# JAVA概述

## JAVA历史

SUN参与Green计划.给家电设计一套嵌入式系统.因为硬件问题当时主流的语言C++难以运行,所以需要有一款较小的语言(.小可以理解为体积小).同时因为硬件厂商比较多.需要软件脱离硬件的干扰,所以引入了虚拟机.(Java中使用的是HotSpot虚拟机).

所以Java具备两个很明显的特点

* 小
* 跨平台

## 几个名词介绍

* JDK Java Development Kit(Java开发工具包)
* JRE Java Runtime Environment(Java 运行环境)

## JDK和JRE的区别 (需要知道)

JRE是只负责程序的执行.程序的产生可以分为以下几个阶段

* 编写原码
* 编译原码得到可以执行文件(二进制文件或者是中间文件,中间文件由虚拟机来执行.)

JDK是负责帮助我们实现Java代码的编译和运行的.(JDK中包含JRE,为了开发者方便运行测试)

## 高级语言和低级语言

计算机可以看的懂的就是低级语言只有0和1.在低级语言基础上添加我们容易识别的指令比如add等.那这就是高级语言.目前主流的高级语言可以分为两类:

### 高级语言分类

* 面向过程语言
  + C
* 面向对象语言
  + JAVA

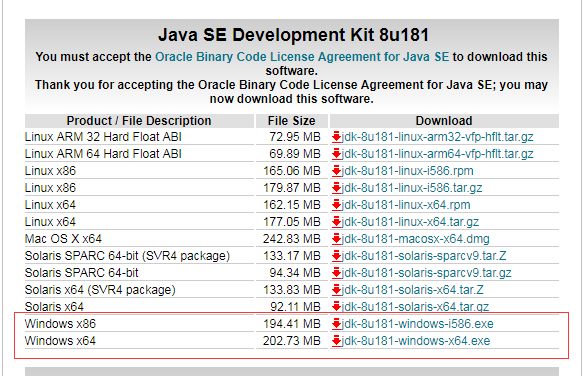
其实语言还可以这样分

* 编译型语言
* 解释型语言

## JDK的安装

### JDK的下载

JDK的下载可以在官网进行下载.[下载地址](https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)



### JDK的安装

JDK的安装堪称傻瓜式安装在不改变安装路径的情况下.一站到底.(如果是已经安装过的.会自动帮你先卸载再安装.)

具体按照自行参考截图.

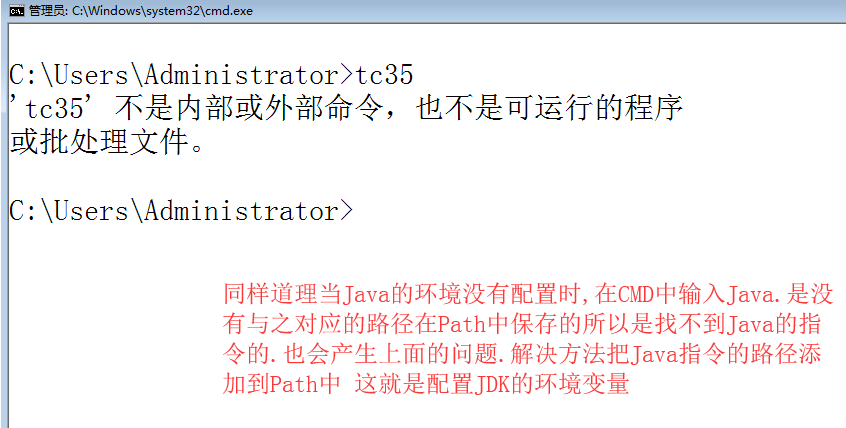
### JDK安装的测试

测试是通过JDK目录下bin目录中的exe文件来进行的.在使用之前我们需要先对JDK进行环境变量配置.

#### 为什么要进行环境变量配置(必须要掌握的)

我们通过在CMD中使用指令(exe文件)来编译和运行代码.比如在CMD(打开方式Window标徽键+R.输入cmd)中输入calc

会弹出计算机.因为CMD有一套环境叫做Path.里面记录了各种目录的详细地址.比如calc的地址就是C:\Windows\System32.当我们在CMD中输入类似的指令时,CMD就会到Path中取寻找(简单理解为把Path中所声明的目录进行遍历,看这些目录中是否有我们所输入的指令文件).如果找到了就执行.如果找不到会报如下错误

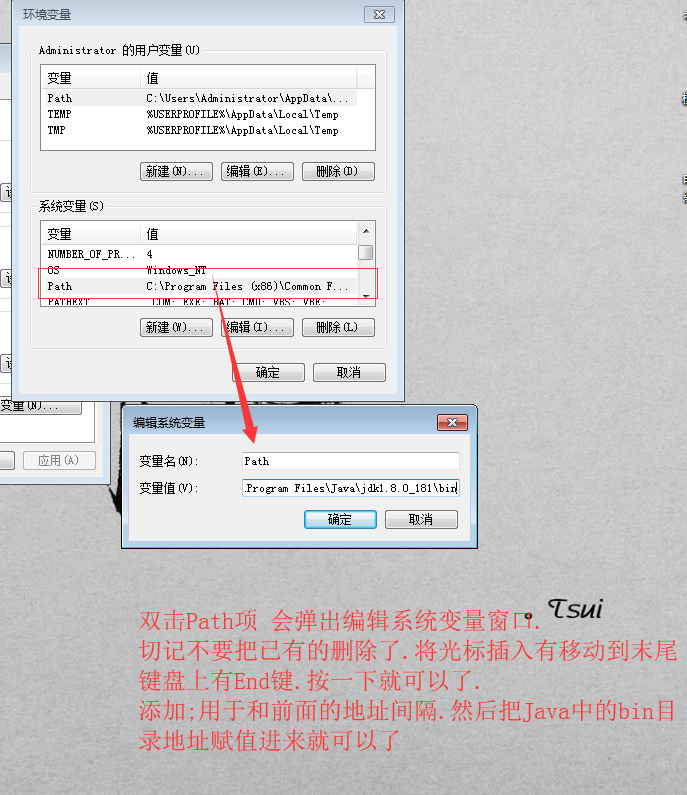


JDK的环境变量全套配置有三个

* JAVA\_HOME
* PATH
* CLASPATH

在JDK1.6之后CLASSPATH可以不配置.所以我们需要配置的只有两个JAVA\_HOME和PATH.最低配置:只配置Path.

我们需要把Java的指令路径添加到Path中.Java的指令在JDK的bin目录下所以我们需要把bin目录的完整地址添加到Path中.



添加完成后可以使用以下三个指令来测试是否安装成功

* Java 运行的指令
* Javac 编译的指令
* Java -version 查看当前版本的指令

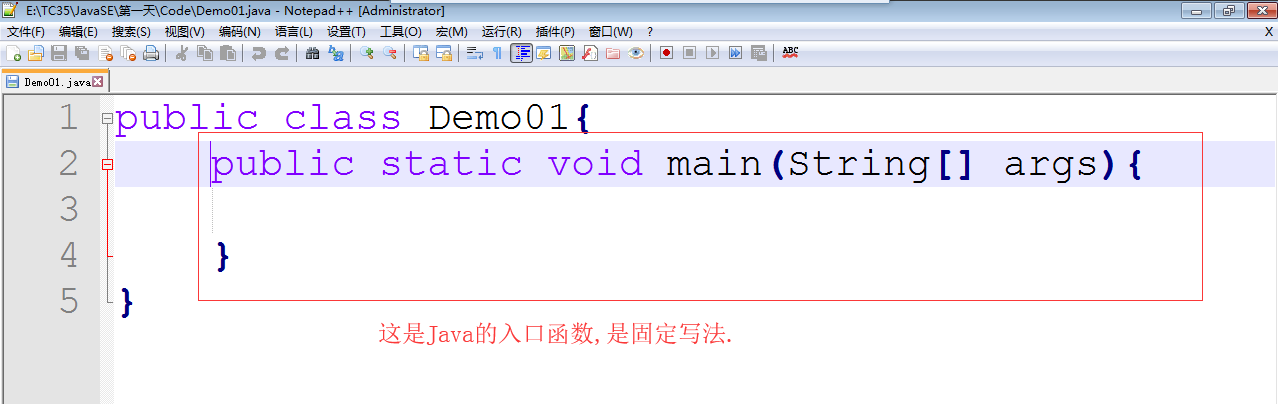
# JAVA开发

Java的原码是保存在.java结尾的文件中.所有的应用都会有一个入口程序.就如同正常的房间都会有正门一样,需要有一个进入的地方.基本上所有的语言,它的入口都叫做main.

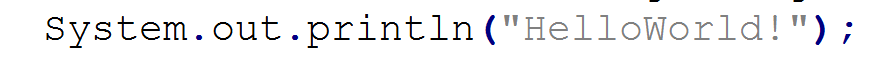
## Java的入口程序(固定写法.不讲道理的)

Java的入口程序写法可以分为以下几个步骤.

1. 提供.java结尾的源文件
2. 提供class.让class的名字和文件名一致
3. 提供入口函数:main
   1. Java的入口函数写法是固定的.

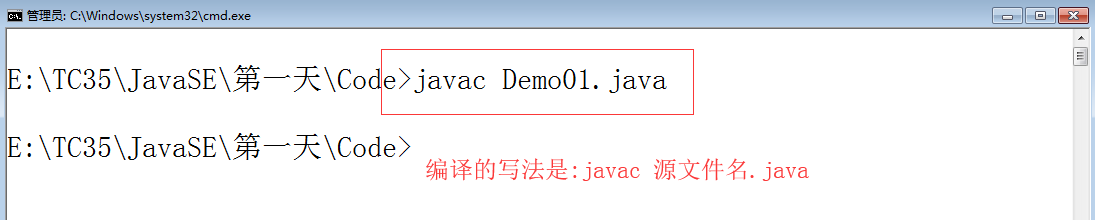


1. 提供输出语句

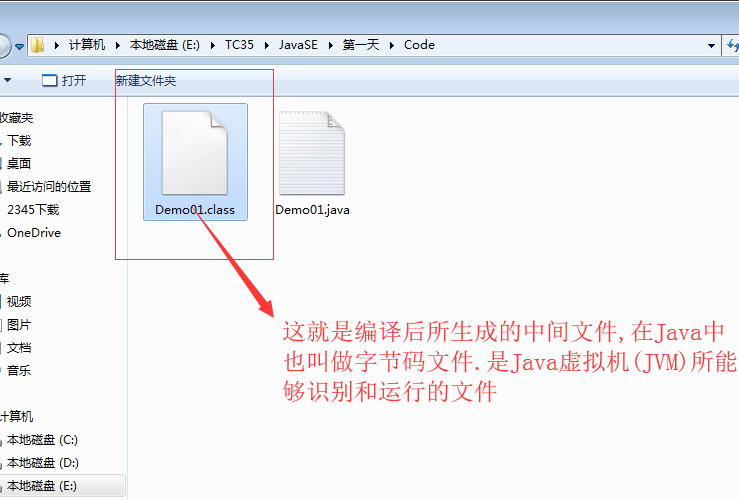


1. 对原码进行编译

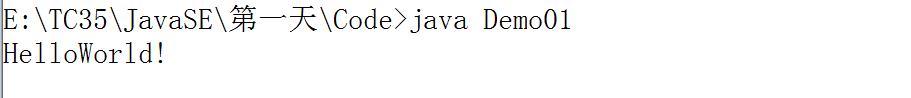
在源文件所在的目录下按住Shift.右键.选择在此处打开命令窗口.



