2 147 483 647。

int 型的变量使用 Java 的关键字 int 来定义。

例如:int age ;

```
int age, grade ;
int age = 25 , grade = 2 ;
```

② byte 型

byte 型变量,计算机内存将分配 1 个字节,共占 8 位,它的取值范围是

Note:

-128～127。

byte 型的变量使用 Java 的关键字 byte 来定义。

例如:byte age ;

byte age, grade ;

byte age = 25 , grade = 2 ;

③short 型

short 型变量,计算机内存将分配 2 个字节,共占 16 位,它的取值范围是-32 768～32 768。

short 型的变量使用 Java 的关键字 short 来定义。

例如:short age ;

short age, grade ;

short age = 25 , grade = 2 ;

④ long 型

long 型变量,计算机内存将分配 4 个字节,共占 64 位,它的取值范围是-9 223 372 036 854 775 808～9 223 312 036 854 775 807。

long 型的变量使用 Java 的关键字 long 来定义。

例如:long age ;

long age, grade ;

long age = 25 , grade = 2 ;

4.浮点类型

浮点类型的变量使用 Java 的关键字 float、double 来定义。

①float 型

它的常量包括 987.65f 、12345f 等.

float 型变量,计算机内存将分配 4 个字节,共占 32 位,它的取值范围是1.4E-45～3.4028235E38。

float 型的变量使用 Java 的关键字 float 来定义。

例如:float sum ;

float sum, king ;

float sum = 555.55f , king = 200.00f ;

②double 型

它的常量包括 987.65d 、12345.09 等(d 可以省略)。

double 型变量,计算机内存将分配 8 个字节,共占 64 位,它的取值范围是4.9E-324～1.7976931348623157E308。

double 型的变量使用 Java 的关键字 double 来定义。

例如:double sum ;

double sum, king ;

Note:

```
double sum = 555.55 , king = 200.00 ;
```

三、以上便是 Java 的基本数据类型，除此之外，还要介绍两种常用的数据类型——数组与字符串

1.数组

数组是由相同数据结构的数据按顺序组成的一种复合型数据类型。它是通过数组名称和数组的下标来使用数组中的数据。

Java 中数组的下标从 0 开始，这点与 C 语言很相似。

声明数组要指定该数组的名称、数组包含的元素的数据类型。

例如：

```
boolean isBlack[ ] ;
char name[ ] ;
int age[ ] ;
byte viso[ ] ;
short grade[ ] ;
float sum[ ] ;
.....
```

若 char 是 Java 已经定义的类，那么可以定义它的数组：

```
char QQ[ ] ;
```

数组还包含二维甚至 N 维数组，这里用二维数组举例：

```
float king[ ][ ] ;
int grade[ ][ ] ;
.....
```

若 Book 是 Java 已经定义的类，那么可以定义它的二维数组：

```
Book computer[ ] [ ];
```

创建数组的过程实质上就是给数组分配内存的过程，因此，在创建数组时，必须指定数组的长度。

例如：

```
int ourAge [ ];
ourAge = new int[5];
int ourAge[ ] = new int[5];
```

2.字符串

Java 中使用 java.lang 包中的 String 类来创建字符串变量。

它的常量包括“atp”、“好好学习”等。

它的变量使用 String 类的构造方法来声明、创建。

例如：

```
String str ;
str = "I like Java !";
String name = "Jackey Cheng";
String text = String("it is Li speaking");
```

以上便是对 Java 常用数据类型相关知识的总结。

Note:

