# JAVA入门

## 1.1JAVA语言基础

### 1.1.1Java概述

1     什么是编程语言？

将人的思想传达给计算机的工具

2.        编程语言的发展

第一代语言：机器语言（打孔机）

第二代语言：汇编语言 MOV A B;

第三代语言：C所有高级语言的基础，面向过程的语言

                   C++面向过程/对象语言

            Java完全面向对象的跨平台语言

                  .net跨语言的面向对象平台

3.        Java发展史：

Java语言出现于1991年，JDK1.0版本正式发布于1995年

Java的摇篮：SUN  Stanford University NetWork

Java之父：James Gosling(詹姆斯•高斯林)

JDK1.0    JDK1.1    JDK1.2    JDK1.3    JDK1.4    JDK5.0   JDK6.0

Java最新版本JDK7.0

4.        Java技术架构体系：

a)        J2SE：桌面应用程序

b)        J2EE：web应用程序

c)        J2ME：手机应用程序

d)       Android-SDK：Android应用程序开发

5.        Java特点：

a)        Java是一种面向对象的编程语言

b)        Java语言是一种编译解释型语言

c)        因为Java语言是基于Java虚拟机运行，所以Java语言可以实现跨平台

d)       Java的GC垃圾回收器

### 1.1.2.JDK环境变量设置

1、             在系统变量里点击新建，变量名填写**JAVA\_HOME**，变量值填写JDK的安装路径，比如：“**C:\Program Files\Java\jdk1.6.0\_26**”了。

2、             在系统变量里点击新建变量名填写**CLASSPATH**，变量值填写“**.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar**”。注意不要忘记前面的点和中间的分号。

3、             在系统变量里找到Path变量，这是系统自带的，不用新建。双击**Path**，由于原来的变量值已经存在，故应在已有的变量后加上“**;%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin**”。注意前面的分号。

### 1.2.3命名规范

       类名

a)       要选取能够表达类的作用的英文单词组成

b)      类名首字母要大写，如果有多个单词构成则**每个单词的首字母都大写**（大小写混排）

**c)** 类名可以由“**字母**”、“**数字**”、“**\_**”和“**$**”组成，但是**不能够以数字开头**

d)      类名允许使用中文，但是不提倡使用

     包名：多单词组成时所有字母都小写。

•       xxxyyyzzz

     类名接口名：多单词组成时，所有单词的首字母大写。

•       XxxYyyZzz

     变量名和函数名：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写。

•       xxxYyyZzz

     常量名：所有字母都大写。多单词时每个单词用下划线连接。

•       XXX\_YYY\_ZZZ

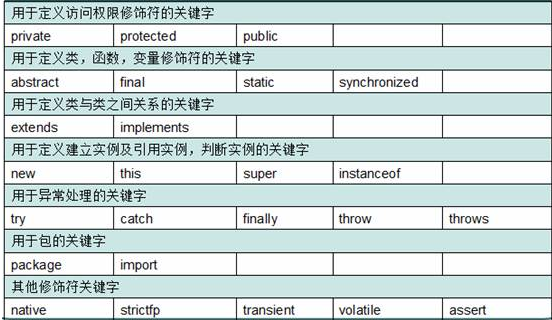
### 1.1.4关键字

关键字的定义和特点

定义：被Java语言赋予了特殊含义的单词

特点：关键字中所有字母都为小写





保留字:goto   const

### 1.1.5注释

     用于注解说明解释程序的文字就是注释。

     提高了代码的阅读性。

     Java中的注释格式：

•       单行注释

•       格式： **//** 注释文字

•       多行注释

•       格式： **/\***  注释文字  **\*/**

•       文档注释

•       格式：**/\*\*** 注释文字 **\*/**

### 1.1.6 常量与变量

                 常量表示不能改变的数值。

                 Java中常量的分类：

1，整数常量。所有整数

2，小数常量。所有小数

3，布尔(boolean)型常量。较为特有，只有两个数值。true false。

4，字符常量。将一个数字字母或者符号用单引号( ' ' )标识。

5，字符串常量。将一个或者多个字符用双引号**(“ ”)**标识。

6，null常量。只有一个数值就是:null.

     对于整数：有四种表现形式。

•       二进制：0,1 ，满2进1.

•       八进制：0-7 ，满8进1. 用**0**开头表示。

•       十进制：0-9 ，满10进1.

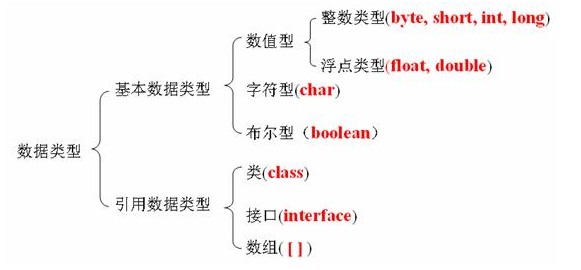
•       十六进制：0-9，A-F，满16进1. 用**0x**开头表示。

常量: 常量在定义的时候要用**final**修饰，同时初始化。常量名全部大写，如果有多个单词用“**\_**”隔开。

final float PI=3.1415;

### 1.1.7 数据类型

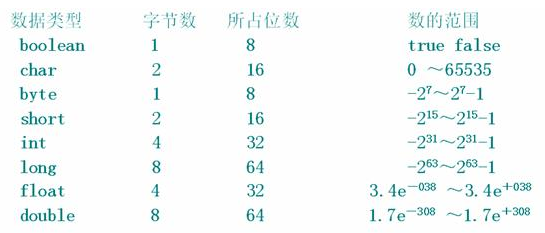
Java语言是强类型语言，对于每一种数据都定义了明确的具体数据类型，在内存中分配了不同大小的内存空间



**注：整数默认：int    小数默认：double**

long a=8888l(L);

float b=8888.88f(F)



### 1.1.8类型转换

     **自动类型转换**（也叫隐式类型转换）

     **强制类型转换**（也叫显式类型转换）

     类型转换的原理

     什么时候要用强制类型转换?

     表达式的**数据类型自动提升**

•       所有的byte型、short型和char的值将被提升到int型。

•       如果一个操作数是long型，计算结果就是long型；

•       如果一个操作数是float型，计算结果就是float型；

•       如果一个操作数是double型，计算结果就是double型。

     分析

•       **System.out.println(‘a’)与System.out.println(’a’+1)** 的区别。a           98

       自动类型提升

**byte b = 3;**

**int x = 4;**

**x = x + b;**//b会自动提升为int类型进行运算。

       强制类型转换

**byte b = 3;**

**b = b + 4;**//报错

**b = (byte)(b+4);**//强制**类型**转换，强制将b+4的结果转换为byte类型，再赋值给b。

                 思考：

**byte b1=3,b2=4,b;**

**b=b1+b2;**

**b=3+4;**

**哪句是编译失败的呢？为什么呢？**

**算术运算符与赋值运算符**

**short** s=3;

s+=4;  **//赋值运算，底层自动做了强转动作，自动完成**

// s=s+4;  //算术运算，会丢失精度，需强转，否则编译出错

System.*out*.println(s);