使用Javac指令进行编译.编译后得到字节码文件

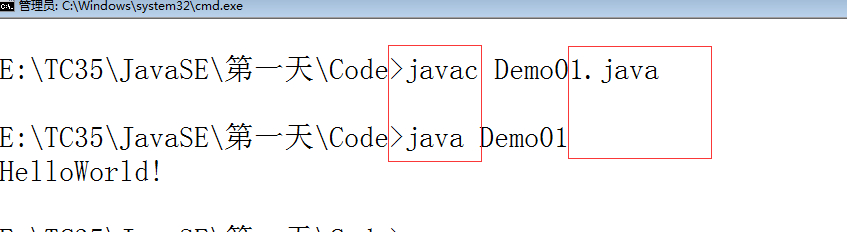


1. 让JRE中的JVM来运行字节码文件

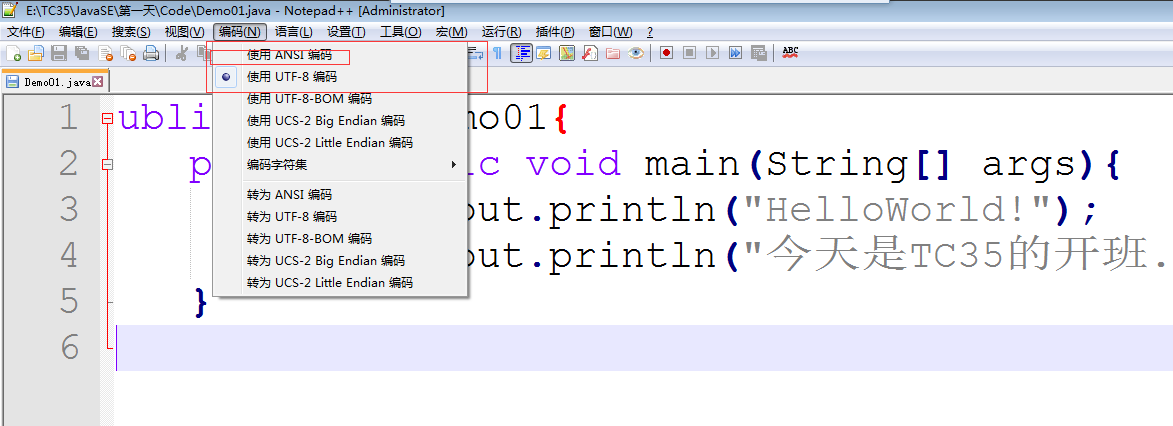
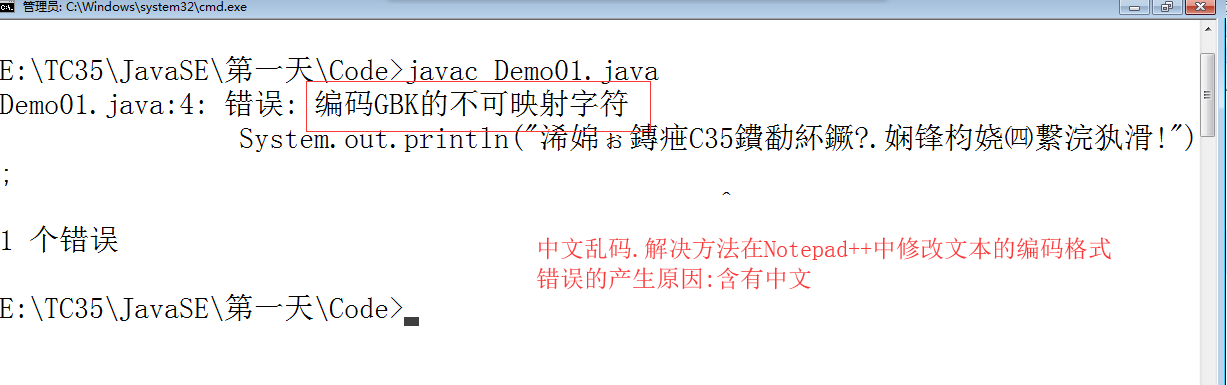


### 编译指令和运行指令写法上的区别

* 编译使用javac.而运行使用java
* 编译指令后面跟的是源文件并且要有后缀
* 运行指令后面跟的是字节码文件.不需要加后缀



## 输出中文时乱码



在修改编码格式的时候记得一定要写剪切整个代码,再修改编码格式,最后再把代码复制回来.

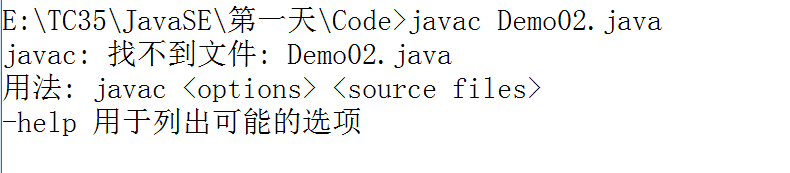
期间可以使用快捷键

* Ctrl + A 全选
* Ctrl + X 剪切
* Ctrl + C 复制
* Ctrl + V 黏贴

## 疑难杂症

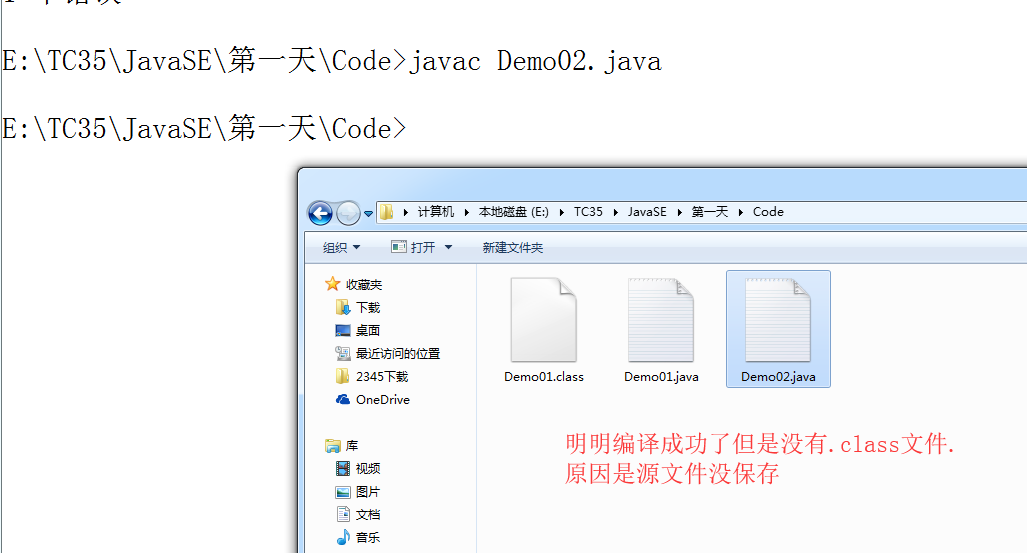
* 保存的文件格式不对

Window默认是隐藏文件后缀名的.所以手工更改容易产生错觉,你觉得你改了,实际并没改.



在Window下修改文件名称有个快捷键F2(单击文件后按F2).为了保障后缀名安全把Window默认的隐藏后缀名取消.

* 及时保存的问题

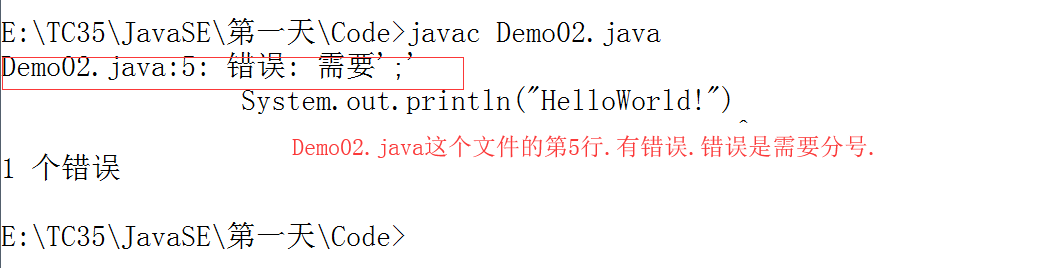


建议养成习惯在代码修改后使用快键键Ctrl + S来进行保存操作.

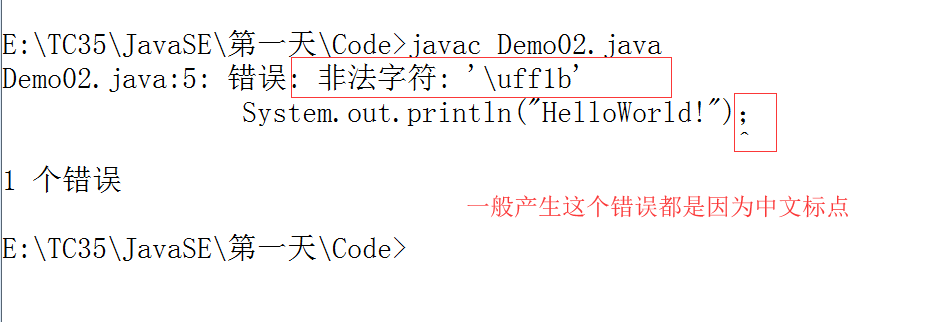
因为我们使用的是Notepad可以注意一个细节就是标签如果是红色就是没保存的.保存之后应该是蓝色的.

* 文件名不合法.Java的源文件文件名中不可以有空格特殊符号等.比如new 1.java.这样的名字是不可以的.
* 代码的规范
  + 换行要缩进.一般默认缩进使用Tab键(制表符键,默认情况下等同于4个空格.禁止敲四个空格)
  + 大括号的写法.大括号一般有两种写法
    - 行末写法
      * 在左大括号前加一个空格
    - 新行写法
* 分号问题

每条语句以分号结尾.



* 中文标点



建议在输入法设置里面选择使用英文标点



* 大小写问题

System和String的首字母是大写的.

## 尝试以下几个写法

* 把class左边的public去掉可以吗
* 文件的名称一定要和class的名称一致吗
* 一个源文件中可以有多少个class
* 输出语句:System.out.println()和System.out.print()的区别
* 输出一个数字.
* 在输出语句中是否可以做加法运算

## 思考问题

* 我不可能把所有的数据都使用输出语句进行打印.有些数据是要被保存的.那怎么保存.

