# 一、常用的dos命令

## 1、操作文件夹

盘符： 进入指定的盘符

dir 列出当前目录下的所有文件及文件夹

md 文件夹名 创建目录（文件夹）

rd 文件夹名 删除目录（文件夹） 注意：只能删除空的文件夹，不能删除非空文件夹，也不能删除文件

cd 文件夹名 进入指定目录

cd.. 返回上层目录

cd \ 返回根目录

## 2、操作文件

echo 数据>文件夹名 创建一个文件并且指定写入的数据

type 文件名 显示文件内容

del 文件名 删除文件，注意只能删除文件，不能删除文件夹

exit 退出dos命令行

cls 清空当前命令行

\* 通配符

Tab 键 命令补全

方向键上下 找回之前写过的命令（当前命令行有效）

## 3、set命令操作环境变量

在实际开发中，有时jdk的版本只需要一次或几次，这种需求，我们一般都是通过set命令来配置临时变量。

set 查看所有环境变量的信息。

set 环境变量名字 查看指定环境变量的信息。

set 环境变量名字 = 路径信息 设置指定环境变量的信息。

set 环境变量名字 = 清空指定环境变量的信息。

Set 环境变量名字 = 路径信息;%环境变量名字% 在原有的基础上添加新的路径信息。

注意：以后但凡看到set设置的环境变量信息，都是临时环境变量，只在当前dos窗口有效。窗口关闭，配置消失。

# 二、java简介

## 1、java的起源

* 1993年，初衷是设计一个机顶盒
* 1995年5月23日，java诞生了 雏形
* 1996年发布jdk1.0版本
* 1997年发布jdk1.1版本
* 1999年java被分成 J2SE、J2EE、J2ME
* 2000年发布jdk1..4版本
* 2004年9月30日，J2SE1.5版本发布，成为java语言发展史上的又一里程碑。

为了表示该版本的重要性，J2SE1.5更名为 java SE 5.0

* 2005年，java版本正式更名为JAVASE、JAVAEE、JAVAME
* 2006年12月，SUN公司发布JRE6.0
* 2009年4月20日，Oracle已74美元收购sun公司，获得Java版权
* 2010年9月，jdk7.0发布，增加了简单的闭包功能
* 2011年7月，甲骨文公司发布java7的正式版本
* 2014年3月，jdk8.0发布
* 2017年9月，jdk9.0发布
* 2018年3月，jdk10.0发布
* 2018年9月，jdk11.0发布

注意：甲骨文公司是Oracle公司在中国的叫法

## 2、java新特性

* JDK1.4 正则表达式，异常链，NIO，日志类，XML解析器，XLST转换器
* JDK1.5 自动装箱、泛型、动态注解、枚举、可变长参数、遍历循环
* JDK1.6 提供动态语言支持、提供编译API和卫星HTTP服务器API，改进JVM的锁，同步垃圾回收，类加载
* JDK1.7 提供GI收集器、加强对非Java语言的调用支持（JSR-292,升级类加载架构
* JDK1.8 Lambda 表达式、方法引用、默认方法、新工具、Stream API、Date Time API 、Optional 类、Nashorn, JavaScript 引擎

## 3、java语言需要考证吗？

不需要，英语四级证书，高学历（985、211或研究生）

# 三、java语言重要特性

## 1、什么是跨平台？

跨平台是只既不依赖操作系统，也不信赖硬件环境，一个操作系统中开发的应用，放到另一个操作系统中依然能够运行。

## 2、java的跨平台（write once run anywhere）

java的跨平台是通过java虚拟机（JVM）来实现的，一次编译，处处运行。

## 3、java虚拟机（JVM）

### （1）作用

解释class文件并通知系统要执行哪些操作。

### （2）java虚拟机跨平台吗？

Java虚拟机不能跨平台，java写出的文件才是跨平台的。

第一是因为所需要的相同资源，在不同操作系统上所需要的指令是不同的。

第二是因为不同操作系统上的可执行文件是不同的。

第三是因为在不同的操作系统（平台）有不同的Java虚拟机（jvm）

### （3）java跨平台的原理

编写java源程序（.java文件）

——》 通过编译程序，编译java源文件，翻译成 java字节码文件（.class文件）

——》 java虚拟机执行编译后的Java字节码文件，形成能够被操作系统（平台）执行的指令。

——》 各个平台通过平台上相对应的JVM去操作这些指令。

### 4、java的三大平台（JAVASE、JAVAEE、JAVAME）

* JAVASE 支持桌面、服务器、嵌入式环境额实时环境的java应用程序开发（QQ、扫雷等）
* JAVAEE 支持企业级的服务器端java应用程序开发，它是在JAVASE基础上构建的，他提供WEB服务，组件模型，管理和通信API，可以用来实现企业级面向服务体系结构和WEB2.0 应用程序
* JAVAME支持移动设备和嵌入式设备上运行的应用程序的开发（手机、iPad等）

# 四、java环境搭建

## 1、JDK简介

JDK是java的开发工具集。 JDK = jre + java开发工具。

JRE是java运行环境。 JRE = JAVA虚拟机（jvm）+核心类库（辅助java虚拟机运行的文件）



## 2、JDK目录介绍

* bin 目录：存放java的编译器、解释器等开发工具（可执行文件）。
  + javac.exe 启动java编译器，对指定的java源文件进行编译，生成.class文件。
  + java.exe 启动java虚拟机解释并执行指定的class文件。
* db目录 ：JDK7附带的一个轻量级的数据库，名字叫做Derby。
* include目录：存放的调用系统资源的接口文件。
* jre 目录：存放java运行环境文件。
* lib 目录：存放java的核心类库文件。
* src.zip文件：JDK提供的java的源代码。

## 3、JRE简介

JRE是java运行环境，包括JAVA虚拟机和java程序运行所需的核心类库等，如果想运行一个开发好的Java程序，计算机中只需要安装JRE即可。

## 4、配置JAVA环境变量

（1）下载并安装jdk，安装路径中不能出现中文和空格及特殊字符

（2）配置环境变量

* JAVA\_HOME E:\JavaEnvironment\jdk1.8
* CLASSPATH .;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;
* path ;%JAVA\_HOME%\bin; %JAVA\_HOME%\jre\bin;

（3）测试

* 在dos窗口分别测试 java、javac、java –version、javac –version四个命令，都能识别即jdk安装成功。

# 五、java入门程序

## 1、写一个 HelloWord程序，使用javac进行编译，使用java进行运行。

## 2、JAVA\_HOME环境变量

（1）为什么要配置JAVA\_HOME？

有时候可能要修改jdk的目录，但是一直修改path可能会不小心修改其他路径。

（2）怎么配置JAVA\_HOME?

创建一个JAVA\_HOME变量，值为jdk的安装目录。

## 3、Path环境变量

（1）为什么要配置path环境变量？

因为在jdk的bin目录下有许多我们在开发中要使用的工具，比如java.exe、javac.exe、jar.exe等。当我们想在电脑的任意位置都可以使用这些Java工具时，那么我们就需要将这些工具所在的路径配置到系统的环境变量中，当我们需要使用这些工具时，系统可以帮助我们找到这些命令。

（2）怎么配置path环境变量？

将jdk中bin目录的路径，加入到path目录中，不要忘记加分号。

## 4、classpath环境变量

（1）classpath的作用?