### 1.1.9运算符

    **分割符：,，;，[]，()**

    **算术运算符:  +，―，\*，/，%，++，――**

    **关系运算符:  >，<，>=，<=，==，!=**

    **布尔逻辑运算符:  !，& , | , ^ , &&，||**

    **位运算符:  &，|，^，~ ， >>，<<，>>>**

    **赋值运算符:  =  扩展赋值运算符:+=，―=，\*=，/=**

    **字符串连接运算符: +**

    **造型操作符：()**

    **三目运算符：？:**

&&：

**和&运算结果是一样的，但是运算过程有点小区别。**

**&：无论左边的运算结果是什么，右边都参与运算**

**&&：当左边为false时，右边不参与运算**

||：

**和|运算结果是一样的，但是运算过程有点小区别。**

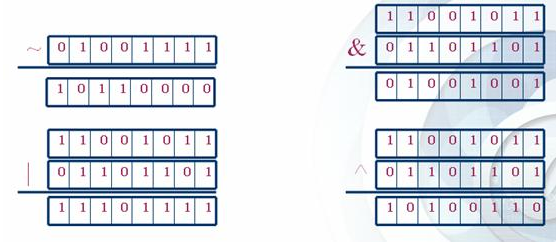
**|：无论左边的运算结果是什么，右边都参与运算**

**||：当左边为true时，右边不参与运算**

    **位运算符功能**

**~ -- 取反       &   -- 按位与**

**|  -- 按位或       ^  -- 按位异或**



**^ ：异或，和或有点不一样**

**^ :运算特点。**

**true ^ true=flase**

**true ^ flase = true**

**flase ^ true= true**

**flase ^ flase =flase**

**^异或的运算规律：**

**^异或的两边同为true或者flase，结果为false**

**两边结果不一样，结果为true**

    **>>：右移运算符，将左操作数向右移动，移位个数由右操作数决定**

    **<<：左移运算符，将左操作数向左移动，移位个数由右操作数决定**

    **>>>：无符号左移位运算符**

**int** a = 8;

**int** b = a >> 1; // 4

**int** c = a << 1; // 8

**int** i = 87;

**int** j = 63;

**int** k = -9;

      // -9 : 11111111 11111111 11111111 11110111.

      // 9 : 00000000 00000000 00000000 00001001.

      // 有符号右移

      // -9>>3: [111]11111111 11111111 11111111 11110.

      // 9>>3: [000]00000000 00000000 00000000 00001

      // 无符号右移

      // -9>>>3:[000]11111111 11111111 11111111 11110.

      // 9>>>3: [000]00000000 00000000 00000000 00001

      i ^ j;

      ~i;

      k << 3;// 左移n位(二进制的右边补n个0，左边去掉n位)：对原来的数值扩大2的n次方倍

      k >> 3;// 右移n位(二进制右面去掉n位，左边补上n个符号位)

      k >>> 3;// 右移n位(二进制右面去掉n位，左边补上个0)

**<<:左移几位其实就是该数据乘以的几次幂**

**例：3<<1=6  3<<2=12  3<<3=24**

**>>:右移几位，其实就是该数据除以2的几次幂**

**练习1：最有效率的方式算出2乘以8等于几？**

**解：  2<<3**

**练习2：对两个整数变量的值进行互换（不需要第三方变量）**

**方法一：          int a=3,b=5**

**a=a+b; b=a-b; a=a-b;**

**这种方法不要用，如果两个整数的数值过大，会超出int范围，会强制转换，数据会变化。**

**方法二：(面试常用)       a=a^b; b=a^b; a=a^b;**

**1.1.10**break和continue

**break:跳出**

break作用的范围：要么是switch语句，要么是循环语句。

记住：当break语句单独存在时，下面不要定义其他语句，因为执行不到。

break跳出当前循环

**例：**

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

      System.*out*.println("abc");

**break**;

      System.*out*.println("def");

}

**编译错误：无法访问的语句**

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

**^**

**continue：继续。**

作用的范围：循环结构。

作用：结束本次循环，继续下次循环。

如果continue单独存在时，下面不要有任何语句，因为执行不到。

break语句：应用范围：选择结构和循环结构。

continue语句：应用于循环结构。

### 1.1.11for

**class** ForDemo {

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

**int** k = 5;

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

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

           System.*out*.print(" ");

        }

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

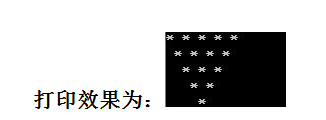
           System.*out*.print("\* ");

        }

        System.*out*.println();

      }

   }

}

#### 1.1.12补充

<<:左移几位其实就是该数据乘以的几次幂

      例：3<<1=6  3<<2=12  3<<3=24

>>:右移几位，其实就是该数据除以2的几次幂

练习1：最有效率的方式算出2乘以8等于几？

           解：    2<<3

练习2：对两个整数变量的值进行互换（不需要第三方变量）

方法一：       int a=3,b=5

                      a=a+b; b=a-b; a=a-b;

这种方法不要用，如果两个整数的数值过大，会超出int范围，会强制转换，数据会变化。

方法二：(面试常用)       a=a^b; b=a^b; a=a^b;

&&：

      和&运算结果是一样的，但是运算过程有点小区别。

      &：无论左边的运算结果是什么，右边都参与运算

      &&：当左边为false时，右边不参与运算

||：

      和|运算结果是一样的，但是运算过程有点小区别。

      |：无论左边的运算结果是什么，右边都参与运算

      ||：当左边为true时，右边不参与运算

^ ：异或，和或有点不一样

^ :运算特点。

true ^ true=flase

true ^ flase = true

flase ^ true= true

flase ^ flase =flase

^异或的运算规律：

^异或的两边同为true或者flase，结果为false

两边结果不一样，结果为true

# 第二章、数组

### 1.2.1一维数组

    数组是用来存储一组相同数据类型数据的数据结构。

    数组也是一种引用类型。

    在数组定义中，不能指定数组的长度，而需要在数组的创建阶段来指定

    **静态初始化**

**int**[] a = {1,3,5,7,9,11};

     String[] s = { **new** String("abc"),

**new** String("123"),

**new** String("xyz")};

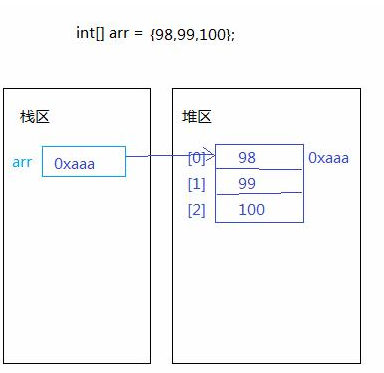
    动态初始化

     int[] a = new int[10];

     for (int i = 0; i < 10; i++) {

        a[i] = 2 \* i + 1;

     }



“改变数组的长度”  System

|  |  |
| --- | --- |
| static void | [arraycopy](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\API.CHM::/java/lang/System.html#arraycopy(java.lang.Object, int, java.lang.Object, int, int))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\API.CHM::/java/lang/Object.html) src, int srcPos, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\API.CHM::/java/lang/Object.html) dest, int destPos, int length)            从指定源数组中复制一个数组，复制从指定的位置开始，到目标数组的指定位置结束。 |

      // 声明并创建一个长度为5的 int 类型的数组

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

      System.*out*.println("改变长度前:" + srcArry.length);

      // 改变数组长度

      // ①,创建一个新数组(在原有长度上 +1)

      int destArray[] = new int[srcArry.length + 10];

// ②,将 srcArray 中的值对应拷贝到 a 中

      System.*arraycopy*(srcArry, 0, destArray, 0, srcArry.length);

      // ③,将新数组赋给 srcArray

      srcArry = destArray;

      System.*out*.println("改变长度后:" + srcArry.length);

### 1.2.2二维数组

创建二维数组

创建一个长度为4的大数组,再创建4个长度为5的小数组,

将4个小数组分别赋值给大数组的四个元素

**int** arr[][] = **new** **int**[4][5];

创建一个长度为3的大数组

**int** arr2[][] = **new** **int**[3][];

arr2[0] = **new** **int**[5];

arr2[1] = **new** **int**[4];

arr2[2] = **new** **int**[2];

遍历二维数组

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

**for** (**int** j = 0; j < arr[i].length; j++) {

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

   }

   System.*out*.println();

}

排序

#### 1，            选择排序

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

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

**if** (arr[i] > arr[j]) {

**int** tem = arr[i];

        arr[i] = arr[j];

        arr[j] = tem;

      }

   }

}

#### 2，        冒泡排序

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

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

      if (arr[j] > arr[j + 1]) {

        int tem = arr[j];

        arr[j] = arr[j + 1];

        arr[j + 1] = tem;

      }

   }

}

②  public static void selectSort\_2(int arr[]) {

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

        int num = arr[i];

        int index = i;