1.2 例题精讲

【例题 1】 将由终端输入的由逗号或空格分隔的姓名解析出来，将首字母大写，并显示出来，当终端输入为 exit 时，程序结束。

精讲 本题的实质就是先将字符串中的姓名提取出来，然后再对提取出的子字符串的首字母进行比较，按升序排列输出。

程序流程如下：

1.接受终端字符输入，判断是否等于 exit，若相等则转到 5 否则继续 2；

2.先使用 Java 类库中的 StringTokenizer 类分析字符串。并将其分解成独立的姓名；

3.对提取出的子字符串的首字母进行大写转换；

4.再将转换好的姓名格式输出，然后转到 1 继续；

5.结束。

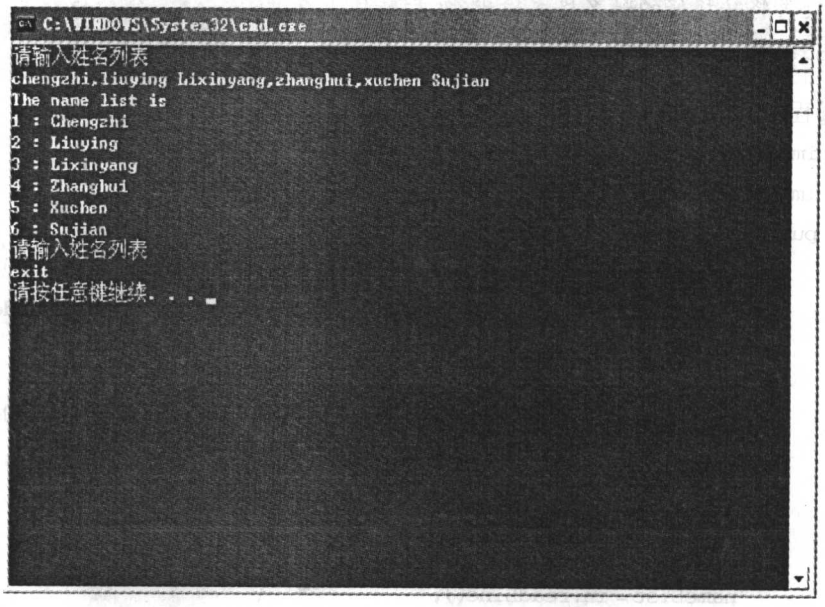
【程序】

```
import java.util.*;
import java.io.*;
public class NameList {
   public static void main(String args[]) throws IOException{
     BufferedReader in = new BufferedReader(new InputStreamReader
     (System.in));
     String nameList = new String(" ");
    //无限循环
    while(true){
      System.out.println("请输入姓名列表");
      //接受用户输入
      nameList = in.readLine();
      if((nameList = = null) || nameList.equals("exit")) {
          break; }
      //由逗号或空格做分隔符
     StringTokenizer token = new StringTokenizer(nameList,",");
      int cnt = token.countTokens();
      int inx = 0;
      String name[] = new String[cnt];
      while(token.hasMoreTokens()) {
          //存入字符串数组
         name[inx] = token.nextToken();
```

Note:

```
            inx++;  }
        System.out.println(" The name list is ");
        for(int i=0;i<cnt;i++){
            String head=new String("");
            String tail=new String("");
            head=name[i].substring(0,1);
            tail=name[i].substring(1,name[i].length());
            //姓名的首位转为大写
            head=head.toUpperCase();
            //首尾字符串相连
            head=head.concat(tail);
            name[i]=head;
            System.out.println((i+1)+" : "+name[i]);  }  }  }  }
```

【运行结果】



【例题 2】 将一个 5×5 的矩阵中最大的元素放在中心，四个角分别放四个最小的元素(顺序为从左到右，从上到下依次从小到大存放)，写一个函数实现它。

精讲 本题看似复杂，其实不难。实质就是在一个二维数组中求四个最小的数及一个最大的数，然后把它们放在指定的位置上。

首先需要确定这五个位置，左上角对应的是 a[0][0]，存放最小的元素；右上角对应的是 a[0][4]，存放第二小的元素；左下角对应的是 a[4][0]，存放第三小的元素；右下角对应的是 a[4][4]，存放第四小的元素；中间的元

Note:

素对应的是 a[2][2],存放最大的元素。

程序流程如下:

1.逐个元素比较,找到最大元素,与 a[2][2]交换位置;

2.找最小的元素,与 a[0][0]交换位置;

3.找除了第 2 步之外的最小元素,与 a[0][4]交换位置;

4.找除了第 2 步、第 3 步之外的最小元素,与 a[4][0]交换位置;

5.找除了第 2 步、第 3 步、第 4 步之外的最小元素,与 a[4][4]交换位置;