指定类搜索的路径（设置了class文件的路径信息）。一旦配置了classpath的路径信息，JVM和java编译器都会根据classpath指定的路径去查找class文件。

（2）如何设置当前目录（.）?

设置classpath的时候加上分号“;”,就会在当前目录下查找。

（3）set classpath = .; 设置路径与set classpath = ;. 有什么区别？

“.”在前边的时候，java虚拟机会从当前路径去寻找class文件，然后再到指定路径去找。

“.”在后边的时候，java虚拟机则会先从指定好的路径去找，然后再在当前路径寻找。

（4）不建议大家将classpath配置成永久性的配置信息。

Jdk5.0 Sun公司也不建议大家强制配置成永久性的环境信息。Jdk5.0之前，jvm虚拟机比较笨，需要强制性的要求配置classpath环境变量。

# 六、自定义标识符

## 1、什么是自定义标识符？

在java程序中，有些名字是可以自定义的，这些可以自定义的名字我们就叫做自定义的标识符。

## 2、自定义标识符的规则（必须遵守的）

1. 标识符的组成元素（字下美数）
   * + - 1. 字母 a-zA-Z
         2. 下划线 \_
         3. 美元符号 $
         4. 数字 0-9
2. 不能以数字开头
3. 标识符是严格区分大小写的
4. 标识符没有长度限制
5. 标识符的命名要有意义（要做到见名知意）
6. 关键字不能用作自定义的标识符

## 3、自定义标识符的规范（不会影响程序运行，是为了美化让人看着舒服）

1）类名和接口名：单词的首字母大写，其他字母小写

2）变量名和方法名：首单词全部小写，其他单词首字母大写

3）包名：全部单词小写

4）常量（static）：全部单词大写，单词之间用下划线分割。

# 七、关键字

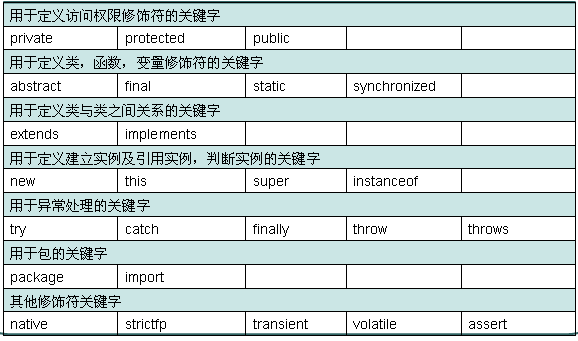
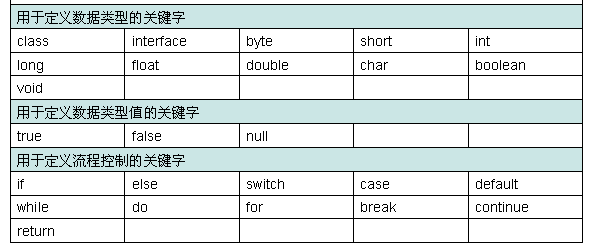
## 1、什么是关键字？

关键字就是在java程序中具备特殊含义的标识符。

关键字一边用来表示一个程序的结构或者数据类型。

## 2、常见的关键字

goto 是java中的保留关键字。就是说现在java还没有使用goto，但是以后会不会使用未定。



# 八、注释

## 1、什么是注释？

注释就是用文字对程序进行说明

注释是写给程序员看的，java编译器会忽视注释的内容，不会对注释内容进行编译。

## 2、注释的使用

通常在程序的开头加入作者，时间，版本，要实现的功能等内容注释，方便后期的维护和程序员的交流。

## 3、注释的类别

* 单行注释 // 注释内容
* 多行注释 /\* 注释内容 \*/
* 文档注释 /\*\* 注释内容 \*/

## 4、文档注释与多行注释的区别

* 多行注释的内容不能用于生成一个开发者文档。
* 文档注释注释的内容可以被JDK提供的Javadoc工具所解析，生成一个以网页文件形式体现的该程序的开发者说明文档。

## 5、使用Javadoc生成开发者文档。

* 生成的开发者文档是 HTML格式。

### 1）Java工具的使用语法

* + 语法：Javadoc options packagenames sourcesfiles
  + options 操作命令
    - -d（directory） 输出文件的目标目录
    - -version 包含@version段（版本号）
    - -author 包含@author（作者）

### 2）使用javadoc工具需要注意的细节

* + 该类必须是 public修饰
  + 文档注释注释的内容一般都是位于类或方法的上面
  + 单行注释可以嵌套使用
  + 多行注释、文档注释不能嵌套使用。

## 6、 使用注释的规范

* 单行注释 一般是在代码的右面
* 多行注释、文档注释 一般是在类和方法的上面

## 7、注释的作用

1）调试程序

2）使用文字对注释进行说明

# 九、常量

## 1、什么是常量？

常量就是程序在运行过程中其值不能发生改变的量。

## 2、常量的类别

整数常量 ：12、22、23

小数（浮点数）常量 ：3.14、4.34

布尔常量 ：true、false

字符常量 ：使用单引号引起来的单个字符 ‘a’、 ‘b’、 ‘c’等

字符串常量 ：使用双引号引起来的的内容，称为字符串常量“hello”、“no”、“你好”等

null常量 ：只有一个值 null

# 十、转义字符

|  |  |  |
| --- | --- | --- |
| 转义字符 | 名称 | Unicode |
| \b | Backspace （退格键） | \u0008 |
| \t | Tab （Tab键盘） | \u0009 |
| \n | Linefeed （换行） | \u000A |
| \r | Carriage Return（回车） | \u000D |
| \\ | Backslash （反斜杠） | \u005C |
| \' | Single Quote （单引号） | \u0027 |
| \" | Double Quote （双引号） | \u0022 |

\r 表示接受键盘输入，相当于按下回车。

\n 表示换行。

\t 制表符，相当于Table键

\b 退格键，相当于Back Space

\’ 单引号

\’’ 双引号

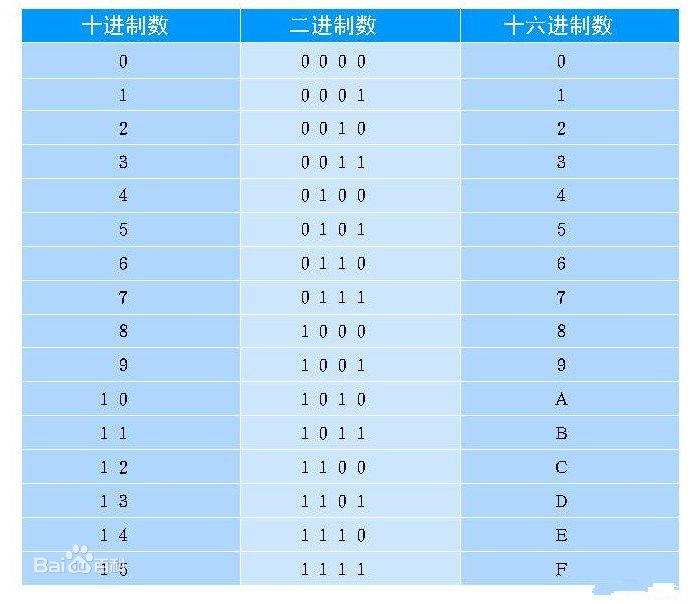
\\ 表示一个斜跨

上述问问题解决：System.*out*.println("teacher said\"java is fun\"");