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

           if (num > arr[j]) {

              num = arr[j];

              index = j;

           }

           if (index != i)

*swap*(arr, index, i);

        }

      }

   }

### 1.2.3快速排序

**public** **class** QuickSort {

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

**int** arr[] = { 34, 35, 18, 92, 46, 28, 55, 73, 64 };

*quick*(arr, 0, arr.length - 1);

**for** (**int** i : arr) {

        System.*out*.print(i + "\t");

      }

   }

**public** **static** **void** quick(**int** arr[], **int** start, **int** end) {

**int** i = start;

**int** j = end;

**int** temp = arr[start];

**while** (i < j) {

// 从后往前比较

**for** (; j > i; j--) {

**if** (arr[j] < temp) {

              arr[i] = arr[j];

              i++;

**break**;

           }

        }

**for** (; i < j; i++) {

**if** (arr[i] > temp) {

              arr[j] = arr[i];

              j--;

**break**;

           }

        }

        arr[i] = temp;

**if** (i > start + 1) {

*quick*(arr, start, i - 1);

        }

**if** (j < end - 1) {

*quick*(arr, j + 1, end);

        }

      }

   }

}

### 1.2.4二分查找

二分查找法（折半查找法）：

前提：数组必须是有序的

**public** **static** **int** halfSearch(**int** arr[],**int** key)

{

**int** max,min,mid;

   min=0;max=arr.length-1;mid=(max+min)/2;

**while**(key!=arr[mid])

   {

**if**(key>arr[mid])

        min=mid+1;

**else**

        max=mid-1;

**if**(max<min)

**return** -1;

      mid=(max+min)/2;

   }

**return** mid;

### 1.2.5栈内存和堆内存

栈内存：

           存储的都是局部变量。

           而且变量所属的的作用域一旦结束，该变量就自动释放。

堆内存：

           存储是数组和对象（其实数组就是对象），凡是new建立在堆内存中。

特点：

1，          每一个实体都有首地址值。

2，          堆内存中的内一个变量都有默认的初始值。

整数：0   小数：0.0或者0.0f   Boolean：false

Char：’\u0000’

### 1.2.6数组工具类 文档注释

Java注释文档的注释方法：/\*\*   \*/

Javac的用法：

javac <选项> <软件包名> <源文件>

注意：如果把一个类文档化，该类必须是public.

javadoc –d . –author –version ArrayTool.java

javadoc –d D:\myclass –author –version ArrayTool.java

/\*\*

 \* 建立一个用于操作数组的工具类，其中包含着常见的对数组操作的函数：最值，排序等。

 \*

 \* @author 康吉

 \* @version V1.0

 \*/

public class ArrayTool {

   private ArrayTool() {

   }

   /\*\*

    \* 获取整型数组的最大值

    \*

    \* @param arr

    \*            接收一个元素为int类型的数组

    \* @return 该数组的最大元素值

    \*/

   public static int getMax(int arr[]) {

      int maxIndex = 0;

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

        if (arr[maxIndex] > arr[i])

           maxIndex = i;

      }

      return arr[maxIndex];

   }

   /\*\*

    \* 对数组进行选择性排序

    \*

    \* @param arr

    \*            接收一个元素为int类型的数组

    \*/

   public static void selectSort(int arr[]) {

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

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

           if (arr[i] > arr[j])

*swap*(arr, i, j);

        }

      }

   }

   /\*\*

    \* 用于给数组进行元素位置的置换

    \*

    \* @param arr

    \*            接收一个元素为int类型的数组

    \* @param i

    \* @param j

    \*/

   public static void swap(int arr[], int i, int j) {

      int tem = arr[i];

      arr[i] = arr[j];

      arr[j] = tem;

   }

   /\*\*

    \* 获取指定的元素在指定数组中的索引

    \*

    \* @param arr

    \*            接收一个元素为int类型的数组

    \* @param key

    \*            要找的元素

    \* @return 返回该元素第一次出现的位置，如果不存在返回-1

    \*/

**public** **static** **int** getIndex(**int** arr[], **int** key) {

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

**if** (key == arr[i])

**return** i;

      }

**return** -1;

   }

   /\*\*

    \* 将int数组转换成字符串，格式是：[e1,e2....]

    \*

    \* **@param** arr

    \*            接收一个元素为int类型的数组

    \* **@return** 返回还数组的字符串表现形式

    \*/

**public** **static** String arrayToString(**int** arr[]) {

      String str = "[";

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

**if** (i == arr.length - 1)

           str = str + arr[i] + "]";

**else**

           str = str + arr[i] + ",";

      }

**return** str;

   }

}

#### 总结

 1、在命令控制台，编译Java文件的命令是\_\_\_\_\_javac\_\_\_\_\_\_\_\_\_\_\_。

2、Java源文件和编译后的文件扩展名分别为\_\_\_java\_\_、\_\_\_class\_\_\_\_

3、java主函数（程序入口）的4要素是\_\_\_public\_\_\_\_\_、\_\_\_\_static\_\_\_\_\_\_\_\_\_\_、\_\_\_\_void\_\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_main\_\_\_\_\_\_。

4、变量的三要素是：\_\_变量类型\_\_\_\_\_\_\_\_\_、变量名、 \_\_\_\_\_\_变量值\_\_\_\_\_\_\_。

5、算术运算符有（写出5个）：\_\_\_\_\_\_+ - \_\_\*\_\_\_/\_\_\_%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

6、java的数据类型有（最少5个）：\_\_整型\_\_浮点型\_\_布尔型\_\_枚举型\_\_引用类型\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

7、java的注释有：\_//\_(单行注释)\_\_\_\_\_、\_/\* \*/（多行注释）\_\_\_\_\_\_\_\_\_\_、\_\_/\*\* \*/(文档注释)\_\_\_\_\_\_\_\_。

8、break和continue的区别：\_\_break 跳出此循环\_\_\_\_\_\_\_continue 结束本次循环，进入下一次循环\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

9、手写代码，打印等腰三角形（5分）：

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

**for**(**int** j=5;j>i;j--){

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

}

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

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

}

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

}

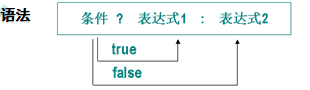
10、如何进行程序调试（3个步骤）：\_\_设置断点\_\_\_\_\_\_\_\_\_、\_\_单步运行\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_观察变量\_\_\_\_\_\_\_\_\_\_。

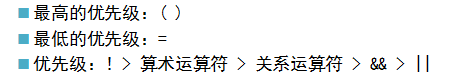
11、写出for循环的语法：\_\_for(初始值;判断条件;迭代值)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

12、if和switch分别在什么场景运用：\_\_\_\_\_if用于区间的选择，switch用于等值的判断\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。



**三目运算符**







**产生（0~5） (int)(Math.random()\*6)**

**其中（Math.random()）产生的是0~0.99999**

**从键盘上输入：**

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

**int a=in.nextInt();**

**String s=in.next();**

**double a1=in.nextDouble();**

**hassNext(); 判断是否输入，**

**for双重循环：1.外面控制行数，2.里面控制每一行的情况**

**数组定义：数组是一个变量，存储相同数据类型的一组数据**

**定义数组的方法：**

**数组基本要素：数组类型，数组名，数组值，数组下标**

**数组长度固定不变，使用时要避免数组下标越界**

**声明数组 int[] score; score=new int[3];**

**int[] score={2,4,5};**

**int[] score=new int[30];(一定要定义长度)**

**int[] score=new int[]{3,4,5}**

**for(int i: score){**

**} 将数组score循环赋给i;**

**+只要两边有一边是字符串，即为连接符，否则就是算术运算符。**

**数组排序：**

/\*\*

\* 插入。。最大到最小排列

\*/

1.定义一个新的数组（因为数组固定大小，不能改变）

2.依次比较，找到最可插入的位置，记录该下标index，并且防止继续往下面找，

使用break。

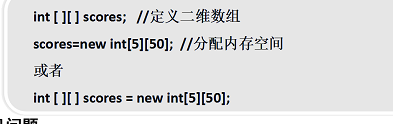
3.然后从最后面，开始，将所有的元素往后移一位，

4.最后插入、将输入的值赋上。arr[index]=num;