6.结束。

【程序】

```
import java.io.*;
import java.util.*;
public class Sort {
   public static void main(String args[]){
      int temp=0;
      int num[][]=new int[5][5];
      String strTemp="";
      String readStr="";
      StringTokenizer token;
      //从文件中读取数据,并存入字符串中
      try {
            File file=new File("in.txt");
            FileReader infile=new FileReader(file);
            BufferedReader in=new BufferedReader(infile);
            while((strTemp=in.readLine())!=null) {
              readStr=strTemp; }
            infile.close();
            in.close(); }
      catch(Exception e0) {
            System.out.println("读取文件错误"); }
      //对字符串进行分析,分解出 25 个数字
      token=new StringTokenizer(readStr,",");
      int cnt=token.countTokens();
      if (cnt!=25) { System.out.println("数字个数错误"); }
      else {
            int idx=0;
            int line=0;
            int row=0;
            String str="";
            while(token.hasMoreTokens()){
```

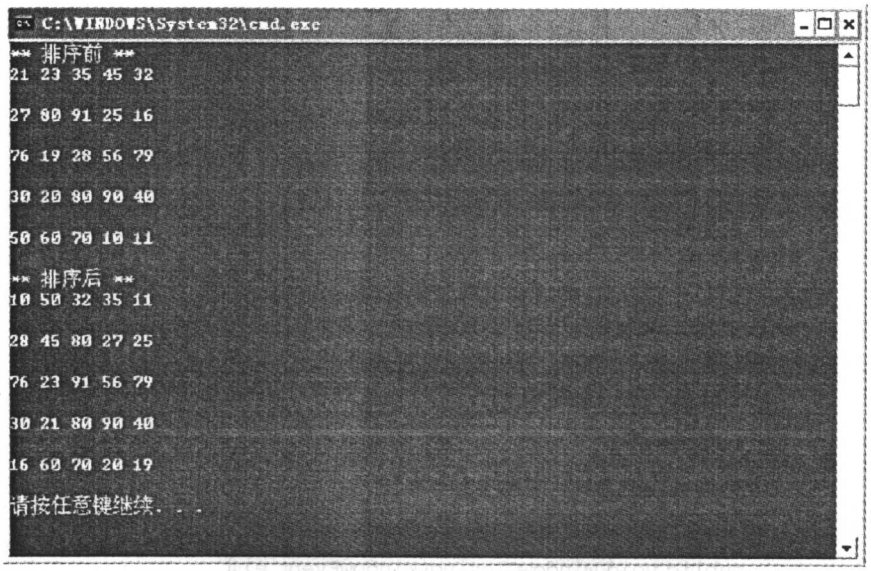
Note:

```
            str = token.nextToken();
            line = idx/5;
            row = idx % 5;
            num[line][row] = Integer.parseInt(str);
            idx + + ;  }
        //排序前将数组输出,以便于比较
        System.out.println("* * 排序前 * *");
        for(int i = 0;i < 5;i + + ){
          for(int j = 0;j < 5;j + + ) {
            System.out.print(num[i][j] + " ");  }
          System.out.println("\n");  }  }
//获取最大值,并将其放入 num[2][2]中
for(int i = 0;i < 5;i + + ) {
      for(int j = 0;j < 5;j + + ) {
        if(num[2][2] < num[i][j]) {
          temp = num[2][2];
          num[2][2] = num[i][j];
num[i][j] = temp;  }  }  }
//获取最小值,并将其放入 num[0][0]中
for(int i = 0;i < 5;i + + ) {
      for(int j = 0;j < 5;j + + ) {
        if(num[0][0] > num[i][j]) {
          temp = num[0][0];
          num[0][0] = num[i][j];
          num[i][j] = temp;  }  }  }
//获取第二小值,并将其放入 num[0][4]中
for(int i = 0;i < 5;i + + ) {
      for(int j = 0;j < 5;j + + ) {
        if(i = = 0&&j = = 0){
          continue;  }
        if(num[0][4] > num[i][j]) {
          temp = num[0][4];
          num[0][4] = num[i][j];
         num[i][j] = temp;  }  }  }
//获取第三小值,并将其放入 num[4][0]中
for(int i = 0;i < 5;i + + ) {
      for(int j = 0;j < 5;j + + ) {
        if(i = = 0&&(j = = 0||j = = 4)) {
          continue;  }
        if(num[4][0] > num[i][j]) {
          temp = num[4][0];
          num[4][0] = num[i][j];
```