注意：换行符就是另起一行，回车符就是回到一行的开头，所以我们平时编写文件的回车符应该确切来说叫做回车换行符

# 十一、进制

进制是一种记数形式，可以用有限的数字符号代表所有数值



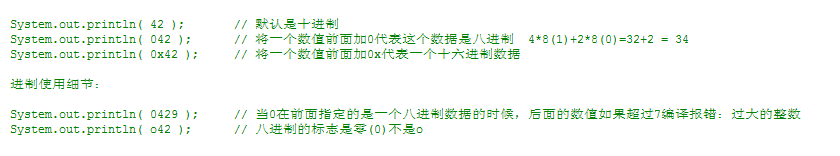
## 6.1、整数的表现形式

二进制：由0和1这两个数字组成。

八进制：由0到7这八个数字组成，为了区别其他数字，开头都是以0开始。

十进制：由0到9这十个数字组成，不能以0开头。

十六进制：由0-9和A-F组成。为了区分其他数字的区别，开头都是以0x开头的。



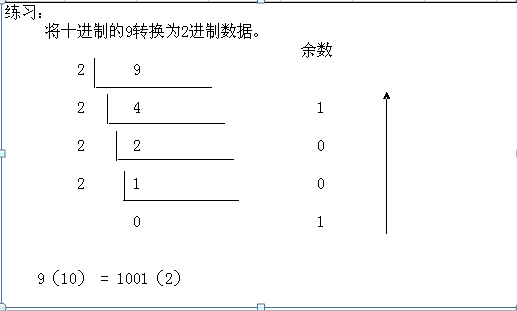
## 6.2进制之间的转换

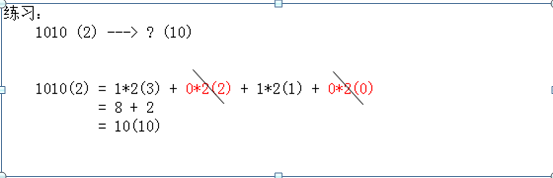
（1）十进制与二进制之间的转换

1）十 转 二 原理：十进制数除以2，直到商为0终止，反向取余，所得结果就是二进制。

例如：9（十进制） ————》 1001（二进制）

2）二 转 十 原理：使用二进制的每一位（从右往左）乘以2的n次方（n从0开始），每次递增1，然后把各部分的值相加即可。





## 6.3二进制、八进制、十进制、十六进制之间的转换。

# 十二、变量

## 7.1变量的概述

* 用于存储可变数据的容器

## 7.2、变量的特点

* 必须有类型
* 必须有大小
* 必须有名字

## 7.3、变量的声明方式

* 数据类型 变量名 = 变量值 ；

# 十三、变量的数据类型

## 1、数据类型

java中存在两种数据类型

* + - 基本数据类型
    - 引用数据类型

## 2、基本数据类型（8个）

* 整数类型（4个）byte、short、int（默认）、long
* 浮点类型（2个）double（默认）、float
* 字符类型（1个）char，可以转换成int类型
* 布尔类型（1个）boolean，只有true、false（默认）两个值

注意：

* 整数类型
  + 默认是int类型，值为0。
  + long类型的数据，需要在数值后加L（不区分大小写），但是推荐大写L。
* 浮点类型
  + 默认是double类型，值为0.0。
  + float类型的数据，需要在数值后加F（不区分大小写）。
* 布尔类型默认值是false。
  + 如果布尔声明一个基本类型的变量时，那么该变量占4个字节。
  + 如果布尔声明一个数组类型的时候，那么该变量占1个字节。
* char字节类型
  + 原理：将字符映射为码表中对应的十进制数据加以存储。

数据类型使用规则：

* 在满足规则的情况下，能小则小
* 现实开发时，一般整数使用int类型，小数使用double类型。

## 3、变量的声明

### （1）变量先声明再赋值

语法： 数据类型 变量名1，变量名2… ；

案例：

int a,b,c；

### （2）变量的初始化

语法：

初始化方式一： 数据类型 变量名1 = 值1；

初始化方式二： 数据类型 变量名1，变量名2 = 值2；

案例：

double d = 3.14；

double f；f = 3.14；

## 4、类型转换

### （1）类型转换的分类

* 小数据类型 ————》 大数据类型，叫做自动类型转换（也叫隐式类型转换）。
* 大数据类型 ————》 小数据类型，叫做强制类型转换（也叫显示类型转换）。

### （2）自动类型转换

1）自动类型转换需要满足的条件

* 两种互相转换的数据类型彼此兼容。
* 目标类型取值范围必须大于源类型。

### （3）强制类型转换

1）强制类型转换需要注意

* 损失精度

## 5、数据类型转换的原理

* 小的可以自动转换成大的，大的强制转换成小的会损失精度。

## 6、数据类型转换需要注意的细节

* 凡是byte、short、char数据类型的数据在进行运算时，都会自动转换成int类型的数据再运算。（先转换成int类型，再进行运算）
* 两个不同数据类型的数据在进行运算的时候，结果取决于大的数据类型。
* 一个整数在没有加上任何标识的时候，默认是int类型
* 10 是一个常量，编译器在进行编译的时候，就能够确定常量的值，byte b=10，在编译的时候，java编译器就检查到10在byte所能表示的范围内，所以就可以赋值。
* java编译器在进行编译的时候，不能确定变量所存储的值，变量存储的值是在运行的时候才在内存中分配空间的。

# 十四、原码、反码、补码

* 在计算机中，如果一个二进制的最高位是1，那么该数据肯定是个负数。
* 如果一个数据的二进制位最高位是0，那么该数据肯定是个整数。
* 负数在计算机中存储的是补码。
* 正数的补码、反码都是其本身。