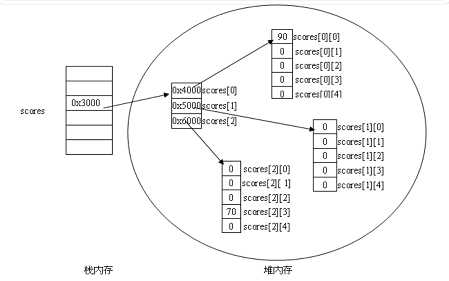


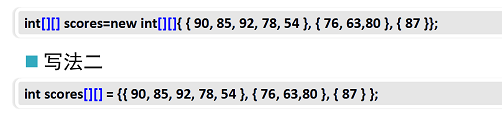
**二维数组**



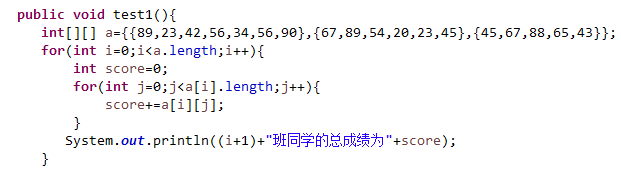


**第一个中括号必须要有值**





**表示共3行，[0][3]=78;**



**项目总结：**

**分别求出每一行的总数。a.length行数**

**a[i].length对应每一行的列数**

**注意是不是应该换行，注意用for语句的时候，是不是i对下面的值有效，可用。**

**是否要在此基础上+1.冒泡排序i不参加于其中。**

**二维码生成。**

**爬虫计划+瀑布流布局+延迟加载**

**大家抓紧时间，预习，做笔记。**

**excel的生成。**

# 第三章 迷你DVD管理系统

## 3.1 要求

☆ 输入的数字不符合要求直接退出程序

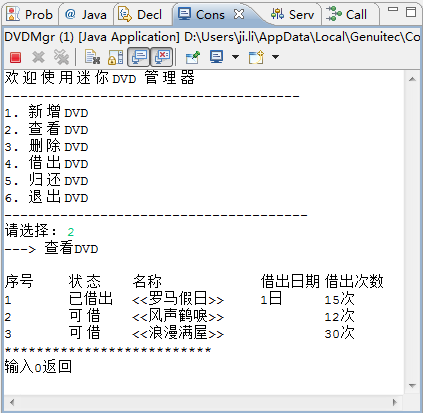
☆ 用户可以选择新增、查看、删除、借出、归还、退出

1.使用数组对象保存DVD信息

2.每张DVD的信息包：名称、是否可借的状态、借出的日期、借出的次数

3.最多存储6张DVD

4.显示该DVD信息（通过name[i]==null来退出循环）



### 3.1.1 新增DVD

要求：向各数组中存储相应的DVD信息

如果已经存储了6个DVD则提示用户货架已满

分析：定位新增位置

插入DVD信息，状态为“可借”

### 3.1.2 删除DVD

要求：按照输入的名称，删除指定DVD信息

删除借出状态的DVD，给出相应提示

删除不存在的DVD，给出相应提示

分析：查找要删除元素

通过把后面的元素依次前移一位，实现删除

最后一个不为空元素置空

思路：找到要删除DVD的位置，记录下标index，将该位置数据清空，再将其后数据元素往前移动

### 3.1.3 借出DVD

要求：按照输入的名称，借出指定DVD

判断输入的日期是否在1-31之间

若该DVD已借出，给出相应提示

若没有该DVD，给出相应提示

分析：查找要借出的DVD

判断是否存在

判断是否已借出

判断用户输入的借出日期

如果可借，则修改state值并且借出次数加1

### 3.1.4 归还DVD

要求：实现归还DVD，并计算租金

如果用户输入的日期大于31或者小于借出日期将提示用户重新输入

如果归还未借出的DVD、不存在的DVD或者归还DVD的时间不合理则分别给出相应提示

分析：查找要归还的DVD

判断是否存在

判断是否未借出

判断用户输入的归还日期

如果已借出，修改state值，并计算和输出租金

## 3.2 算法实现

**public** **class** Test {

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

String[] name=**new** String[6];//DVD名

String[] status=**new** String[6];//DVD借阅状态

**int**[] date=**new** **int**[6];//DVD日期

**int**[] count=**new** **int**[6];//DVD借阅次数

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

//初始化

//第一张DVD

name[0]="《罗马假日》";

status[0]="已借阅";

date[0]=1;

count[0]=15;

//第二张DVD

name[1]="《风声鹤唳》";

status[1]="可借阅";

count[1]=12;

//第三张DVD

name[2]="《浪漫满屋》";

status[2]="可借阅";

count[2]=30;

**int** q=1;

**do**{

System.***out***.println("欢迎使用DVD迷你管理器");

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

System.***out***.println("1.新增DVD");

System.***out***.println("2.查看DVD");

System.***out***.println("3.删除DVD");

System.***out***.println("4.借出DVD");

System.***out***.println("5.归还DVD");

System.***out***.println("6.退出DVD");

System.***out***.println("请选择：");

**int** choose=in.nextInt();

**switch**(choose){

//新增DVD

**case** 1:

System.***out***.println("--->新增DVD");

**boolean** flag=**false**;

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

**if**(name[i]==**null**){

System.***out***.println("请输入新增DVD的名称：");

String addName=in.next();

name[i]=addName;

status[i]="可借阅";

System.***out***.println("添加新增DVD成功");

flag=**true**;

**break**;

}

}

**if**(flag==**false**)

System.***out***.println("用户货架已满，无法添加");

**break**;

//查看DVD

**case** 2:

System.***out***.println("--->查看DVD");

System.***out***.println("序号 状态 名称 借出日期 借出次数");

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

**if**(name[i]!=**null**)//通过name[i]==null来退出循环

System.***out***.println((i+1)+" "+status[i]+" "+name[i]+" "+date[i]+" "

+ " "+count[i]);

}

**break**;

**case** 3:

System.***out***.println("--->删除DVD");

System.***out***.println("请输入删除DVD名称：");

String deleteName=in.next();

**int** index=-1;//要删除DVD处的位置标识

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

**if**(name[i]!=**null**){

**if**((name[i].equals(deleteName))&&(status[i].equals("已借阅"))){

System.***out***.println("该DVD处于借阅状态，无法删除");

**break**;

}

**else** **if**((name[i].equals(deleteName))&&(status[i].equals("可借阅")))

{

index=i;//用index记录要删除DVD的位置

name[i]=**null**;//将该位置的内容置空

}

}

}

**if**(index==-1){

System.***out***.println("该DVD不存在，无法删除");

}**else**{//当该DVD存在时，进行移位操作

**for**(**int** i=index+1;i<name.length;i++){

**if**(name[i]!=**null**){

name[i-1]=name[i];

status[i-1]=status[i];

date[i-1]=date[i];

count[i-1]=count[i];

}

}

System.***out***.println("删除成功");

}

**break**;

**case** 4:

System.***out***.println("--->借出DVD");

System.***out***.println("请输入需要借出的DVD名称");

String borName=in.next();

**boolean** flag1=**false**;

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

**if**(name[i]!=**null**){

**if**((name[i].equals(borName))&&status[i].equals("已借阅")){

System.***out***.println("该DVD处于借阅状态，无法借出");

**break**;

}

**if**((name[i].equals(borName))&&status[i].equals("可借阅")){

System.***out***.println("请输入借出日期：");

**int** borDate=in.nextInt();

**if**(borDate>=1&&borDate<=31){

flag1=**true**;

status[i]="已借阅";

date[i]=borDate;

count[i]+=1;

System.***out***.println("借出成功");

**break**;

}

}

}

}

**if**(flag1==**false**){

System.***out***.println("该DVD不存在，无法借出");

}

**break**;

**case** 5:

System.***out***.println("--->归还DVD");