# 预习

* 变量和常量
  + 变量和常量的区别
* 数据类型
  + 数据类型间的转换
  + 进制
    - 进制间的转换
    - 原码反码和补码
* 运算符
  + 算术
  + 关系

[作业复习 3](#_Toc526868766)

[Java的注释 4](#_Toc526868767)

[变量和常量 4](#_Toc526868768)

[字面常量 5](#_Toc526868769)

[变量的声明 5](#_Toc526868770)

[变量的使用 5](#_Toc526868771)

[数据类型 6](#_Toc526868772)

[原生数据类型 6](#_Toc526868773)

[不同分类的默认类型 7](#_Toc526868774)

[字符类型的两种写法 7](#_Toc526868775)

[布尔数据类型的赋值是否可以用其它来代替 8](#_Toc526868776)

[在不同平台Java数据类型所占大小是否有改变 9](#_Toc526868777)

[原生数据类型的具体大小 9](#_Toc526868778)

[数据类型间的转换(必须掌握) 9](#_Toc526868779)

[浮点数的注意点 11](#_Toc526868780)

[进制(了解就行) 11](#_Toc526868781)

[常用的进制 11](#_Toc526868782)

[二进制 12](#_Toc526868783)

[八进制 12](#_Toc526868784)

[十进制 12](#_Toc526868785)

[十六进制 12](#_Toc526868786)

[进制间的准换 12](#_Toc526868787)

[练习 13](#_Toc526868788)

[参考链接 13](#_Toc526868789)

[原码反码补码(了解一下) 13](#_Toc526868790)

[负数的原码求反码 13](#_Toc526868791)

[负数的反码码求补码 14](#_Toc526868792)

[运算符(必须掌握的) 14](#_Toc526868793)

[运算符所涉及的基本概念补充 15](#_Toc526868794)

[操作数 15](#_Toc526868795)

[左值和右值 15](#_Toc526868796)

[算术运算符 16](#_Toc526868797)

[关系运算符 17](#_Toc526868798)

[关系运算的结果是什么数据类型. 17](#_Toc526868799)

[逻辑运算符 18](#_Toc526868800)

[赋值运算符 19](#_Toc526868801)

[算术和赋值运算中的特殊点 20](#_Toc526868802)

[自增和自减运算 20](#_Toc526868803)

[练习题 21](#_Toc526868804)

[算术+赋值运算 22](#_Toc526868805)

[自增/自减与算术+赋值的特殊点 23](#_Toc526868806)

[位运算符 23](#_Toc526868807)

[其它运算符(三目运算符) 23](#_Toc526868808)

[Java中从控制台录入数据(熟练使用.大宝天天见) 23](#_Toc526868809)

[可能遇到的错误操作 25](#_Toc526868810)

[思考 25](#_Toc526868811)

# 作业复习

* 把class左边的public去掉可以吗
  + 可以去掉
* 文件的名称一定要和class的名称一致吗
  + 文件名称是可以和class的名称不一致的.编译不受名称影响.但是在执行的时候需要注意所生成的字节码文件的名称是文件的名称还是class的名字.(测试结果是class的名字)
* 一个源文件中可以有多少个class
  + 可以
  + 一个源文件中可以有多个class.但是有且只有一个class能够被public修饰,并且这个class的名字一定要和文件名称一样.
* 输出语句System.out.println()和System.out.print()的区别
  + 可以
  + Print打印完成后不会换行
* 输出一个数字.
  + 可以
  + 只有输出数字.真假(true/false)是不加双引号
  + 如果要输出的是字符需要用单引号
  + 输出其它的所有内容都必须用双引号
* 在输出语句中是否可以做加法运算

# Java的注释

Java中有三种注释

* 单行注释 //
  + 单行注释在规范上不推荐使用行末注释(必须遵守的)
* 多行注释 /\* \*/
* 文档注释(也叫作JavaDoc注释) /\*\* \*/

# 变量和常量(必须掌握)

应用程序在计算机中运行的时候会产生各种数据.比如当前运行的word那就会有涉及到字体大小的数据,字体颜色的数据等.这些数据都是保存在内存中的.它保存在一个叫做变量的地方.变量就是一个容器.

变量的作用是保护,存储数据并且方便查找使用.因为变量有名称,如果我们需要使用被保存的值.只要通过变量的名称就可以操作该数值了.在所有语言中变量声明格式基本一样,写法是:数据类型 变量名称 = 值

在Java中声明变量的格式如下

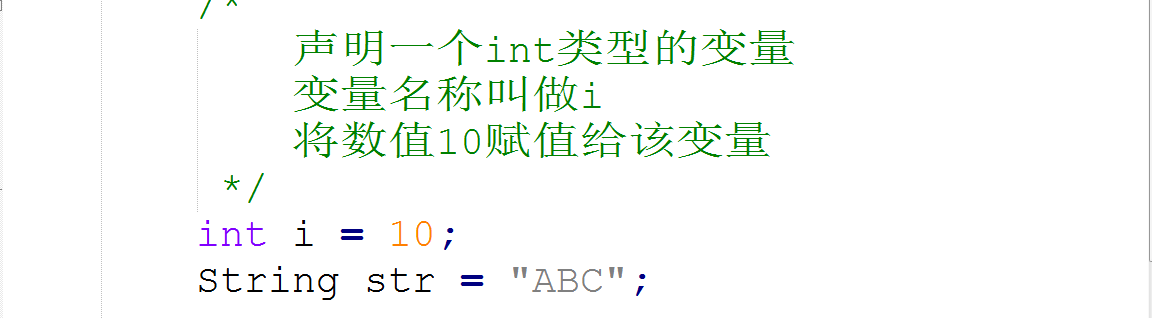
数据类型先教大家两种:

* 整数:int
* 字符串:String

## 字面常量

凡是以数值本身出现的都叫做字面常量.字面常量在内存中是唯一的.

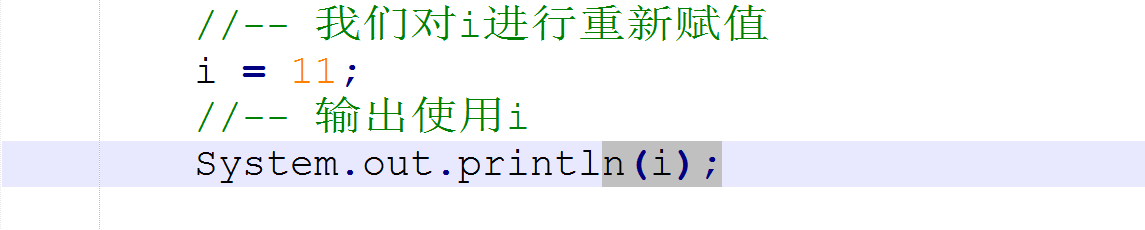
## 变量的声明



## 变量的使用

变量的使用是通过变量名称来实现的.使用方式一般可以分为两种使用:

* 重新赋值
* 输出使用



# 数据类型

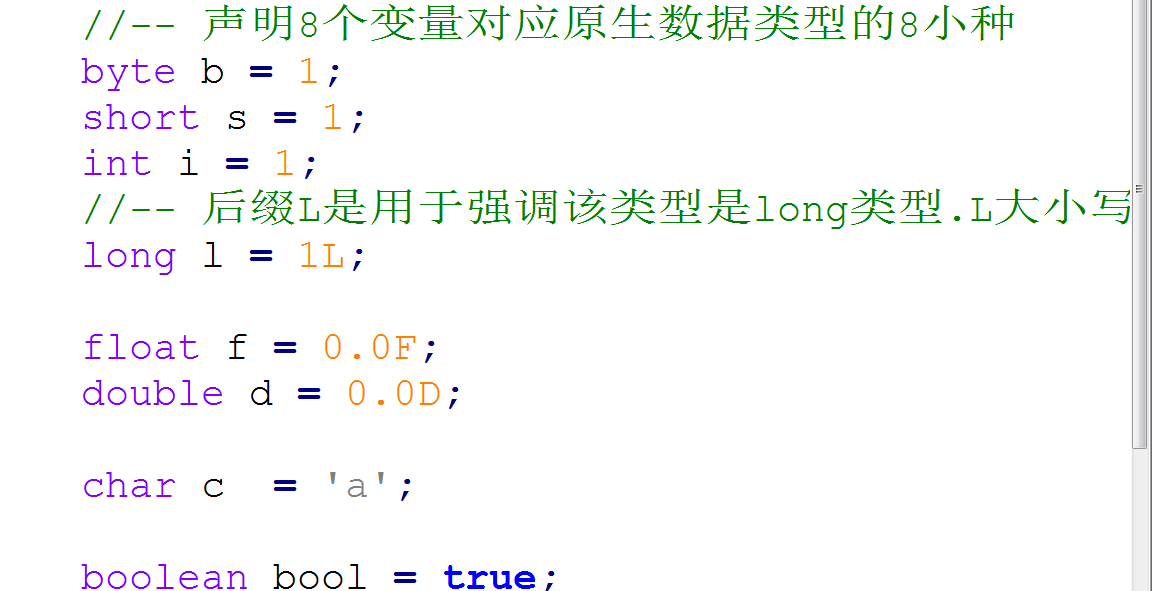
在Java中有两大类数据类型:

* 原生数据类型
* 引用数据类型
  + 只要不是原生类型的都是引用类型
    - String是什么类型?String不再原生的4类8种中.所以是引用

## 原生数据类型

Java中的原生数据类型可以分为4大类8小种

|  |  |  |  |
| --- | --- | --- | --- |
| 分类 | 具体类型 | 大小范围 | 示例 |
| 整数类型 | byte 字节 | 1字节(8bit) |  |
| short 短整型 | 2字节-~+-1 |  |
| int 整型 | 4字节 |  |
| long长整型 | 8字节 |  |
| 浮点类型 | float 单精度 | 4字节IEEE754规范 |  |
| double 双精度 | 8个字节IEEE754规范 |  |
| 字符类型 | char 字符(单字符) | 2字节 |  |
| 布尔类型 | boolean 布尔型 | 1字节.实际占1/8 |  |



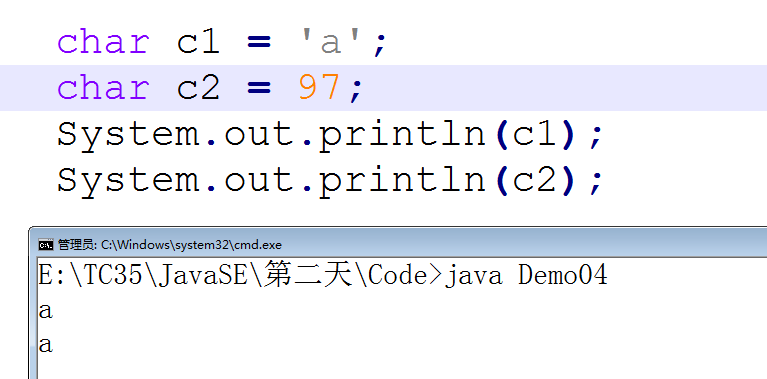
## 不同分类的默认类型

|  |  |
| --- | --- |
| 数据分类 | 数据类型 |
| 整数分类 | int |
| 浮点类型(就是小数) | double |
| 字符类型 | char |
| 布尔类型 | boolean |

## 字符类型的两种写法

Java中的字符类型有点特殊.它有两种赋值方式

* 十进制赋值
* 字符赋值



Java采用的是Unicode编码.Unicode码中包含了ASCII编码.而char的取值对应的就是ASCII码中的内容.所以我们在使用char类型的时候.如果赋值不带有单引号就默认为是使用十进制赋值.在输出的时候会在ASCII码表中寻找十进制值所对应的字符并输出.

## 布尔数据类型的赋值是否可以用其它来代替

这里其实是针对有C语言基础的.在C中有这么一句非0为真.所以在C中一般都是人为的定义宏来作为真假使用.而Java直接提供了布尔类型.该类型是不具备和其它任意类型进行转换的.

布尔类型只有两个值.不可以赋值其它内容.

* 真true
* 假false

## 在不同平台Java数据类型所占大小是否有改变

该问题也是针对C语言的在.16位,32位和64位机器上C语言中int有不同的大小.那Java如果遇到等同情况是否也存在不同大小呢?