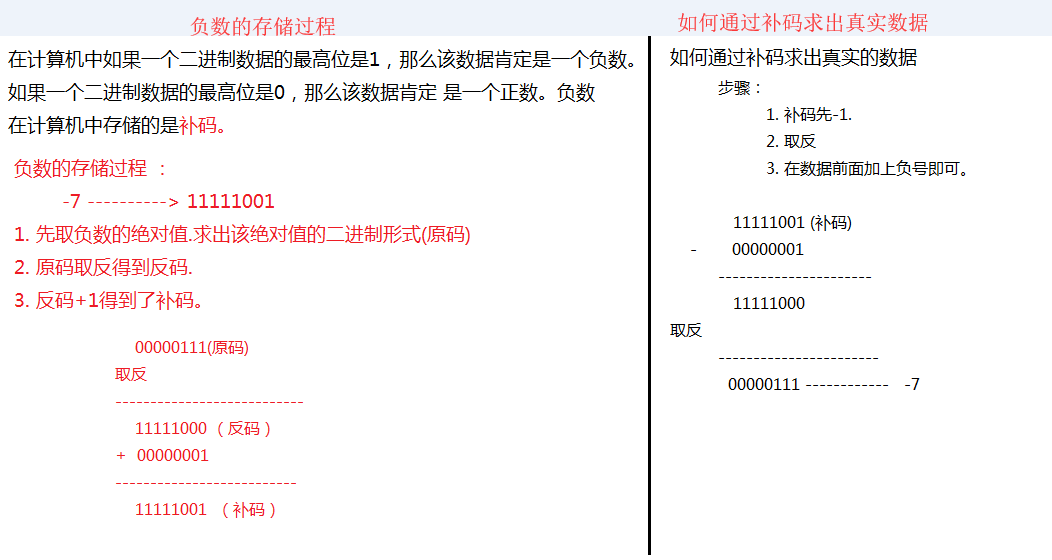
## 1、负数的存储过程。

### （1）负数在计算机中是通过补码的形式进行存储的。

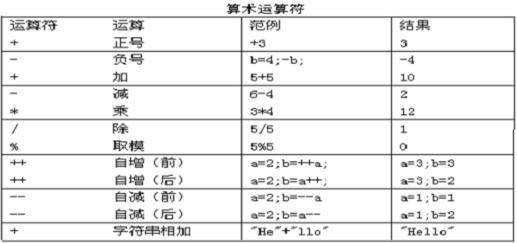
* 第一步：获取该负数的绝对值。求出该绝对值的二进制形式（原码）。
* 第二步：原码取反得到反码。取反 就是将所有的1和0进行交换。
* 第三步：反码 +1得到补码。

### （2）通过补码得到真实数据

* 第一步：将二进制的补码进行 -1操作
* 第二步：取反得到原码
* 第三步：将二进制原码还原成真实的十进制数据，在前边加上负号。



# 十五、算数运算符



## 1、正数、加法、连接符（+）

### （1）连接符的作用

* 让任何数据都可以和字符串进行拼接
* 如果 + 号用于字符串时，那么 + 号就是一个连接符，并不是做加法

### （2）连接符注意事项：

* 任何类型的数据和字符串使用连接符进行连接，结果都是字符串类型的数据。

## 减法、乘法、除法（取商）

* 1. 进行减法、乘法运算。
  2. 取商只取整数，舍去余数。

## 5、取余（%）

* 做取余（取模）运算，结果是余数。
* 在java中做取模运算时，结果的正负号是取决于被除数的。

## 自增（+ +）

* 自增就是数值本身 +1
* 前自增：先进行 +1操作，然后被使用
* 后自增：先使用，然后再 +1，新数据用以下次使用
* 后自增在jvm虚拟机的运行原理：
  + 因为后自增会使用到没有+1之前的值，jvm会先声明一个变量用来保存没有进行 +1运算之前的值。

int i = 0; i++;

原理：

int temp = i； //temp = 0;

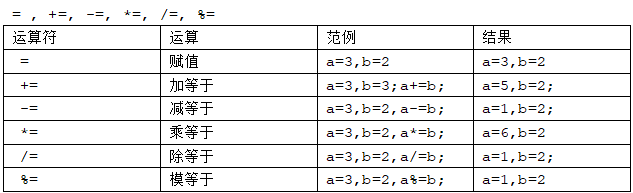
自增 i = i + 1;

返回 return temp; // temp用作了表达式的结果。

## 7、自减（- -）

* 自增就是数值本身 -1
* 前自增：先进行 -1操作，然后被使用
* 后自增：先使用，然后再 -1，新数据用以下次使用。

# 十六、赋值运算符

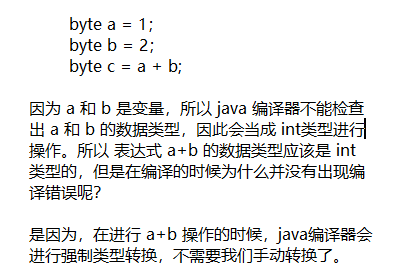


* a += b；

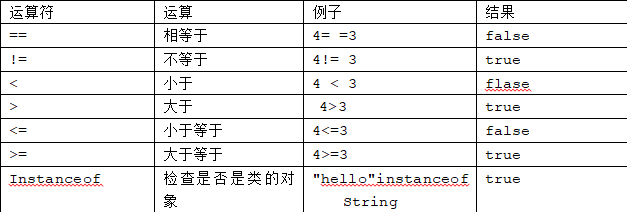
相当于

a = a + b;

* 其他运算符作用相同。
* 注意：



# 十七、比较运算符



1、 判断是否相等（==）

（1）== 用于判断两个基本数据类型的变量的时候，比较的是两个变量所存储的值是否一致。

（2）== 用于判断两个引用数据类型的变量的时候，比较的是两个 引用类型变量所记录的内存地址是否一致。

2、不等于（!=）

1、比较运算符使用细节

* + 比较运算符两边的数据类型必须一致
  + 比较运算符的结果都是返回一个布尔值（true或false）

# 十八、逻辑运算符

* 逻辑运算符的作用是用于连接布尔表达式的。



1. 双与（&&） 和单与（&）的区别
2. 异或（^）

* 只要表达式两边的结果相同，结果就是false。（一真一假结果为true，两真或两假结果为false。）

# 十九、位运算符

## 1、与（&）

* 可以把1看成true，把0看成false。（两个true为1，只要存在一个false就是0）

## 2、或（|）

* 可以把1看成true，把0看成false。（两个false为0，只要存在一个true就是1）

## 3、异或（^）

* 可以把1看成true，把0看成false。（两个false或两个true为0，两个不同为1）

规律：如果操作数 a 连续异或同一个操作数两次，那么结果还是操作数 a。

应用：可以对数据进行加密

## 4、取反（~）

* 按位取反运算符是 将内存中的 补码 按位取反



# 二十、位移运算符

## 1、左移（<<）

* 规律：一个操作数进行左移运算的时候，结果就是等于操作数乘以2的n次方。n是左移的位数
* 举例：
  + - 3<<1 = 3\*2(1) = 6;
    - 3<<2 = 3\*2(2) = 12;

## 2、右移（>>）

* 规律：
  + 一个操作数进行右移运算的时候，如果该操作数是一个正数，那么左边空缺位用0来补；如果该操作数是一个负数，那么用1来补。
  + 一个操作数进行右移运算的时候，结果就是等于操作数除以2的n次方。n是右移的位数。
* 举例：
  + - 3>>1 = 3/2(1) = 1;
    - 3>>2 = 3/2(2) = 0;

## 3、无符号右移（>>>）

* 无符号右移（>>>） 与 右移（>>）的区别
  + 进行右移运算的时候，如果操作数是一个正数，那么左边的空缺位使用0补；如果操作数是一个负数，那么左边空缺位使用1补
  + 进行无符号右移运算的时候，不管操作数是正数还是负数，都统一使用0补。

# 二十一、三元运算符（三目运算符）

* 格式:

表达式 ？ 值1：值2;

* 使用细节 ：使用三维运算符时
  + 一定要使用该表达式返回的结果； System.out.println(age>18 ? “成年” : “未成年”);
  + 或者是定义一个变量接受该表达式返回的结果。 String result = age>18 ? “成年” : “未成年”;

# 二十二、if 控制流程语句

## 1、语句

* 使用分号分隔的代码就是一个语句。

## 2、顺序语句

* 按照代码顺序从上往下执行所有的代码，就是顺序语句。

## 3、if判断语句

### （1）格式

* + if(判断条件1){

符合判断条件1执行的代码;

}else if(判断条件2){

符合判断条件2执行的代码;