Note:

```
                num[i][j] = temp;  } } }
        //获取第四小值,并将其放入 num[4][4]中
        for(int i = 0;i < 5;i + + ) {
              for(int j = 0;j < 5;j + + ) {
                if((i = = 0||i = = 4)&&(j = = 0||j = = 4)) {
                  continue;  }
                if(num[4][4] > num[i][j]) {
                  temp = num[4][4];
                  num[4][4] = num[i][j];
                  num[i][j] = temp;  } } }
        //将排序后的结果输出
        System.out.println("* * 排序后 * *");
        for(int i = 0;i < 5;i + + ){
              for(int j = 0;j < 5;j + + ) {
                System.out.print(num[i][j] + " ");  }
              System.out.println("\n");  } }
```

【运行结果】



【例题 3】　将文本文件中的 N 组数字列进行比较，若两组数字列只有一位不同，则将两组数字列合并，反复合并，直到不能再合并为止。

精讲　本题看似复杂，其实不难，这里涉及到一个算法，如看两组数字是否只有一位不同，可将两数相减，若差值只有一位不为零的话，那么就说明两数只有一位不同。我们可以根据这个算法对数据进行比较，符合的话就将两个数列同一位上不同的两个数相加，将改变后的结果放到一个数列中，去掉另一个数列就行了。然后反复使用该方法进行合并。

程序流程如下：

Note:

1.将文件中的数列读到字符数组中；

2.比较是否有符合合并条件的数列，若没有的话转到 4 结束，否则转到 3 继续；

3.将符合条件的数列合并，并在文本框中输出数列的条数，并转到 1；

4.结束。

【程序】

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
class Ball extends JFrame implements ActionListener{
   JButton OKBtn=new JButton("OK", new ImageIcon("pic.gif"));
   JTextArea text=new JTextArea("",20,25);
   Container mainPane=getContentPane();
   Ball(){
    //初始化界面
    setVisible(true);
    setBounds(100,100,300,300);
    mainPane.setLayout(new GridLayout(1,2));
    OKBtn.setHorizontalTextPosition(AbstractButton.CENTER);
    OKBtn.setVerticalTextPosition(AbstractButton.BOTTOM);
    JScrollPane scrollpane=new JScrollPane(text);
    mainPane.add(scrollpane);
    mainPane.add(OKBtn);
    //响应鼠标事件
    OKBtn.addActionListener(this);
    addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent e){
        System.exit(0);  }  });  }
   public void actionPerformed(ActionEvent e){
    if(e.getSource()==OKBtn){
     int rowCnt[]=new int[2];
     do{
       rowCnt= union();}
     while(rowCnt[0]!=rowCnt[1]);  }  }
   //数列合并方法
```

Note:

```
public int[] union(){
 int cntTemp[] = new int[2];
 String strTemp = "";
 //确定文件中数列的条数
 try {
   File file = new File("test.txt");
   FileReader infile = new FileReader(file);
   BufferedReader in = new BufferedReader(infile);
   while((strTemp = in.readLine())! = null) {
     cntTemp[0] + + ;  }
   infile.close();
   in.close();}
 catch(Exception e01) {}
 String temp[] = new String[cntTemp[0]];
 int cnt = 0;
 //将文件中的数列读到字符串数组中
 try {
   File file = new File("test.txt");
   FileReader infile = new FileReader(file);
   BufferedReader in = new BufferedReader(infile);
   while((strTemp = in.readLine())! = null) {
     temp[cnt + + ] = strTemp;  }
   infile.close();
   in.close();}
 catch(Exception e02) {}
 for(int i = 0;i < cnt;i + + ){
   for(int j = i + 1;j < cnt;j + + ){
     if(! temp[i].equals("BeThrowed")) {
       //使用上面讲述的算法进行合并
       long varTemp1 = 0;
       long varTemp2 = 0;
       long varTemp3 = 0;
       long length = 0;
       long mod = 1;
       varTemp1 = Long.parseLong(temp[i]);
       varTemp2 = Long.parseLong(temp[j]);
       varTemp3 = varTemp1 - varTemp2;
```