Java不存在这个问题.因为Java直接面向的是虚拟机.不直接和硬件挂钩.只要虚拟机的设置是固定的,那在不同的平台上.Java的数据类型大小是一样的.

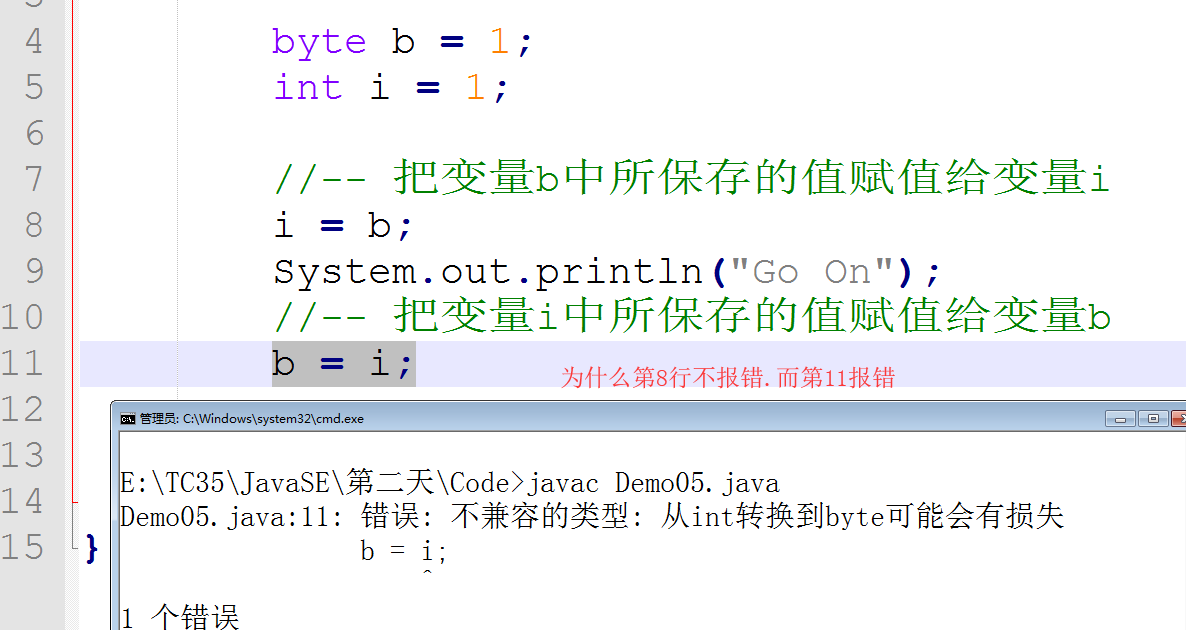
## 原生数据类型的具体大小

在计算机中最小的单位是bit(位).每位的取值要么0要么1.而字节是我们使用的最小的单位.1字节等于8位.所以1个字节的取值可能是.而整数是有正负之分的(Java是有符号语言,这个符号指的是正负号,使用sign.如果是无符号就是unsign).即需要对进行平分.等于256,平分即-128~+128.这里面要考虑0的存在.所以对上面的值进行修改得到-128~+127.正数一般省略前面的+,所以就是-128~127.

## 数据类型间的转换(必须掌握)

Java中原生数据类型之间是可以互相转换的(引用数据类型第七章之前不讨论).布尔类型除外.即我们所讨论的转换是指原生中的剩余7种类型之间的转换.这7种之间的转换可以分为两种转换方式:

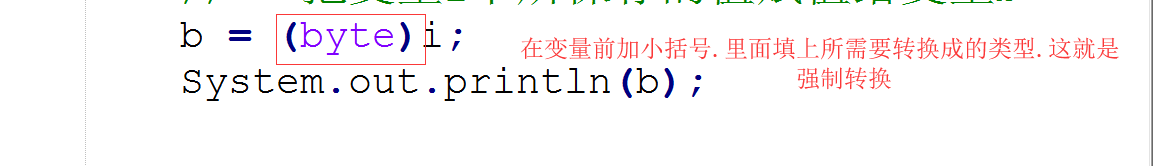
* 强制转换(把大的赋值给小的)
  + 因为看的见,所以也叫做显式转换.
* 自动转换(把小的赋值给大的)
  + 因为看不见,所以也叫做隐式转换.



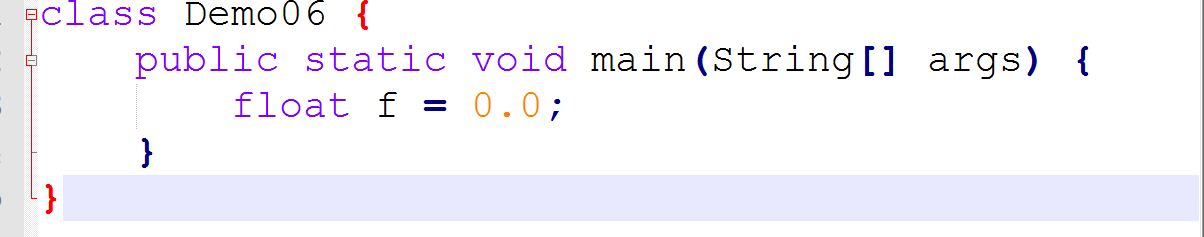
举一个例子.有俩瓶子一个500ML一个1000ML.问把那个瓶子中的内容导入那个瓶子会产生溢出现象.只有把10000ML的倒入500ML的会产生.

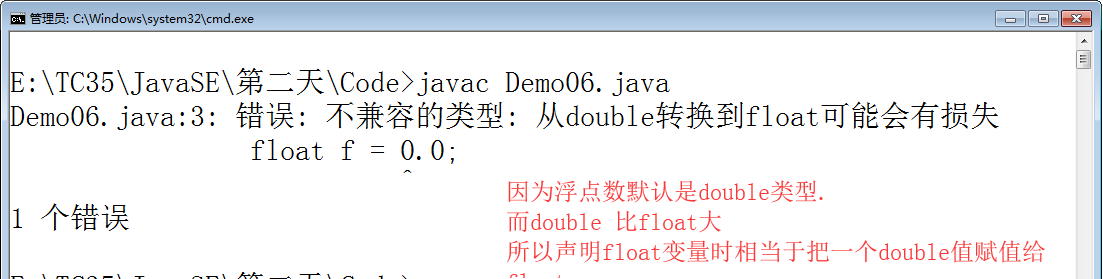
同理把这个例子理解到上面的赋值语句中.赋值语句中涉及两个类型:byte和int.int大.所以我们把byte赋值给int,对于int来说可以容纳下byte的数值.但是反过来把int赋值给byte.byte装不下int的内容.比如把+128赋值给byte.byte就存储不了.所以到时赋值失败!

遇到这种情况我们就需要使用强制转换.写法如下



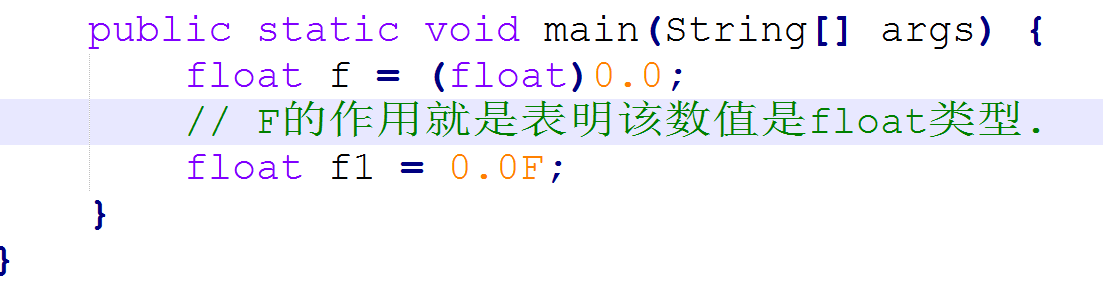
## 浮点数的注意点





解决方法有两个

* 强转
* 在数值的后面添加F



# 进制(了解就行)

## 常用的进制

进制实际就是进位的意思.比如古代结绳计数

### 二进制

逢二进一

### 八进制

逢八进一

### 十进制

逢十进一

### 十六进制

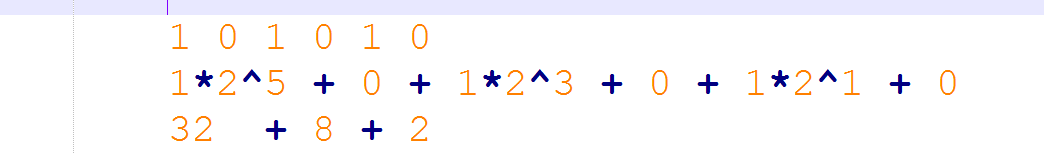
十六进制的取值是0-9和A-F

逢十六进一

## 进制间的准换

十进制转二进制.使用短除2.把每次短除2的余数保留下来.从下向上读就是该十进制数对应的二进制数

二进制转十进制.用每个值乘上2的X次方值.X的取值是该数在二进制写法中的位数.位数从0开始



### 练习

* 把十进制100转成二进制
  + 1100100
* 把1001001转成十进制
  + 73
* 08是几进制(这是曾经的一道面试)
  + 这那个进制都不是.因为写法不对.
    - 二进制以0b作为前缀(仅限在代码保存成int时使用)
    - 八进制是以0作为前缀开始的.比如07
    - 十六进制是以0x作为前缀的.比如0x1F
      * 1代表16.F代表15.加起来对应的十进制就是31

## 参考链接

[进制间的转换](https://jingyan.baidu.com/article/495ba84109665338b30ede98.html)

## 原码反码补码(了解一下)

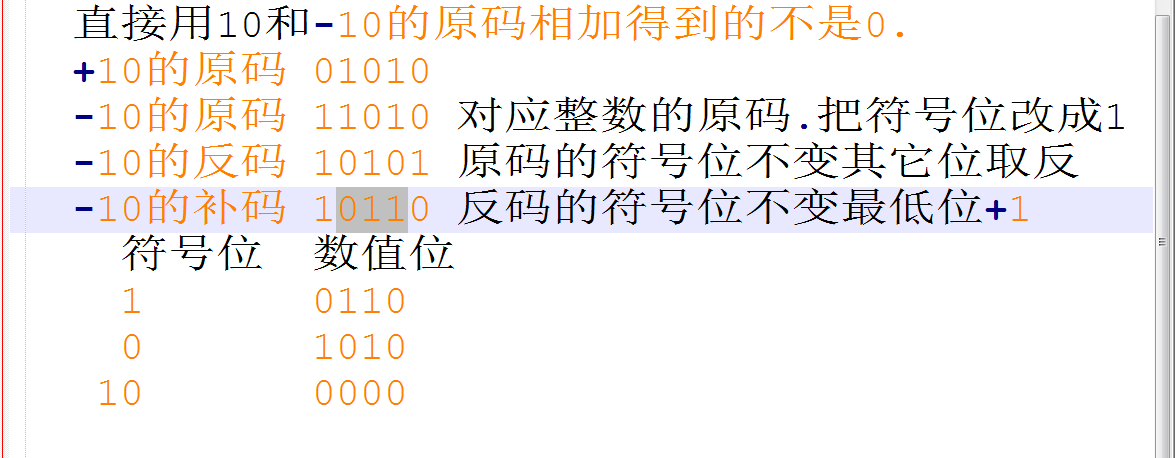
原码反码补码其实是针对负数的.因为正数原反补都是自身.而计算机在处理数据时都是按照补码来处理的.