System.***out***.println("请输入需要归还的DVD名称");

String retName=in.next();

**boolean** flag2=**false**;

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

**if**(name[i]!=**null**){

**if**((name[i].equals(retName))&&status[i].equals("可借阅")){

System.***out***.println("该DVD处于可借阅状态，无需归还");

**break**;

}

**if**((name[i].equals(retName))&&status[i].equals("已借阅")){

System.***out***.println("请输入归还日期：");

flag2=**true**;

**int** retDate=in.nextInt();

**if**(retDate>31&&retDate<=date[i]){

System.***out***.println("请重新输入归还日期");

retDate=in.nextInt();

}

**else**{

status[i]="可借阅";

**int** price=0;

**if**(retDate<date[i]){

System.***out***.println("归还成功，租金为："+price);

}**else**{

price=10\*(retDate-date[i]);

System.***out***.println("归还成功，租金为："+price);

}

}

**break**;

}

}

}

**if**(flag2==**false**){

System.***out***.println("该DVD不存在，无需归还");

}

**break**;

**default**:

System.***out***.println("--->退出DVD");

System.***out***.println("谢谢使用");

**break**;

}

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

System.***out***.println("输入0返回：");

q=in.nextInt();

}**while**(q==0);

}

}

# 第五章 封装

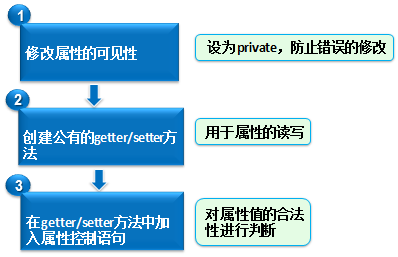
## 5.1封装

封装：将类的某些信息隐藏在类内部，不允许外部程序直接访问，而是通过该类提供的方法来实现对隐藏信息的操作和访问

封装的两大原则：把尽可能多的东西藏起来，对外提供便捷的接口

把所有的属性封装起来

封装的步骤：



封装的好处：

便于使用者正确使用系统，防止错误修改属性

有助于系统之间的松耦合，提高系统独立性

提高软件的可重用性

降低了构建大型系统的风险

## 5.2包

为什么需要包？

Windows树形文件系统：文档分门别类，易于查找和管理

使用目录解决文件同名冲突问题

如何放两个同名的类而不冲突：使用包

包的作用：允许类组成较小的单元（类似文件夹），易于找到和使用相应的文件

防止命名冲突区分名字相同的类

有助于实施访问权限控制

建包规则：作为Java源代码第一条语句

用package声明包，以分号结尾

包命名规则：包名由小写字母组成，不能以圆点开头或结尾

包名之前最好加上唯一的前缀，通常使用组织倒置的网络域名

包名后续部分依不同机构内部的规范不同而不同

导包关键字：import

使用包的注意事项：一个类同时引用了两个来自不同包的同名类，必须通过完成类名来区分

每个包都是独立的，顶层包不会包含子包的类

package和import的顺序是固定的，package必须位于第一行（忽略注释行），只允许有一个package语句，其次是import，接着是类的声明

## 5.3类的访问修饰符

### 5.3.1类的访问修饰符

public修饰符：公有访问级别

默认修饰符：包级私有访问级别

### 5.3.2类成员的访问修饰符



### 5.3.3static修饰符

成员变量：静态变量，可以直接通过类名访问

成员方法：静态方法，可以直接通过类名访问

代码块：静态代码块，当Java虚拟机加载类时，就会执行该代码块

static变量的作用：

（1）能被类的所有实例共享，可作为实例之间进行交流的共享数据

（2）如果类的所有实例都包含一个相同的常量属性，可把这个属性定义为静态常量类型，从而节省内存空间

Static方法：

a.静态方法：可直接通过类名访问

1.静态方法中不能使用this和super

2.不能直接访问所属类的实例变量和实例方法

3.直接访问类的静态变量和静态方法

b.实例方法：通过实例访问

可直接访问所属类的静态变量、静态方法、实例变量和实例方法

c.静态方法必须被实现

main()就是最常用的静态方法

static代码块：JVM加载类时，加载静态代码块

如果有多个静态块，按顺序加载

类的构造方法用于初始化类的实例，类的静态块用于初始化类，给类的静态变量赋值

## 第五章 总结

☆ JVM->class->static->构造方法->普通变量和方法等

☆ this:调用本类 super:调用父类

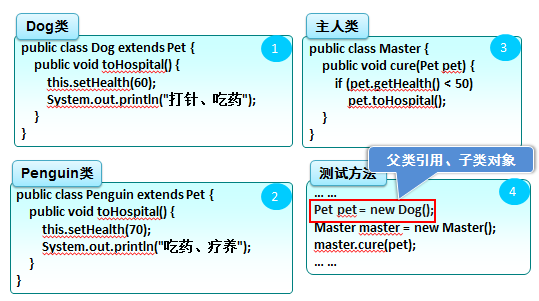
# 第七章 多态

## 7.1多态

生活中的多态：同一种事物，由于条件不同，产生的结果也不同

程序中的多态：同一个引用类型，使用不同的实例而执行不同操作

使用多态优化后的代码：



## 7.2 抽象类

1.抽象方法：抽象方法没有方法体

抽象方法必须在抽象类里

抽象方法必须在子类中被实现，除非子类是抽象类

2.向上转型：父类的引用指向子类对象。自动进行类型转换

注意：<父类型> <引用变量名> = new <子类型>();

此时通过父类引用变量调用的方法是子类覆盖或继承父类的方法，不是父类的方法

此时通过父类引用变量无法调用子类特有的方法

例如：Pet pet=new Dog();

3:向下转型：将一个指向子类对象的父类引用赋给一个子类的引用，即：父类类型转换为子类类型。需强制类型转换

注意：<子类型> <引用变量名> = (<子类型> )<父类型的引用变量>;

在向下转型的过程中，如果没有转换为真实子类类型，会出现类型转换异常

例如：Dog dog=(Dog)pet

4.使用父类作为方法的形参，是Java中实现和使用多态的主要方式

使用父类作为方法的返回值，也是Java中实现和使用多态的主要方式

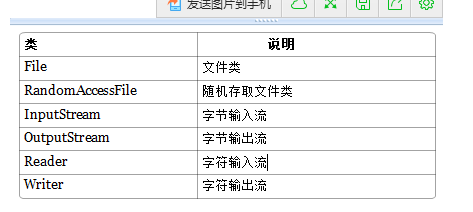
## 7.3 instanceof的使用

如何减少在向下转型的过程中，没有转换为真实子类类型的类型转换异常？

答：Java中提供了instanceof运算符来进行类型的判断

注意：使用instanceof时，对象的类型必须和instanceof后面的参数所指定的类在继承上有上下级关系

# 第七章 IO流





字符流的由来：

其实就是：字节流读取字节数据后，不直接操作，而是先查指定的编码表，获取对应的文字，再对这个文字进行操作。

**简单说：字节流+编码表。**

输入流和输出流是相对于设备而言的。

将外设中的读取到内存中：输入

将内存中的数据写入到外设中：输出

File file1=**new** File("C:\\a.txt");

[InputStream](http://baike.baidu.com/subview/1860884/1860884.htm)：继承自InputStream的流都是用于向程序中输入数据的，且[数据单位](http://baike.baidu.com/subview/5564170/5603604.htm)都是字节（8位）。

[OutputStream](http://baike.baidu.com/subview/1860888/1860888.htm)：继承自OutputStream的流都是程序用于向外输出数据的，且数据单位都是字节（8位）。

Reader：继承自Reader的流都是用于向程序中输入数据的，且数据单位都是[字符](http://baike.baidu.com/view/263416.htm)（16位）。

Writer：继承自Writer的流都是程序用于向外输出数据的，且数据单位都是字符（16位）。

[输入流](http://baike.baidu.com/subview/11752493/12120427.htm): 程序可以从中读取数据的流。

输出流: 程序能向其中写入数据的流。

按数据传输单位分:

[字节流](http://baike.baidu.com/view/1645587.htm): 以字节为单位传输数据的流

[字符](http://baike.baidu.com/view/263416.htm)流: 以字符为单位传输数据的流

按功能分:

[节点流](http://baike.baidu.com/view/2150237.htm): 用于直接操作目标设备的流

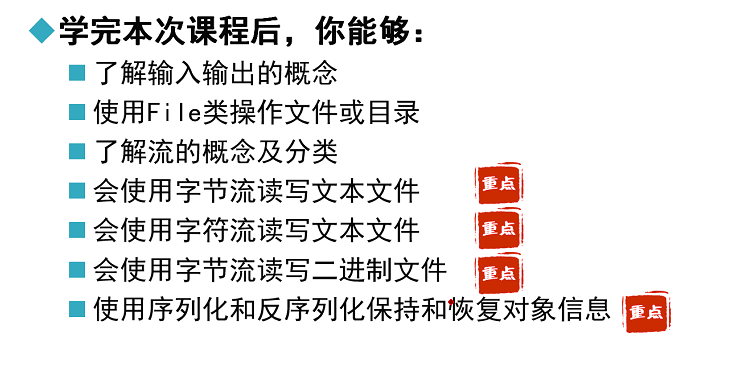
[过滤流](http://baike.baidu.com/view/2166528.htm): 是对一个已存在的流的链接和封装，通过对数据进行处理为程序提供功能强大、灵活的读写功能。

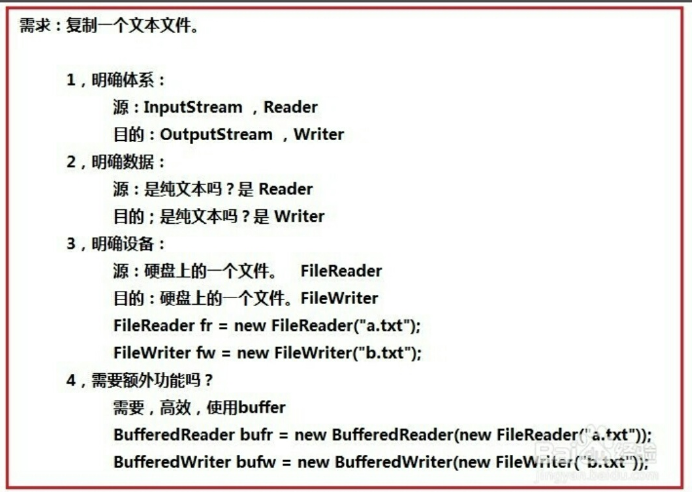
Java的核心库java.io提供了全面的IO接口。包括：文件读写、标准设备输出等。Java中IO是以流为基础进行输入输出的，所有数据被串行化写入输出流，或者从输入流读入。

流是一个很形象的概念，当程序需要读取数据的时候，就会开启一个通向[数据源](http://baike.baidu.com/view/286828.htm)的流，这个数据源可以是文件，内存，或是网络连接。类似的，当程序需要写入数据的时候，就会开启一个通向目的地的流。这时候你就可以想象数据好像在这其中“流”动一样。

[Java](http://baike.baidu.com/subview/29/12654100.htm)把这些不同来源和目标的数据都统一抽象为数据流。Java语言的输入输出功能是十分强大而灵活的，美中不足的是看上去输入输出的代码并不是很简洁，因为你往往需要包装许多不同的对象。

在Java类库中，IO部分的内容是很庞大的，因为它涉及的领域很广泛:[标准输入输出](http://baike.baidu.com/view/7151757.htm)，文件的操作，网络上的数据流，字符串流，对象流，zip文件流。



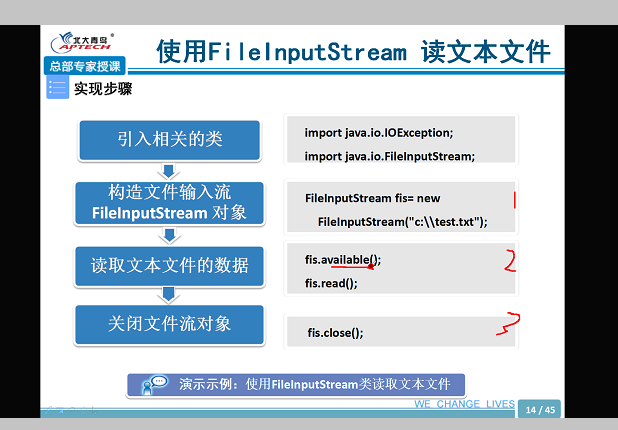
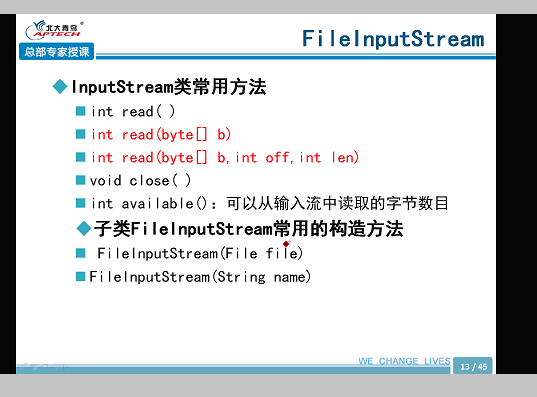












门定义为抽象类，锁定义为接口

防盗门继承门，实现锁的接口

所有方法默认都是： public abstract

接口不可以被实例化

实现类必须实现接口的所有方法

可以实现多接口 implements

接口中的变量都是静态变量。

public static final

接口有比抽象类更好的特性：

1.可以被多实现

2.设计和实现完全分离

3.更自然的使用多态

4.更容易搭建程序框架

5.更容易更换实现

接口中默认的是常量，和抽象方法

相同点

代表系统的抽象层

都不能被实例化

都能包含抽象方法

用于描述系统提供的服务，不必提供具体是实现

不同点

在抽象类中可以为部分方法提供默认实现，而接口中只能包含抽象方法

抽象类便于复用，接口便于代码维护

一个类只能继承一个直接的父类。

接口做系统与外界交互的窗口

接口提供服务

接口本身一旦制定，就不允许随意修改

抽象类可完成部分功能实现，还有部分功能可作为系统的扩展点

JAVA中的接口

属性都是全局静态常量

方法都是全局抽象方法

无构造方法

一个类可以实现多个接口，非抽象类实现接口时必须实现接口中的全部方法

抽象类便于

# 第八章

Java的异常处理是通过5个关键字来实现的；

try catch finally throw（手动抛出异常） throws(声明异常)

在try-catch块后加入finally块

是否发生异常都执行

catch语句的顺序：先子类后父类

try-catch-finally

catch,finally可选，两者至少出现一个

throw 抛出对象

throws 可以抛出多个异常类

## 8.1集合框架(Collection)

数组的局限性（长了，浪费，短啦不够用）

位于java.util包

API(应用程序接口)

API文档（应用程序接口参考文档）

接口,Collection

List(ArrayList LinkedLIst)

Set(HashSet TreeSet)

Map(HashMap TreeMap)

6个实现类

Iterator

Collection接口存储一组不唯一，无序的对象

List接口存储一组不唯一，有序的对象

Set接口存储一组唯一，无序的对象

Map接口存储一组键值对象，提供key到value的映像

ArrayList实现了长度可变的数组，在内存中分配连续的空间，有序的，

有下标。所以遍历元素和访问元素的效率比较高

LinkedList采用链表存储方式，插入、删除元素时的效率高

ArrayList常用方法

add(Object o)从0开始添加

add(int index,Object o)

get(int index)

contains(Object o)判断列表中是否存在指定元素

remove(Object o)删除列表中指定元素

remove(int index)

Collection接口常用通用方法还有：clear()、isEmpty()、iterator()、

toArray()

HashSet是Set接口常用的实现类

Set中存放对象的引用

采用equals()方法比较两个对象是否相等

如何遍历Set集合

方法1：通过迭代器Iterator实现遍历

获取Iterator:Collection接口的iterate()方法

it.hasNext() it.next()