}else{

不符合判断条件1和2执行的代码;

}

### （2）if语句要注意的细节

* + 如果符合条件后只有一句代码需要执行，那么可以省略大括号。但是建议不要省略，因为结果不清晰。
  + if语句的判断条件后不能添加分号，否则会影响执行效果。

### （3）if-else 与三元运算符非常像，怎么选择？

* 三元运算符格式： 表达式？值1：值2；
* 三元运算符的优点：结构简洁
* 三元运算符的缺点：符合条件必须要返回一个结果，不能执行。

# 二十三、switch选择判断语句

## 1、格式

switch（选择）{

case 值1：

符合你选择的执行代码；

break；

case 值2：

符合你选择的执行代码；

break；

......

default：

你的选择都不符合以上选项时执行的代码；

break；

}

## 2、switch语句注意事项

* switch使用的变量只能是byte、short、int、char、String数据类型，String数据类型是从jdk7.0的时候开始的。
* case后边跟的，必须是一个常量
* switch的停止条件：
  + switch语句一旦匹配上了其中一个case，那么就会执行对应的case中的语句，执行完毕之后如果没有遇到break关键字或者是结束switch语句的大括号，那么switch语句不会再判断，按照代码的顺序从上往下执行所有的代码，直到遇到break或结束switch语句的大括号为止。
  + 遇到break或结束大括号。
* 在switch语句中，无论代码中顺序如何，永远是先判断case语句，没有符合的case语句再执行default语句。（建议default语句放在最后边）

## 3、if语句与switch语句非常相似，怎么选择呢？

switch的优点：结构清晰

switch的缺点：如果判断的条件是一个范围，使用switch语句操作就非常麻烦了。

# 二十四、while循环语句

## 1、格式

* while（循环条件）{

循环语句；

}

## 2、循环语句注意事项

* while循环语句一般是通过一个变量来控制其循环次数
* while循环语句的循环体代码如果只有一句的时候，那么可以省略大括号。

# 二十五、do-while循环语句

## 1、格式

do {

循环语句 ;

}while(循环条件);

## 2、do-while与while的区别

* do-while循环语句是先 执行 后 判断，不管判断条件是否为true，都至少执行一次。
* while 循环语句是先 判断 后 执行，如果判断条件为false，则不执行循环体。

# 二十六、for循环

## 1、格式

for(初始化语句A；判断语句B；循环后的语句D){

循环语句C；

}

## 2、for循环语句要注意的事项

* for(;;) 这种写法是一个死循环，相当于while(true)。
* for循环语句的初始化语句只会执行一次，只是在第一次循环的时候执行
* for循环语句的循环体语句只有一句话时，大括号可以省略不写，但是不建议省略

## 3、for循环练习

* 打印一个五行五列的矩形

|  |
| --- |
| for(int i=1;i<=5;i++){ // 外层for循环控制行;  for(int j=1;j<=5;j++){ // 内层循环控制列  System.out.print("\*");  }  System.out.println();// 换行  } |

* 打印九九乘法表

|  |
| --- |
| // 打印九九乘法表  for(int i=1;i<10;i++){ // 外层控制行数  for(int j=1;j<=i;j++){ // 内层控制列， j 值不大于 i 的值  System.out.print(i+"\*"+j+"="+i\*j+" ");  }  System.out.println();  } |

# 二十七、转义符 \

## 1、常用的转义符

* \b BackSpace（退格键）
* \t Tab键 （制表符，一个Tab键等于四个空格键）
* \r 回车 （把光标移动到当前行的首位置上，如果后续继续输入，则会将对应位置的内容覆盖掉）
* \n 换行 （把光标移动到下一行的首位之上，不会覆盖上一层内容）

## 2、\r回车 和 \n换行 的区别

* '\r' 回车，回到当前行的行首，而不会换到下一行，如果接着输出的话，本行以前的内容会被逐一覆盖；
* '\n' 换行，换到当前位置的下一行，而不会回到行首；
* Unix系统里，每行结尾只有“<换行>”，即"\n"；Windows系统里面，每行结尾是“<回车><换行>”，即“\r\n”；Mac系统里，每行结尾是“<回车>”，即"\r"；。一个直接后果是，Unix/Mac系统下的文件在Windows里打开的话，所有文字会变成一行；而Windows里的文件在Unix/Mac下打开的话，在每行的结尾可能会多出一个^M符号

# 二十八、break、continue、return关键字

## 1、break

### （1）break的使用范围

* break只适用于switch和for循环中

### （2）break的作用

* break用于switch的作用是 结束一个switch语句。
* break作用于for循环的作用是 结束当前所在的循环语句

### （3）笔试题：break目前位于内层for循环的，如何才能让break作用于外层for循环？

* 可以对for循环化做标记，break作用与外层for循环。
* 标记的命名只要符合标识符的命名规则即可。（字下美人数骆驼）

|  |
| --- |
| // brake位于内层 for循环，怎样让 break作用于外层 for循环？    // 对 for循环添加标记  outer:for(int i=0;i<3;i++){  inner:for(int j=0;j<2;j++){  System.out.println("Hello World!!!");  break outer; // 使 break 作用于外层 outer for循环  }  } |

## 2、continue

### （1）continue作用范围

* continue只能作用域循环语句

### （2）continue作用

* 跳过本次循环内容，继续下一次循环。

### （3）continue注意事项

* continue后边不能跟其他语句，因为是永远也执行不到的
* continue也可以配合标记进行使用。

## 3、标记

* 可以在一行代码的起始位置之前，加上一个标记，用来标记这行代码
* 标记的命名规则需要符合标识符的命名规则

# 二十九、函数

## 函数的格式

修饰符 返回值类型 函数名（形参列表）{

方法体（需要被封装的功能代码）；

}

## 函数的特点

1. 函数定义好之后，需要被调用才会执行的。
2. main函数是jvm调用的，不需要我们手动去调用的。

## 3、如何定义一个函数？

1. 返回值类型
2. 是否存在未知的参数（是否存在要调用者确定的参数）

## 4、注意

* 如果一个函数的返回值类型是具体的数据类型，那么必须保证在任意情况下都保证有返回值。（除了返回值类型是void外）

# 三十、return关键字

## 1、return的作用

（1）返回数据给函数的调用者

（2）函数一旦执行到return关键字，那么马上结束该函数。（能结束一个函数）

## 2、注意：

* 一个函数的返回值类型是void，那么也可以出现return关键字，但是return后边不能有数据。

# 三十一、函数的重载

## 1、函数的重载

* 在一个类中，同时出现两个及以上的同名函数，这个叫做函数的重载。

## 2、函数重载的作用

* 同一个函数名出现了不同的函数，以应对不同个数或不同数类型的参数。

## 3、函数重载的要求

* 函数名一致
* 函数的形参列表不一致（参数个数或对应的参数数据类型不一致）

# 三十二、数组

## 1、什么是数组？

* 数组是同一种数据类型数据的集合容器

## 2、数组的格式

* 数据类型 [] 变量名 = new 数据类型 [长度];
* 数据类型 [] 变量名 = {值1，值2 ……}

## 3、分析数组

* int [] arr = new int[50];

1. 左侧：

* int [] arr 声明了一个int类型的数组变量，变量名为arr
  + int 表示该数组容器只能存储int类型的容器
  + [] 表示这是一个数组类型
  + Arr 变量名