### 负数的原码求反码

最左边的一位是符号位(也是高位).求反码时高位不变.其它位取反.

### 负数的反码码求补码

符号位不变,最低位+1



# 运算符(必须掌握的)

所有的计算机语言都会有运算符,并且都基本分为以下四类

* 算术运算符
* 关系运算符
* 逻辑运算符
* 赋值运算符
* 位运算符
  + 今天不讲(最后补充)
* 其它运算符

## 运算符所涉及的基本概念补充

### 操作数

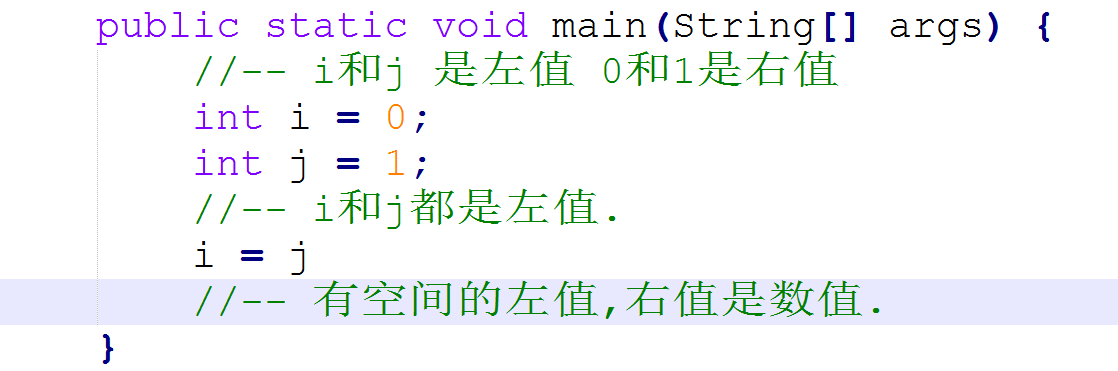
运算符左右两边的被叫做操作数.比如加法a+b.那在+号左右两边的a和b就是操作数.

根据操作数的多少运算符常见的可以分为以下几种

* 一元运算符
  + 比如表达是正数还是负数的+/- :+1
  + 取反也是一元运算符!
* 二元运算符
  + 常见的数学运算基本都是 1+2
* 三元运算符
  + ?: 等到明天条件结构讲完时学.

### 左值和右值

特别是针对赋值符号有左值和右值之分.比如下面代码



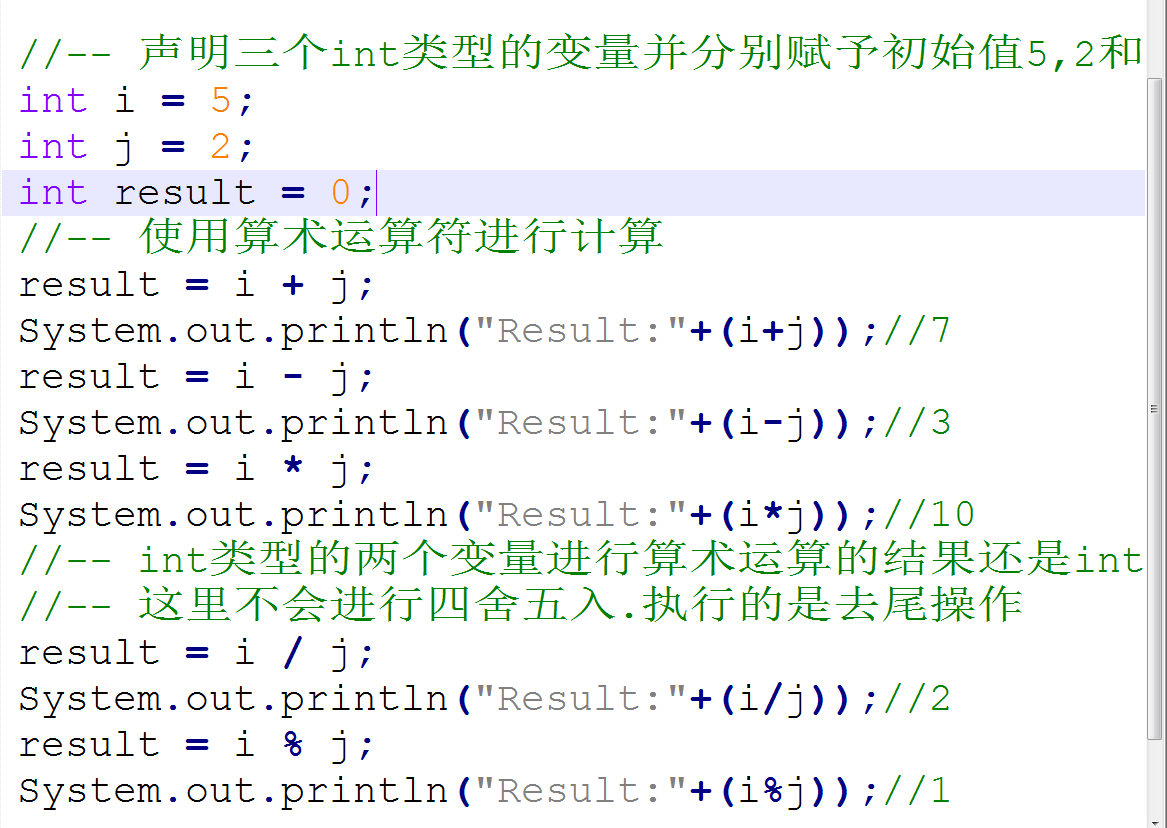
所以要求等号左边必须是左值.至于等号右边.左右值都可以

因此只有左值才有空间放下数据.

## 算术运算符

算数运算可以理解为就是数学中的加减乘除以及求余数5个运算符.

|  |  |
| --- | --- |
| 运算符 | 含义 |
| + | 求加法 |
| - | 求减法 |
| \* | 求乘法 |
| / | 求除法 |
| % | 求余数 |



把等号右边赋值给等号左边.最终的类型由等号左边决定.可以尝试把result的类型修改为其它类型测试.

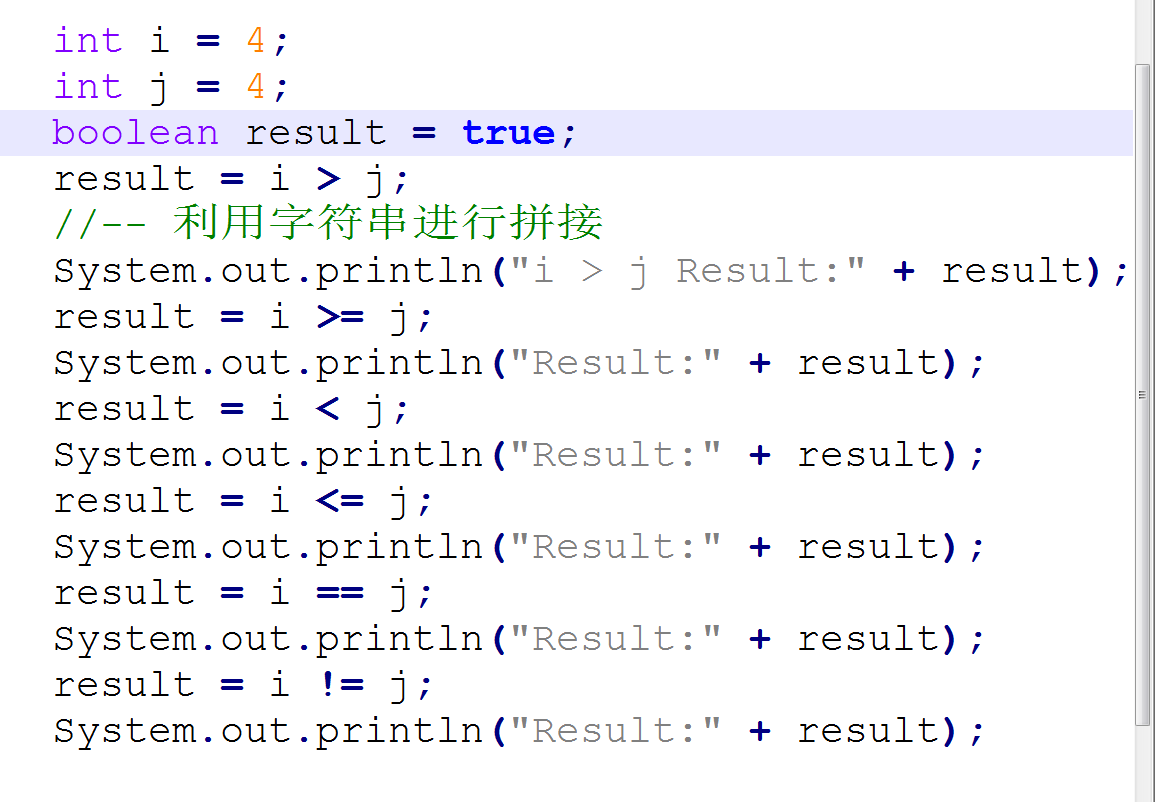
## 关系运算符

关系就是关系的意思.有哪些关系

|  |  |
| --- | --- |
| 运算符 | 含义 |
| > | 大于 |
| >= | 大于或等于(只要满足一个就行) |
| < | 小于 |
| <= | 小于或等于(只要满足一个就行) |
| == | 等于 |
| != | 不等于 |

### 关系运算的结果是什么数据类型.

关系运算的结果是布尔类型.



## 逻辑运算符

逻辑运算符是专门用于处理布尔类型之间的运算.主要体现在以下三点

|  |  |  |
| --- | --- | --- |
| 运算符 | 含义 | 结果 |
| && | 逻辑与 | 同为真时为真. |
| || | 逻辑或 |  |
| ! | 取反 |  |

逻辑运算符所操作的操作数必须是布尔类型或者是能够产生布尔值的表达式



## 赋值运算符

赋值运算符就比较简单.凡是带有等号的都是赋值运算.赋值运算需要注意是从右向左运算

在使用赋值时需要注意两点

* 等号左边必须是左值(等号左边必须有空间)
* 等号右边必须是一个值或者是能够产生之的表达式或者是能够提供值的变量
* 赋值需要遵守一个原则:等号左右两边的数据类型必须等价