Map

通过key对value进行操作

最常用的实现类是HashMap

Object put(Object key,Object val)以“键-值”对的方式进行存储

Object get(Object key)

Object remove(Object key) 删除由指定的键映射的“键-值对”

int size() 返回元素的个数

Set keySet() 返回键的集合

Collection values() 返回值的集合

boolean containsKey(Object key)

Collections.binarySearch() -（插入点）-1

.sort(list) 升序

.max()

.min()

实现类 String Date Calendar StringBuffer

Math Random

DataInputStream 类 与FileInputStream类结合使用读取二进制文件

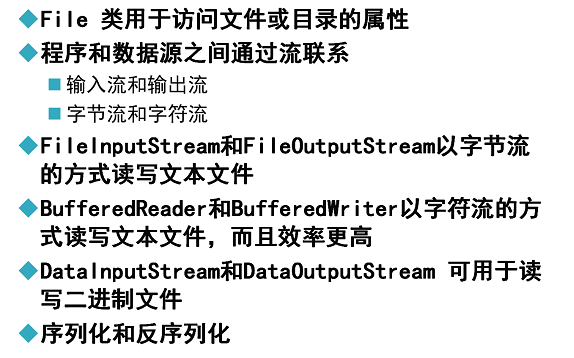
读取二进制文件

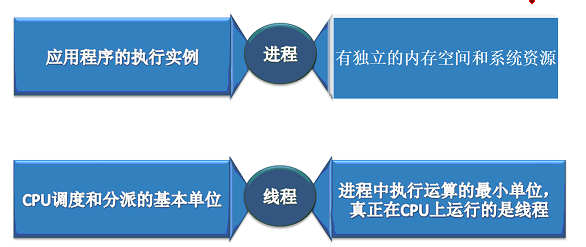
DataOutputStream类 与FileOutputStream类

序列化是将对象的状态写入到特点的流中的过程

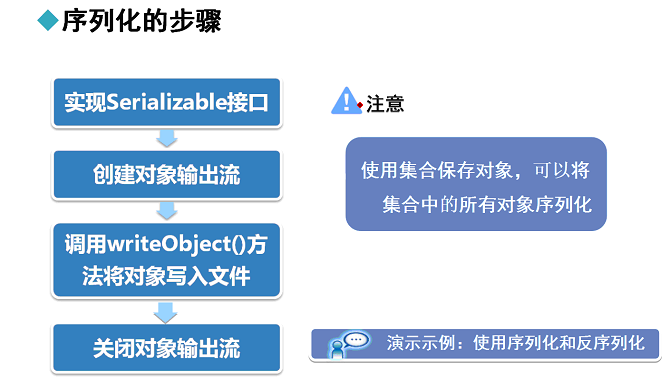
要序列化的对象必须实现Serializable接口

字节流，字符流，输入流，输出流





真正在CPU上运行的是线程



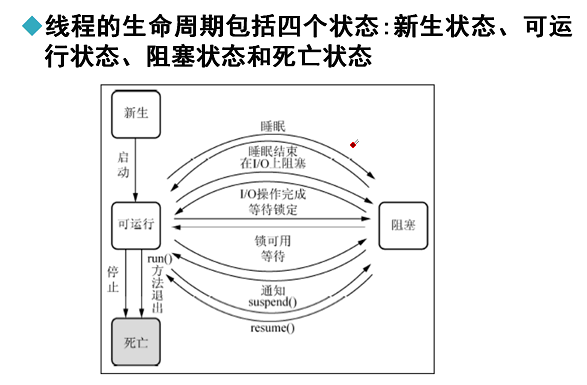
在JAVA中创建线程的两种方式

继承java.lang.Thread类

实现java.lang.Runnable接口

重写run方法。

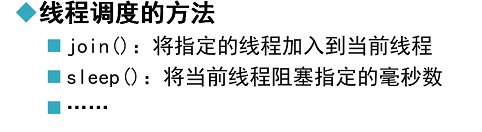




优先级：反映线程的重要或紧急程度。

线程的优先级用1-10表示，默认5

setPriority(int grade) 设置优先级 myThread.setPriority(10);



同步代码块 synchronized（this）（锁）

锁住本类

线程同步，线程调度。。

IO流

枚举指由一组常量组成的类型

[Modifier]enem enumName{

}

public enum Genders{Male,Female}性别枚举

public class Student{public Genders sex;}枚举类型的变量

Student stu=new　student();

stu.sex=Genders.male

包装类

集合不允许存放基本数据类型数据，存放数字时，要用包装类型

Object-Boolean,Character,Byte,Short,Integer,Long,Float,Double

除Character类外，其他包装类可将一个字符串作为参数构造它们的实例。

Boolean类构造String类型时，字符串内容为True不考虑大小写

Number包装类，String类型时，不能为null。否则NumberFormatExceptio

包装类，转换为基本类型

int intled=integerId.intValue();

toString();以字符串形式返回包装对象表示的基本类型数据（基本类型-》字符串）

String sex=Character.toString('男');

String id=Integer.toString(25）

string.indexof("在");

搜索第一个出现的位置。必须是字符

.lastindex("")最后出现的字符

subString()提取从字符串位置到指定位置的字符串

subString(int,int)(前面包括，后面不包括)

trim()//去除前后空格

//

StringBuffer s=new StringBuffer(string);

s.insert(i,'s');

String 是不可变对象

经常改变内容的字符串最好不要使用String

StringBuffer是可变的字符串

字符串经常改变的情况可使用StingBuffer,更高效

如何获得当前日期？

java.util.Date类：表示日期和时间

Date date=new Date();

String s="yyyy年MM月dd日 HH:mm:ss";

SimpleDateFormat s=new SimpleDateFormat(s);

.format 将日期转化成字符串

.parse 将字符串转化为日期

什么是文件

相关记录或放在一起的数据的集合

文件一般存储在哪里？

File file=new File()

操作文件的权限，等信息

exists() 目录是否存在

isFile() 判断是否是文件

isDirectory() 是否是目录

getPath() 相对路径

getAbsolutePath() 绝对路径

getName() 获取文件或目录的名称

delete() 删除此对象指定的文件或目录

createNewFile() 创建名称的空文件，不创建文件夹

length() 返回文件的长度，单位为字节，如果不存在，则返回空

按流向区分 1.输出流 2.输入流

InputStream OutputStream Writer Reader

编码格式：国际通用utf-8.

syso(System.getProperty("file.enconding"));

Reader fr=new FileReader("D:\\");

缓冲流。提高字符流读取文本文件的效率

-wait()：使一个线程处于等待（阻塞）状态，并且释放所持有的对象的锁；

-sleep()：使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要处理InterruptedException异常；

-notify()：唤醒一个处于等待状态的线程，当然在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且与优先级无关；

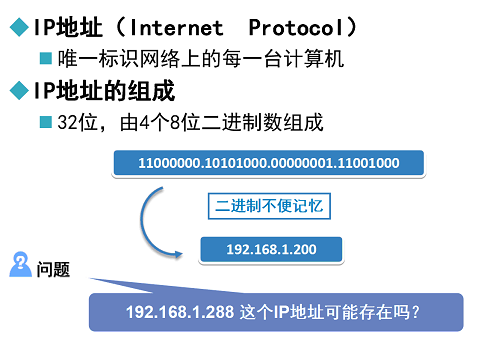
-notityAll()：唤醒所有处于等待状态的线程，该方法并不是将对象的锁给所有线程，而是让它们竞争，只有获得锁的线程才能进入就绪状态；