1. 右边：

* new int[50] 创建了一个长度为50的int类型的数组。
  + - new 创建数组对象的关键字
    - int 表示该数组容器只能存储int类型的数据
    - [] 表示这是一个数组类型
    - 50 该数组最多能存储50个数据，数组的容量。

## 4、成员变量与局部变量

（1）成员变量

* 成员变量就是声明在类的内部，方法的外部。也叫类变量。

1. 局部变量

* 如果一个变量是在方法（函数）的内部声明的，那么该变量就是一个局部变量。

## 5、对象数据

（1）基本数据类型数据（8种）

（2）对象数据：八种基本数据类型之外的数据，都是对象数据。

（3）垃圾对象：对象如果没有变量引用了，那么该对象就是垃圾对象。

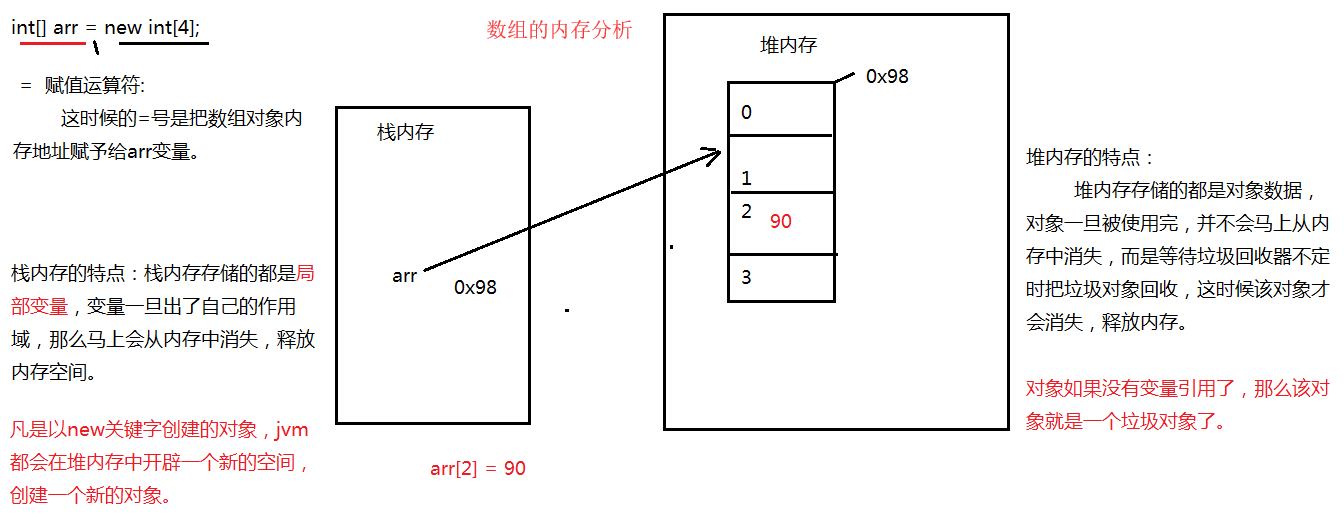
## 6、内存

### （1）栈内存

* 栈内存的特点：栈内存存储的都是局部变量，变量一旦出了自己的作用域，那么马上就会从内存中消失，释放内存空间。

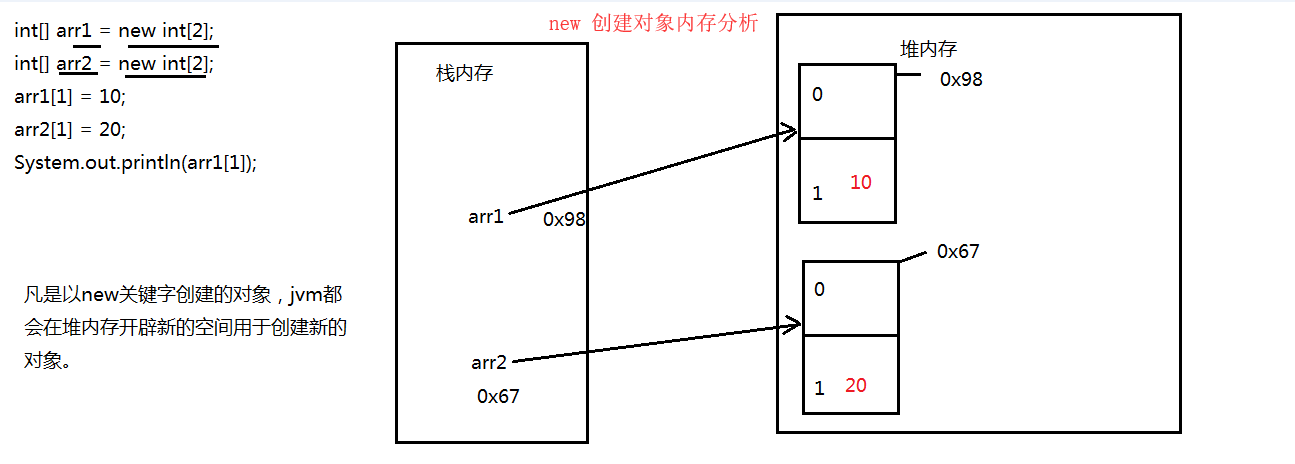
### （2）堆内存

* 堆内存的特点：堆内存存储的都是对象数据，对象一旦被使用完，并不会马上消失，而是等待垃圾回收（GC）不定时把垃圾对象（不再使用的对象）回收，这时候对象才会消失，并释放内存。
* 垃圾对象：对象如果没有变量引用了，那么该对象就是垃圾对象。



## 7、new 关键字

* 凡是以new关键字创建的对象，jvm虚拟机都会在堆内存中开辟一个新的空间，创建一个新的对象。



## 8、数组在内存中的存储

int [] arr = new int [50];

* new int [50]在堆内存中创建了一个对象，分配了一个空间，并赋予一个内存地址。
* int [] arr（局部变量）在栈内存中创建了一个变量
* = 赋值运算符，将堆内存中创建的对象的地址，指向了栈内存中的变量。

## 9、数组的好处

* 数组的好处： 对分配到数组对象中每一个数据都分配一个编号（索引值、角标、下标）,索引值的范围是从0开始，最大是： 长度-1.