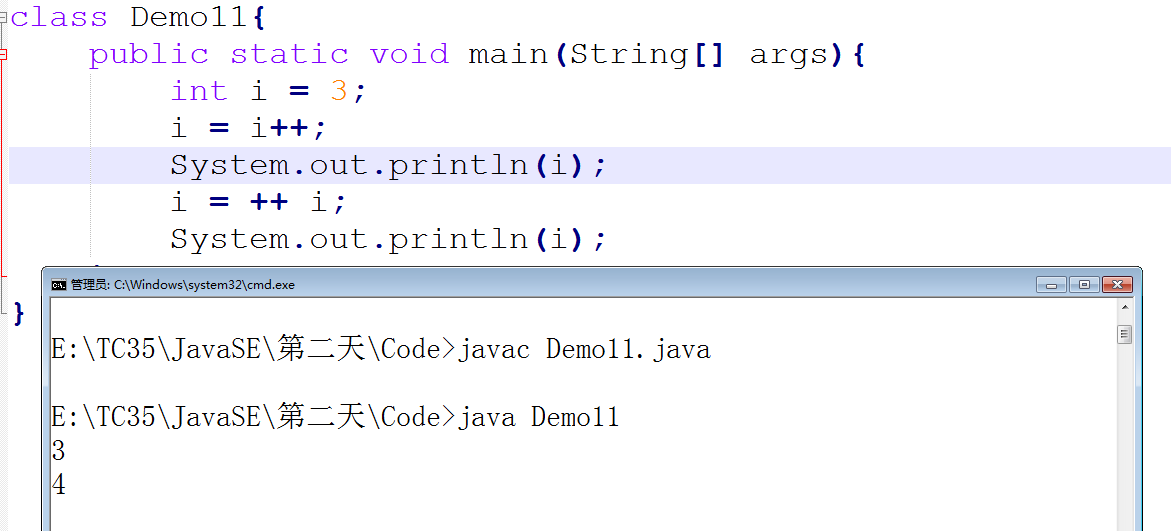
## 算术和赋值运算中的特殊点

### 自增和自减运算

自增和自减就是++和--.本身没有问题.但是有前缀和后缀之分.以自增为例就存在两种写法

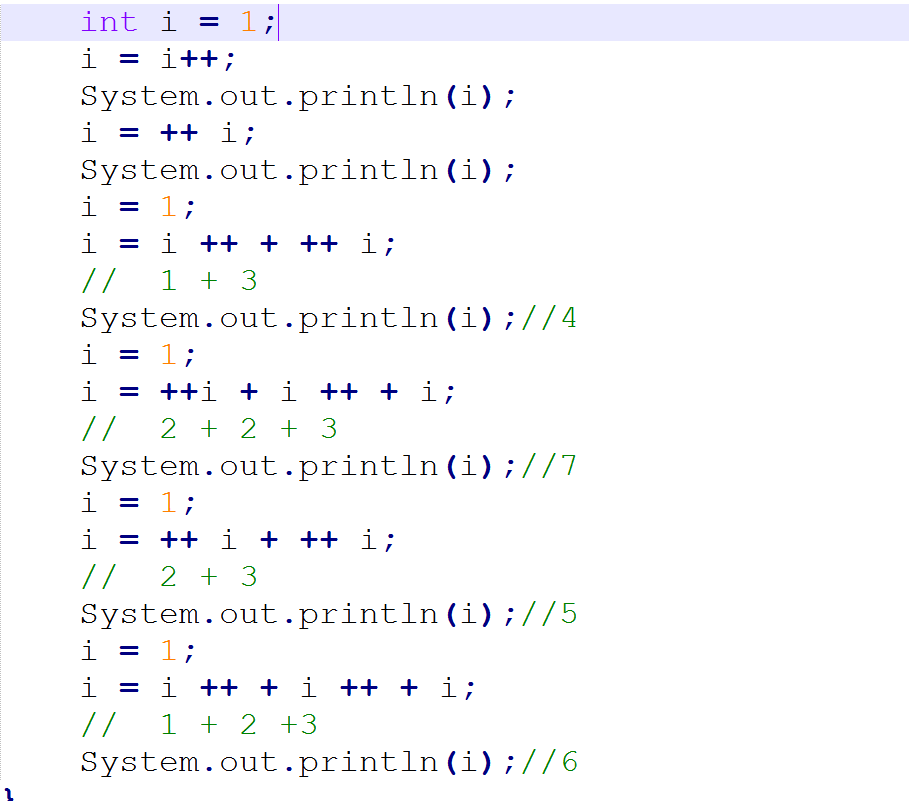
* ++i
* i++

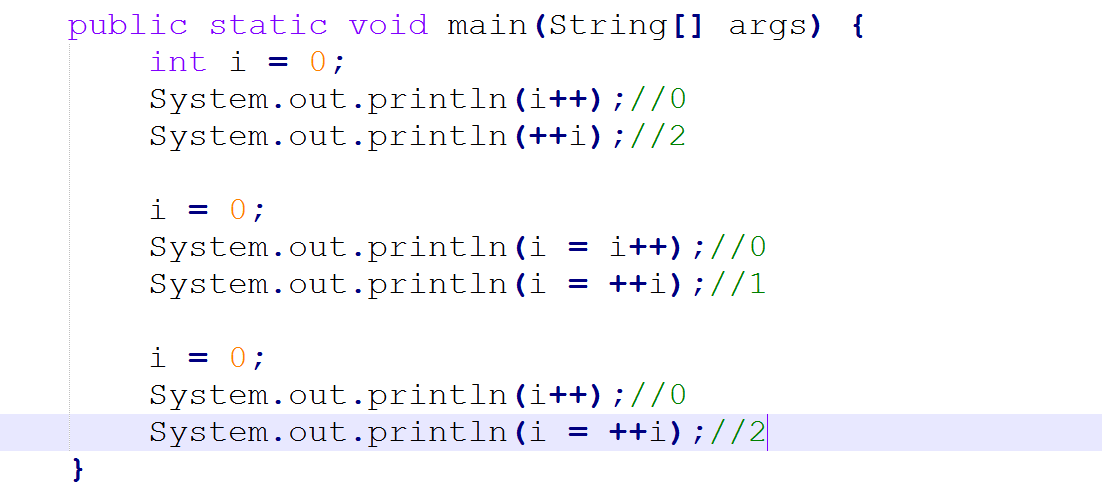
这两种写法如果单独存在都是自加1的意思.如果遇到其它运算就有区别了比如下面



++在后是先输出(把i的值传递出来)再执行自增运算

++在前是先自增(把i的值+1).再把i的值传递出来





### 练习题

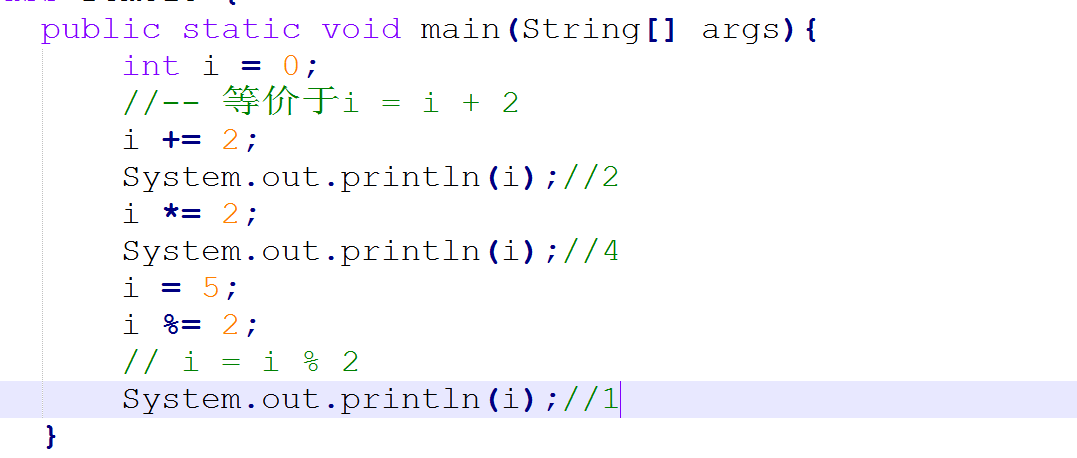
* 假设有变量i其值为1.要求使用9个加号最终结果为5.

### 算术+赋值运算

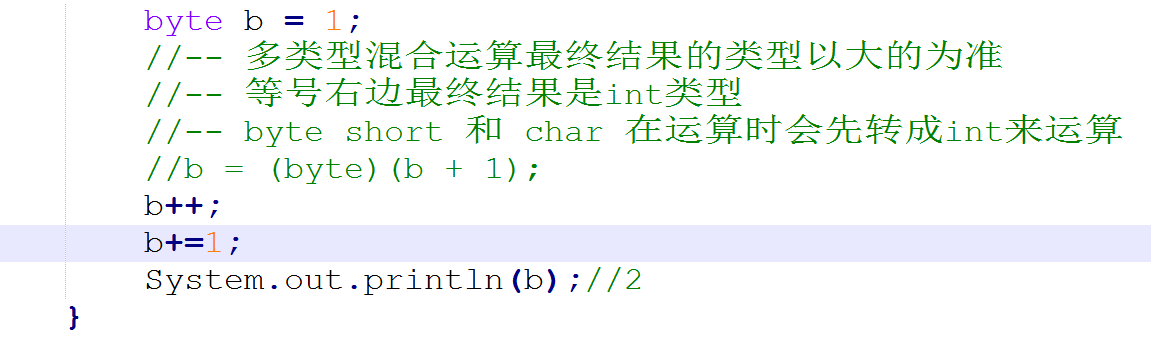
是赋值运算的一种.可以说每一种算数运算符都可以有与之对应的赋值运算

|  |  |
| --- | --- |
| 算术 | 赋值 |
| + | += |
| - | -= |
| \* | \*= |
| / | /= |
| % | %= |

除此以外位运算可以存在与之对应的位+赋值运算.但在实际的使用过程中以算数+赋值的使用率比较高一些.



### 自增/自减与算术+赋值的特殊点



使用这两种运算符在遇到byte short和char这三种数据类型的时候是不会改变数据类型的.(不需要我们进行强转操作)

## 位运算符

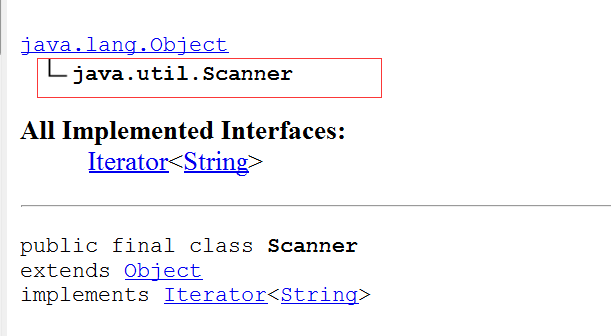
## 其它运算符(三目运算符)

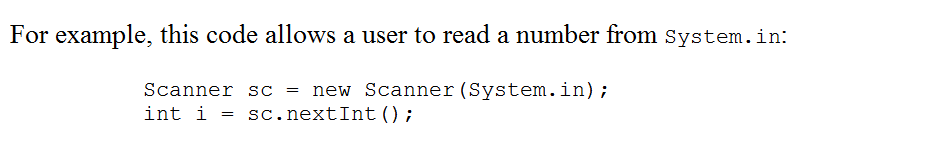
# Java中从控制台录入数据(熟练使用.大宝天天见)