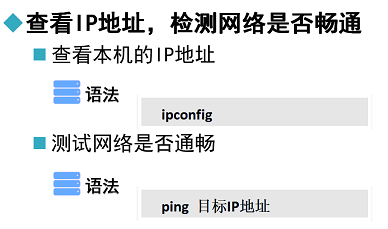
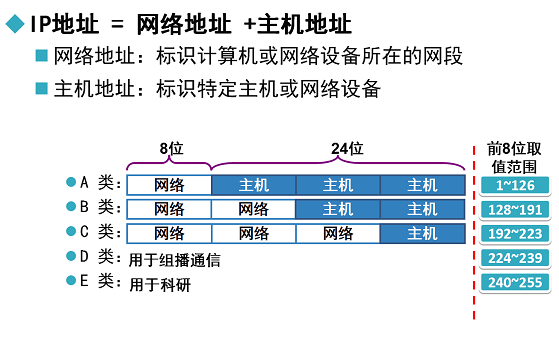
线程的同步是保证多线程安全访问竞争资源的一种手段

在Java多线程环境中，为保证所有线程的执行能按照一定的规则执行，JVM实现了一个线程调度器，它定义了线程调度的策略，对于CPU运算的分配都进行了规定，按照这些特定的机制为多个线程分配CPU的使用权。这小节关注线程如何进行调度，了解了java线程调度模式有助于后面并发框架的深入探讨。

一般线程调度模式分为两种——抢占式调度和协同式调度。抢占式调度指的是每条线程执行的时间、线程的切换都由系统控制，系统控制指的是在系统某种运行机制下，可能每条线程都分同样的执行时间片，也可能是某些线程执行的时间片较长，甚至某些线程得不到执行的时间片。在这种机制下，一个线程的堵塞不会导致整个进程堵塞。协同式调度指某一线程执行完后主动通知系统切换到另一线程上执行，这种模式就像接力赛一样，一个人跑完自己的路程就把接力棒交接给下一个人，下个人继续往下跑。线程的执行时间由线程本身控制，线程切换可以预知，不存在多线程同步问题，但它有一个致命弱点：如果一个线程编写有问题，运行到一半就一直堵塞，那么可能导致整个系统崩溃。

## 8.2网络通信

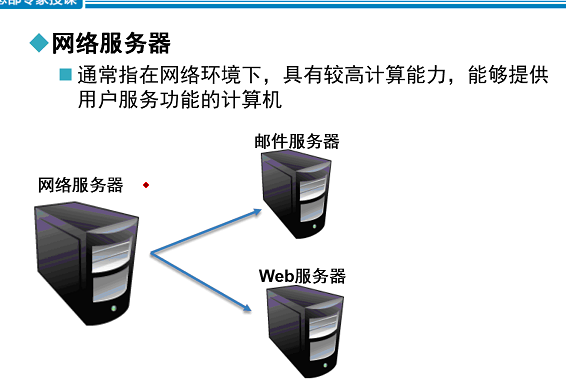




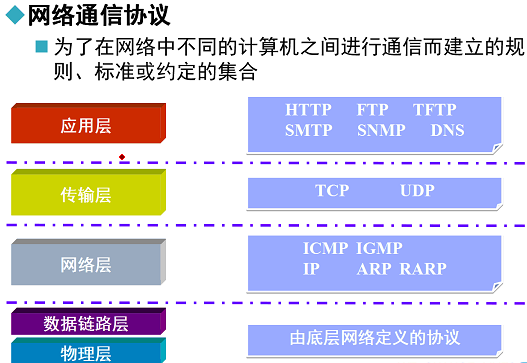
DNS域名解析

访问网站时，为什么输入网址而不是IP地址。

访问域名，会转换为IP地址。



请求和响应



应用层：HTTP,FTP,TFTP,SMTP,SNMP,DNS

传输层：TCP（文件，慢）,UDP(快，视频和广播)

网络层：ICMP IGMP IP ARP RARP

数据链路层，物理层：由底层网络定义的协议.

、

# 第九章 XML（Dom sex Dom4j）

理解XML概念及优势

会编写格式良好的XML文档

了解XML中特殊字符的处理方式

了解解析器及命名空间概念

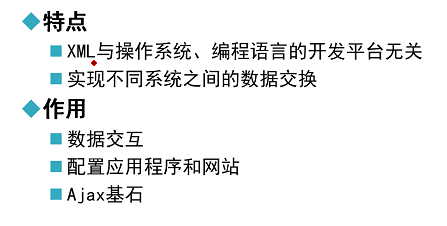
了解DOM树节点构造

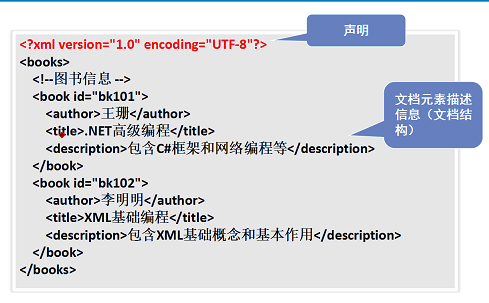
会使用DMO操作XML数据

会使用DOM4J操作XML数据

XML(Extensible Markup Language)可扩展标记语言

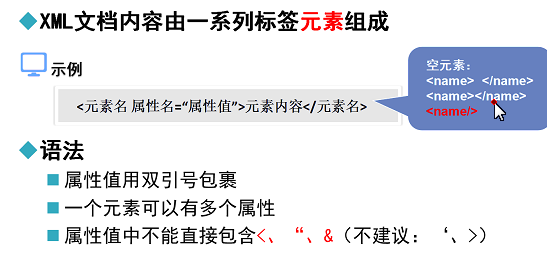
特点：

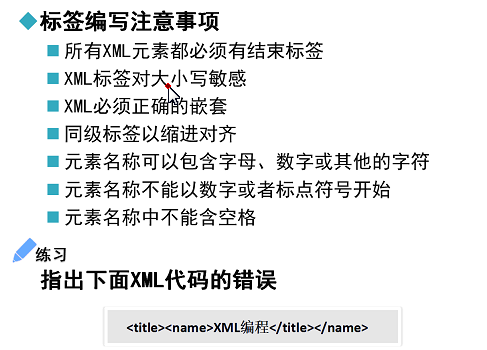




<?xml version=”1.0” encoding=”utf-8”>

<元素明 属性名=”属性值”>元素内</元素名>

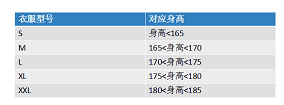


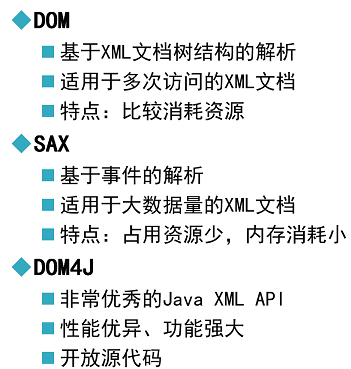


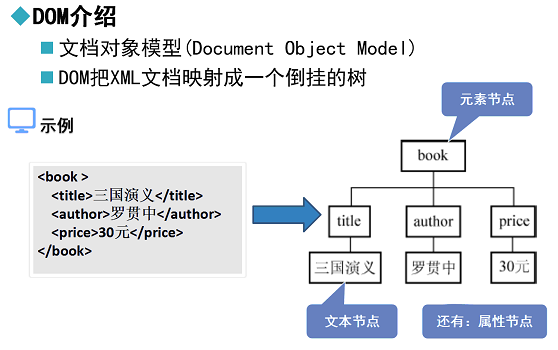
元素名称不能包含空格，不能以数组或者标吊符号开始。可以包含字母、数字或其他的字符。

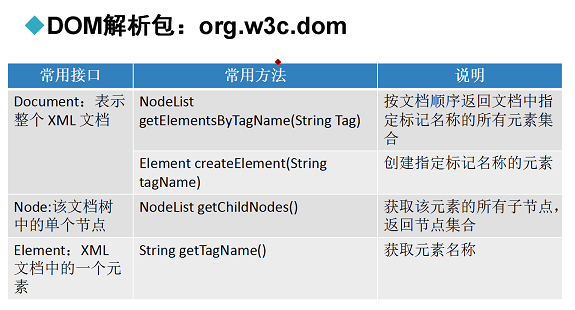














Xml解析空间