## 10、数组的使用

* 数组的有一个length 的属性，可以查看数组 的容量
* 查看数组中的所有数据（for循环）。

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

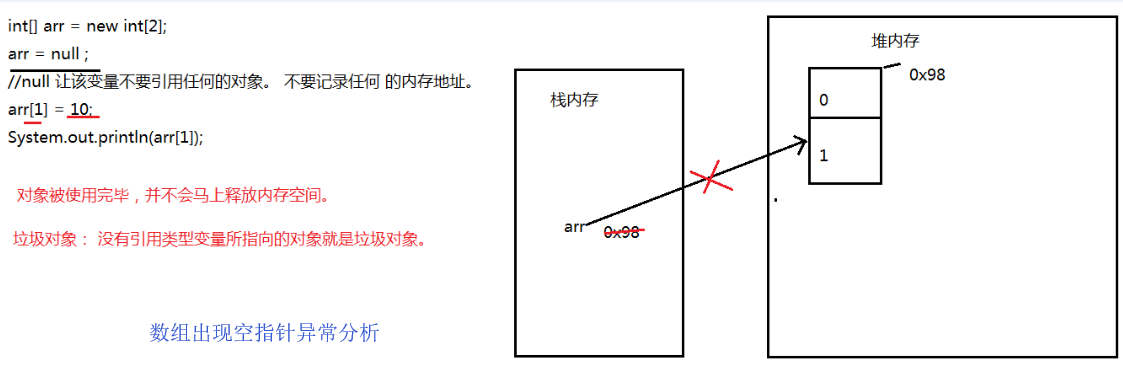
System.out.println(arr[index]);

}

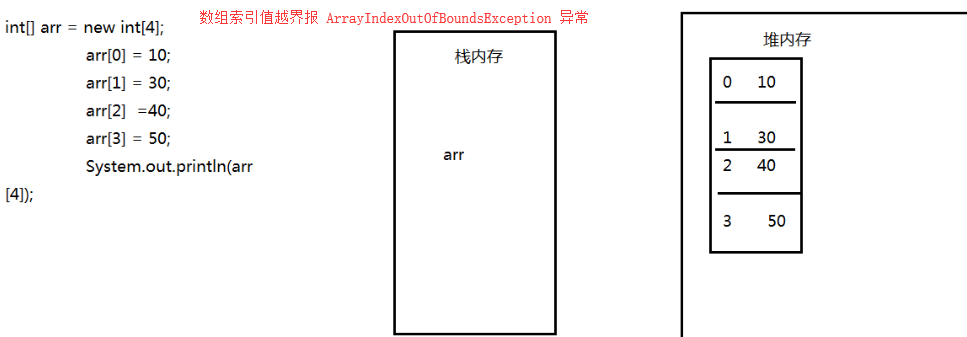
循环遍历数组时，最大长度应写成 小于数组对象的最大长度，不能写死。

## 11、数组常见的问题

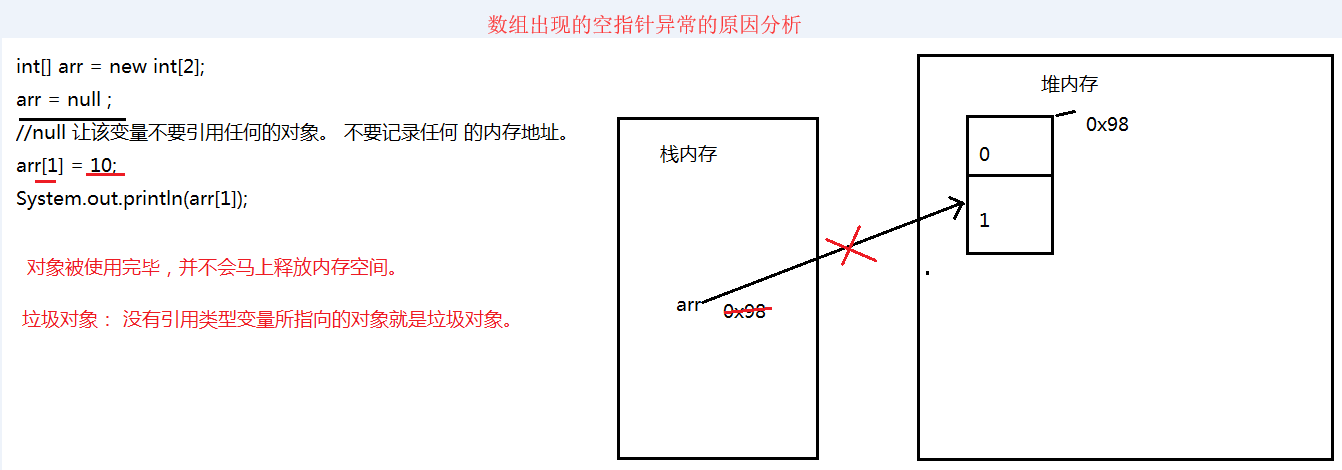
1. NullPointerException空指针异常
   * 原因：引用类型的变量没有指向任何对象，而访问了该对象的属性或者是调用了该对象的方法



1. ArrayIndexOutOfBoundsException 索引值越界。
   * 原因：访问了不存在的索引值



1. null值可以让变量不再引用任何对象，不要记录任何的内存地址。
2. 对象被使用完毕后，并不会马上释放内存，而是会等垃圾回收器（GC）不定时的进行垃圾回收。



## 12、数组的初始化方式

### （1）静态初始化

* 数据类型 [] 变量名 = {元素1，元素2，……};

### （2）动态初始化

* 数据类型 [] 变量名 = new 数据类型[数组长度];

### （3）初始化方式的选择

* 如果一开始就已经确定了数据，那么这时候选择 静态初始化方式。
* 如果数据一开始还不明确，那么这时候建议使用 动态初始化方式。

## 13、排序算法

### （1）冒泡排序

* 就是将两个相邻的元素进行比较，符合条件的进行位置交换

|  |
| --- |
| // 冒泡排序：将两个相邻的元素  // 一共有 arr.length 个元素，最后需要用arr[arr.length-2] 与 arr[arr.length-1]两个元素进行比较  // 一共 arr.length 个元素，只需找出 arr.length-1-j 个最大数即可  // 最大为 arr[arr.length-2] 与 arr[arr.length-1] 进行比较。 所以第一个比较数（外层循环）索引值 不能大于 arr.length-2  public static void selectSort(int [] arr){  for(int i=0;i<arr.length-1;i++){ // 外层循环是需要比较的次数，一共需要比较多少轮  for(int j=0;j<arr.length-1-i;j++){ // 内层循环：将相邻元素进行比较，将符合条件的元素进行交换  if(arr[j]>arr[j+1]){  int temp=arr[j];  arr[j]=arr[j+1];  arr[j+1]=temp;  }  }  }  } |