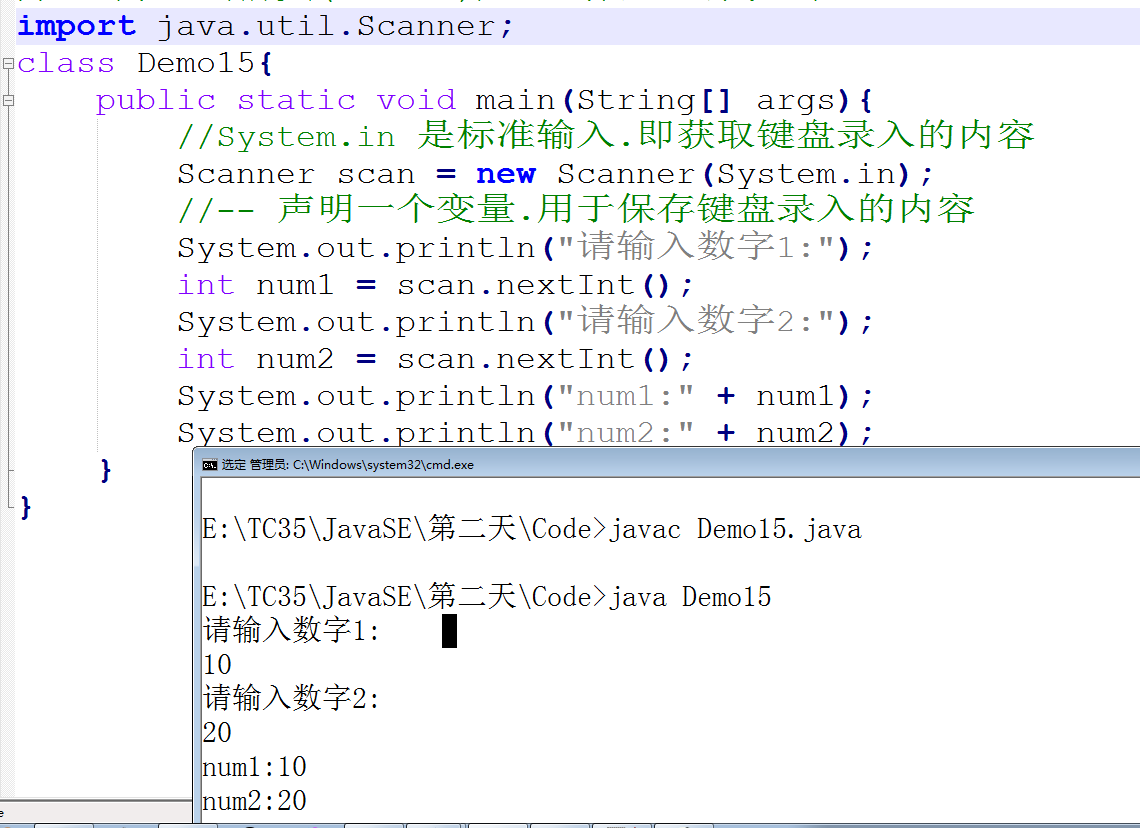
在Java中.有庞大的类库供我们使用.这些是SUN和Oracle已经为我们提供好的.只要按照他们规定的使用规范就可以直接使用.

想要从控制台录入数据需要使用一个工具类:Scanner.下面是它的使用流程

* 告诉当前类我要使用一个工具类了
  + 就是导包
* 创建Scanner的变量
  + 到引用哪里就是对象(今天先知道这个名字.下周再说)
* 使用已经定义好的工具来获取键盘录入.

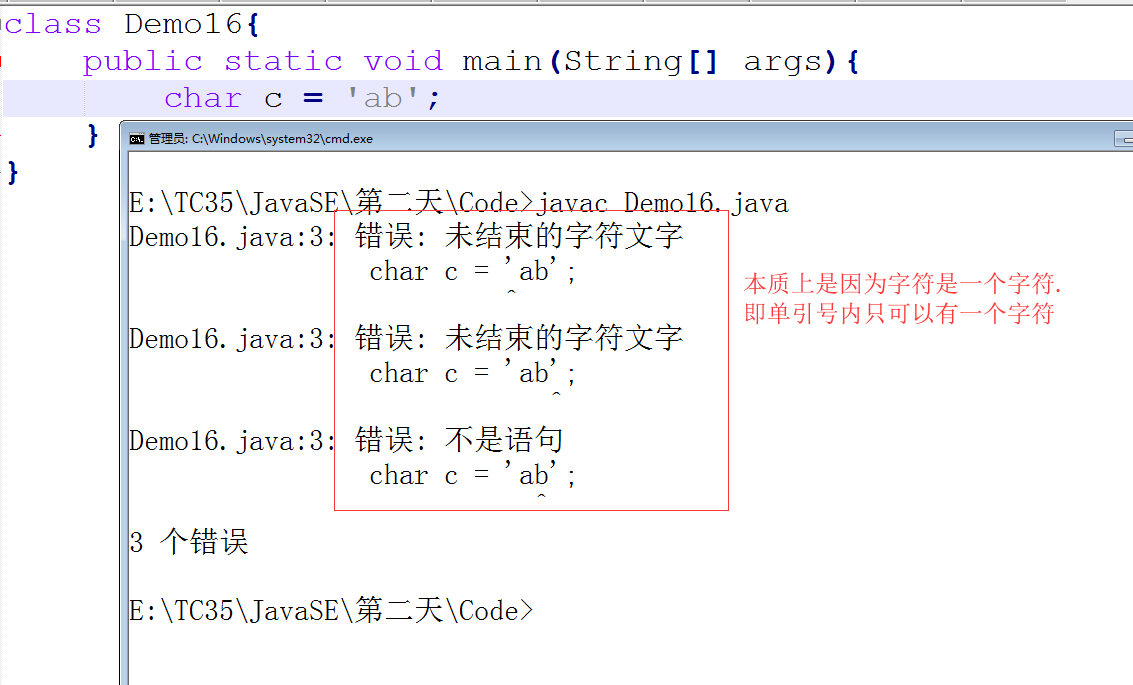






# 可能遇到的错误操作

* 使用char保存了两个字符



# 思考

* 我们所学过的运算符有哪些可以作为左值和右值的.

# 家庭作业

1. 完成下面的题目

参考家庭作业文本

1. 制作第一天和第二天的思维导图

# 预习

1. 条件结构
   1. If
   2. Switch
2. 关键字break在Swtich的使用
3. Java中获取随机的两种方式
   1. Math.random
   2. Random

[复习作业 1](#_Toc526954656)

[关键字 1](#_Toc526954657)

[保留字 1](#_Toc526954658)

[Break关键字 1](#_Toc526954659)

[随机数 2](#_Toc526954660)

[条件结构 2](#_Toc526954661)

[条件结构的概念 3](#_Toc526954662)

[IF结构 4](#_Toc526954663)

[单项If 4](#_Toc526954664)

[双项IF 5](#_Toc526954665)

[嵌套IF 5](#_Toc526954666)

[Swtich结构 7](#_Toc526954667)

[参考资源 8](#_Toc526954668)

# 复习作业

# 关键字

关键字:KeyWord是预先已经定义好的具有特殊含义的字符序列.

## 保留字

## Break关键字

Break的作用只有一个离开当前代码块(即大括号.类除外).

# 随机数

所有语言中都会有随机数功能.这些随机数都是伪随机数.本质是取当前时间的毫秒数.按照某个算法进行计算求得数字就是所获取的随机数.

Java中提供了2种获取随机数的方式

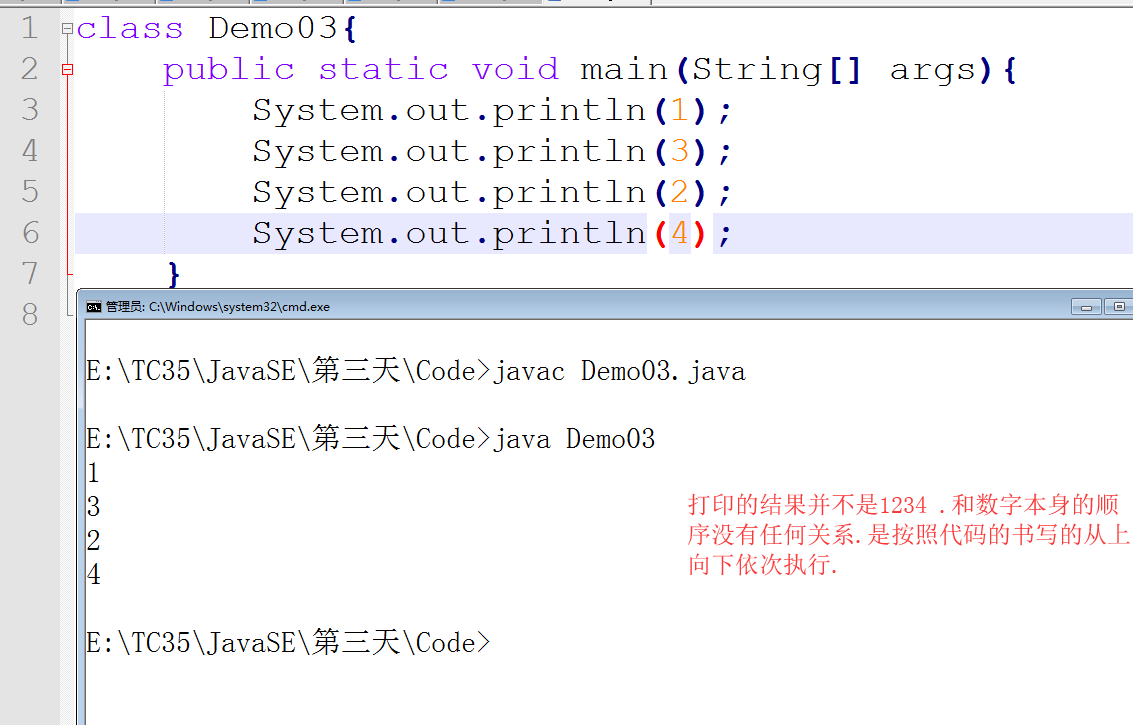
* 使用Math下的random()方法
* 使用Random工具类



# 条件结构

任何一门语言,默认都会有三种执行流程

* 顺序结构(默认的流程)



* 条件结构
* 迭代结构(循环结构)

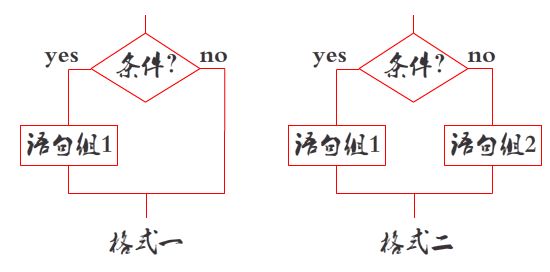
## 条件结构的概念

日常生活中并不是一直沿着直线走的.比如会遇到红灯判断走还是停,会遇到十字路口判断是走转还是右转还是直行.像这样需要进行判断,再根据判断结果进行选择的就叫做条件结构.