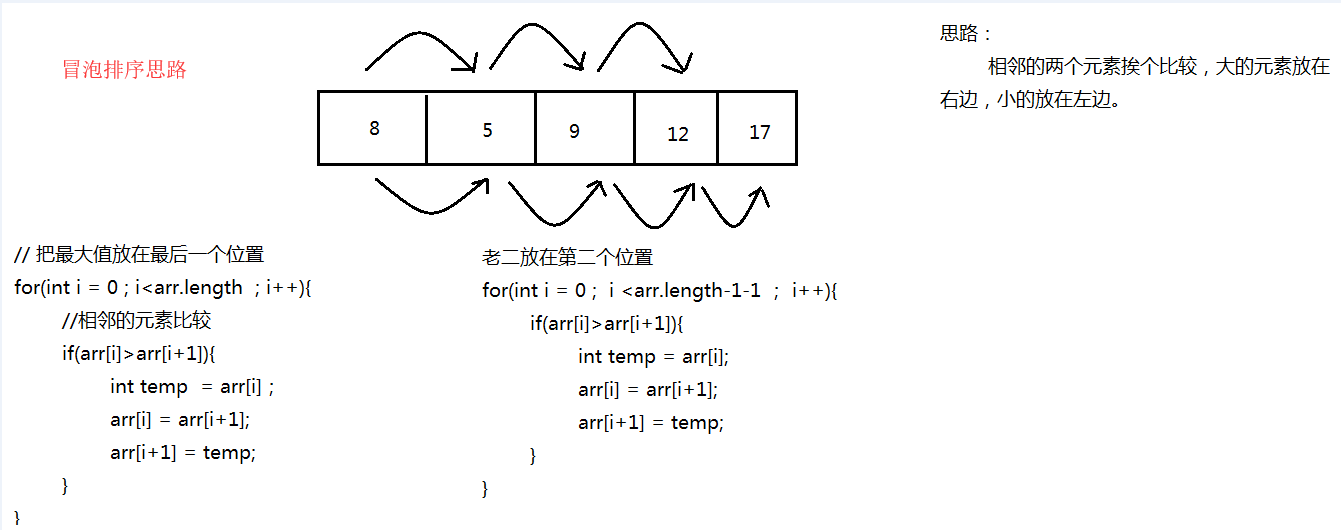
### （2）选择排序（直接排序）

* 使用一个元素与其他元素挨个比较依一次，符合条件的交换位置。

|  |
| --- |
| // 直接排序（选择排序）：使用一个元素与其他的元素挨个比较一次，符合条件的交换位置。  // 用第一个数字与后边所有的数字进行比较，符合条件的交换位置  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]){  int temp=arr[i];  arr[i]=arr[j];  arr[j]=temp;  }  }  }  } |

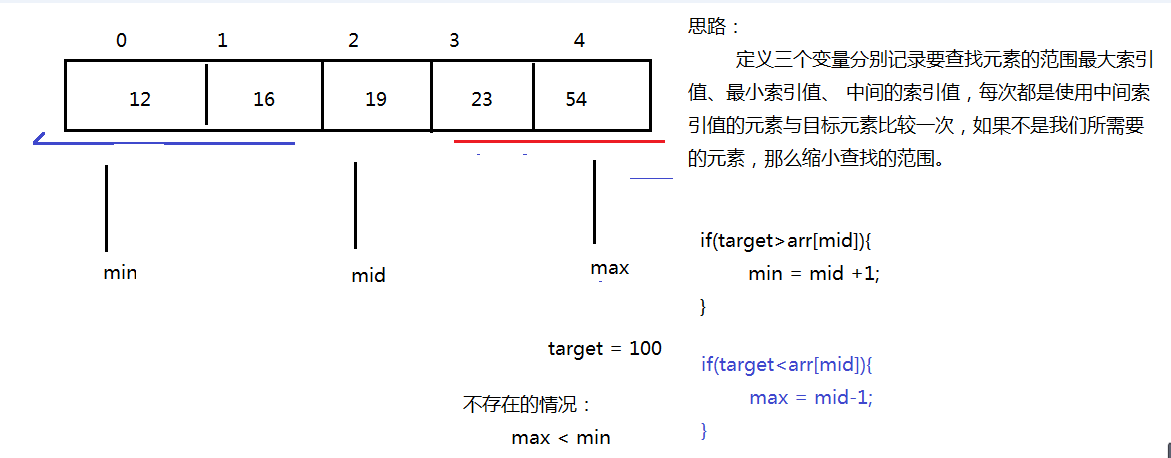
### （3）选择排序与冒泡排序的区别

* 选择排序 是用当前位置的元素与后边所有元素挨个比较，符合条件的交换位置
* 冒泡排序 是从第一个元素开始，两个相邻的元素进行比较，符合条件的交换位置



## 14、二分查找法：

* 用于查找数组内是否存在指定元素



|  |
| --- |
|  |

## 15、翻转数组内的元素

|  |
| --- |
| // 翻转数组内的元素  for(int min=0,max=arr.length-1;max>min;min++,max--){  char temp = arr[min];  arr[min]=arr[max];  arr[max]=temp;  } |

# 三十三、Arrays工具

## 1、sort 冒泡排序

## 2、binarySearch 二分查找法（折半查找法）

## 3、toString 将变量以字符串的形式表现出来

# 三十四、二维数组

## 1、格式

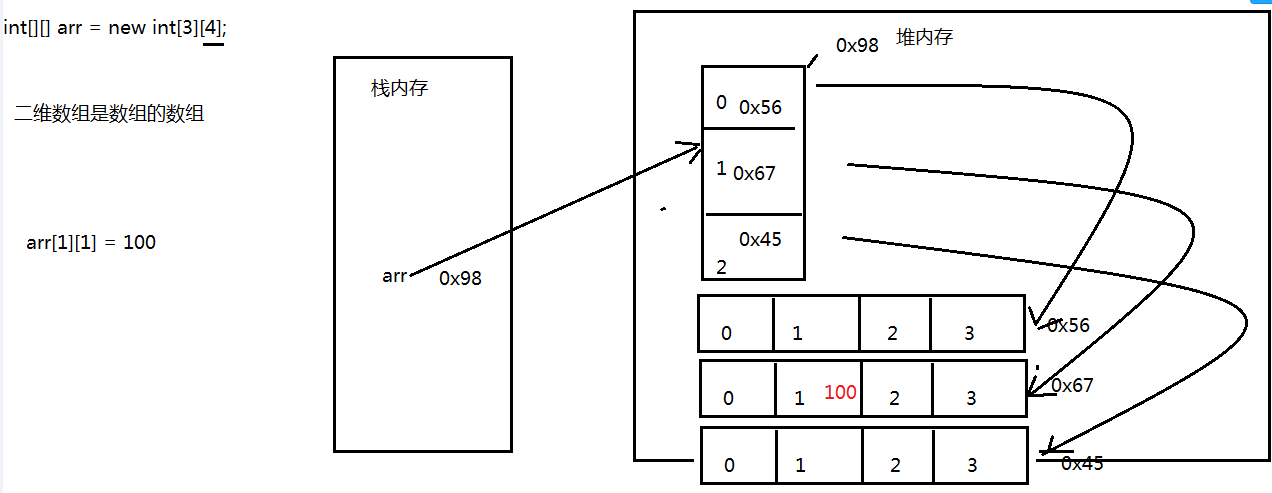
数据类型 [] [] 变量名 = new [长度1] [长度2];

注：

长度1 是变量的长度，即可以包含多少个数组

长度2 是变量中每个数组的长度。

## 2、在jvm中的内存存储方式



## 3、二维数组的初始化方式

### （1）静态初始化

数据类型 [] [] 变量名 = {{元素1，元素2……}，{元素1，元素2……},……}

### （2）动态初始化

数据类型 [] [] 变量名 = new [长度1] [长度2];

注意：

动态初始化二维数组，可以只写长度1，不写长度2。表示创建了一个长度为长度1的空二维数组，数组中没有元素。

### （3）动态构建二维数组

|  |
| --- |
| *// 二维数组* **int**[][] arrs = **new int**[5][8]; *// 构造一个动态初始化的二维数组* **int** f = 6; *// 为二维数组动态赋值* **for** (**int** i = 0; i < arrs.**length**; i++) {  **for** (**int** j = 0; j < arrs[i].**length**; j++) {  arrs[i][j] = f;  f++;  } }  *// 遍历二维数组* **for** (**int** i = 0; i < arrs.**length**; i++) {  **for** (**int** j = 0; j < arrs[i].**length**; j++) {  System.***out***.print(arrs[i][j] + **"\t"**);  }  System.***out***.println(); } |