Java中的条件结构有两种结构

* IF结构
* Switch结构

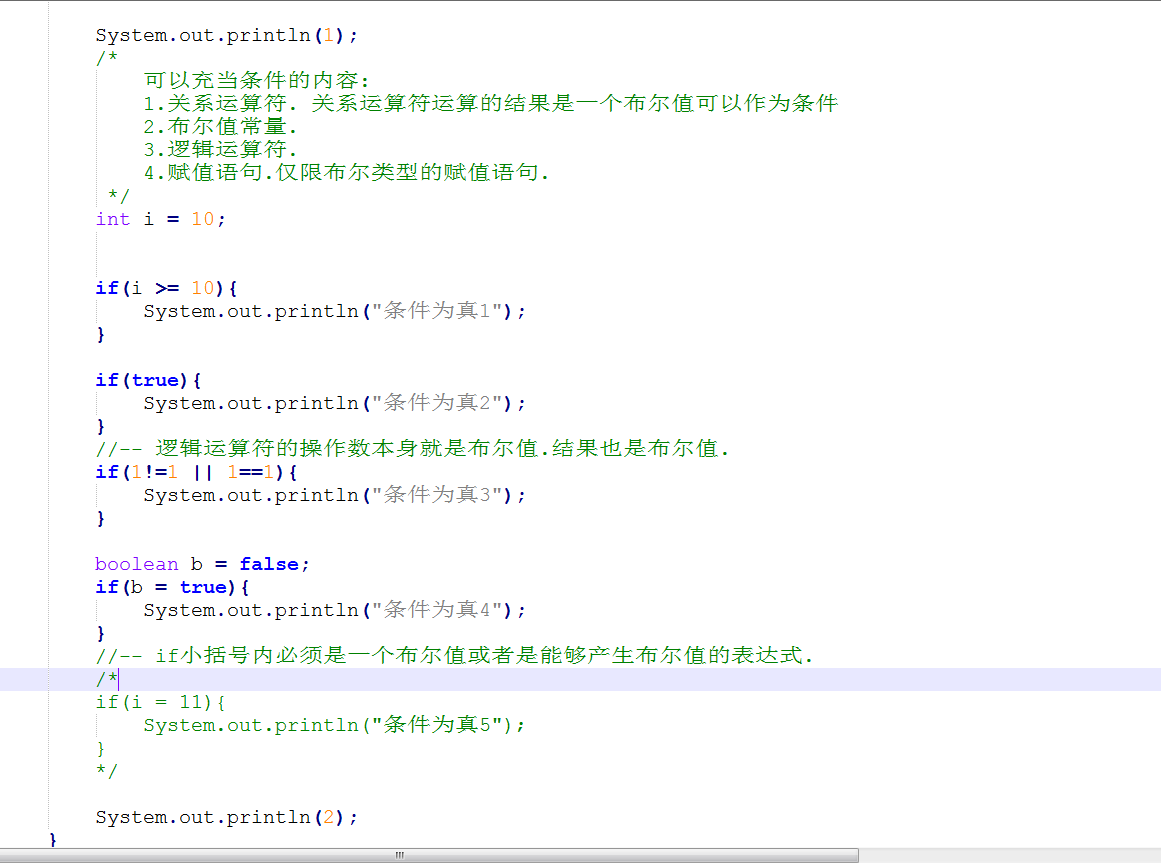
### IF结构



我们按照IF的写法可以把IF语句分成三种写法

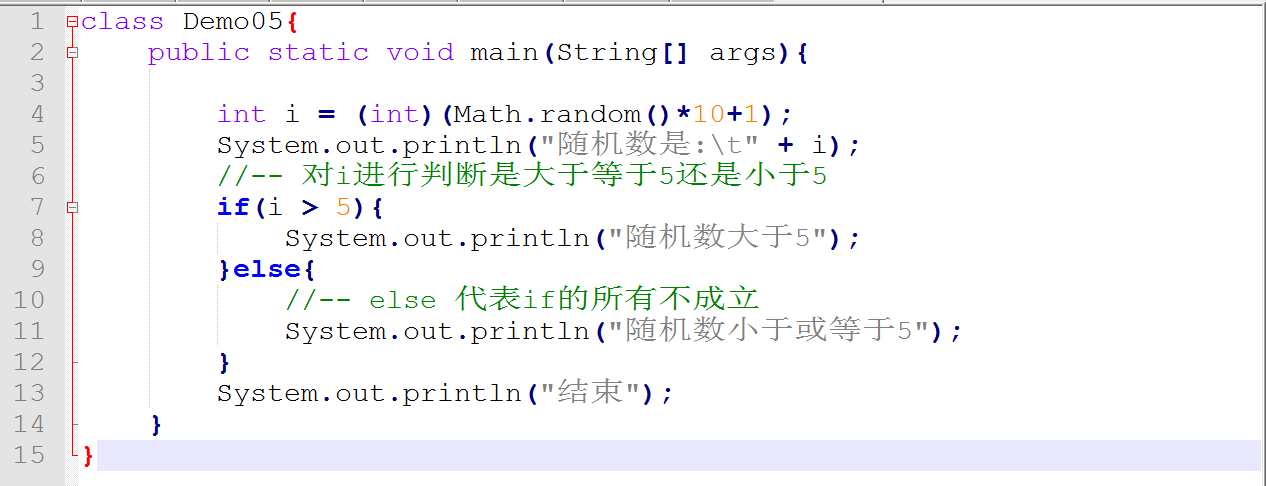
#### 单项If

只有一个IF的语句.类似于我们日常生活中所说的如果-就语句.



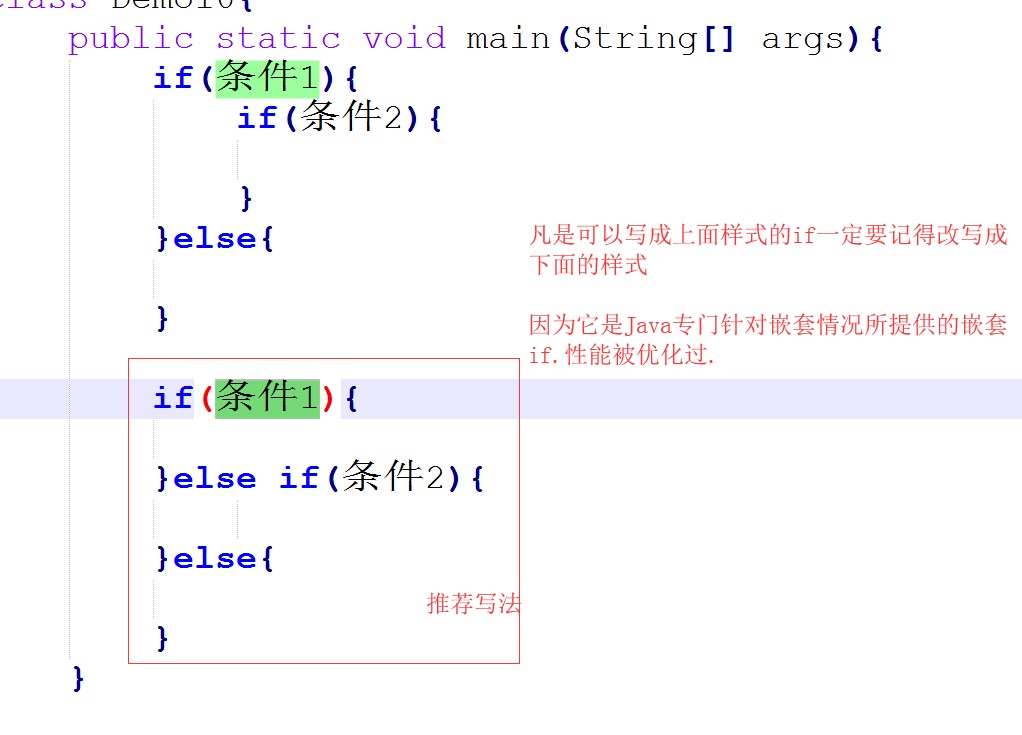
#### 双项IF

双项IF是带有IF反面的语句.即包含条件不成立的情况该怎么办的语句.



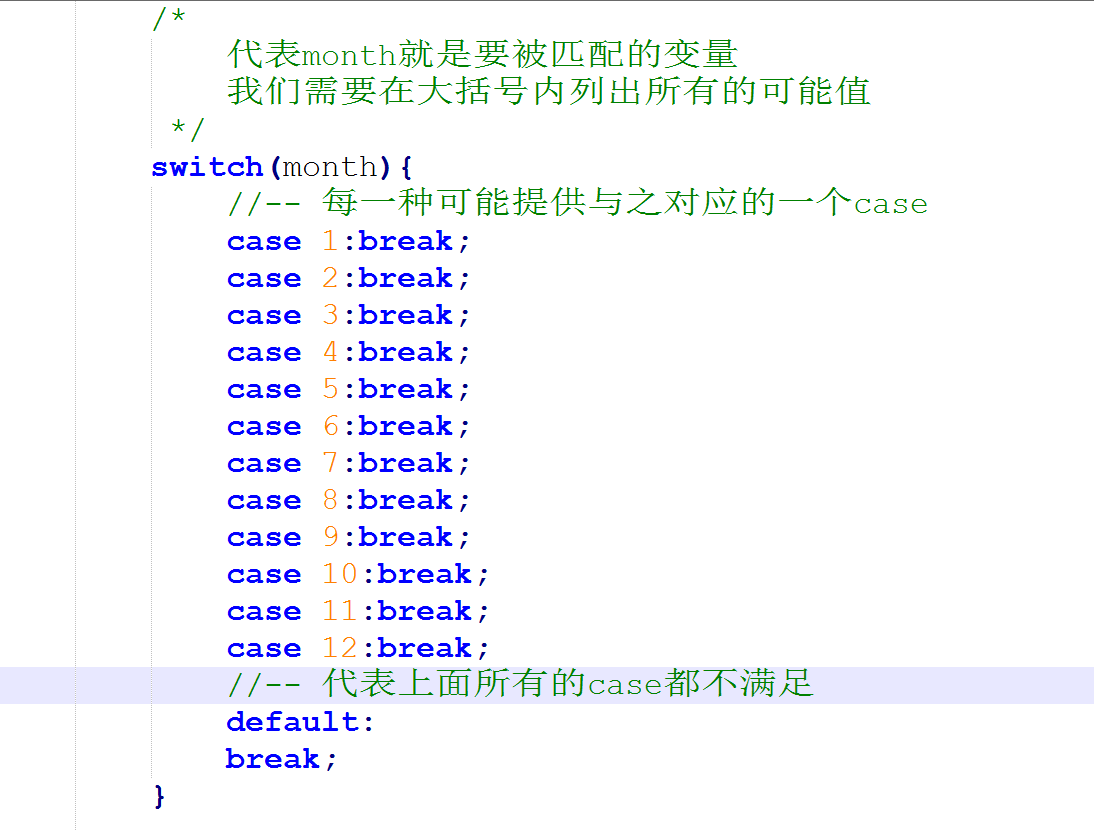
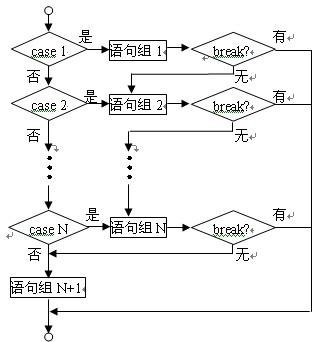
#### 嵌套IF

就是在IF里面在写个IF



### Swtich结构

Switch也是条件结构的一种.但是它一般用于等值匹配.



#### Switch能够接受的类型

IF的小括号中接受的是布尔值(能够产生布尔值的所有表达式都可以).但是switch不是.switch可以接收的是整数(byte short int)或者字符串(String JDK1.7以上版本.)

# 参考资源

* [谷歌Java代码规范](https://google.github.io/styleguide/javaguide.html)
* [谷歌Java代码规范中文版](https://www.cnblogs.com/lanxuezaipiao/p/3534447.html)

# 家庭作业

1. ~~完成家庭作业World文档~~ 10/10，22：29
2. ~~完成思维导图~~10/10，22：47
3. ~~测试switch能够接受的各种数据类型(把8种原生各测一遍.记录报的错误)~~ 10/10，23：48
4. 计算个人所得税

# 预习

1. 循环
   1. For(预习)
   2. While(周五的内容)
   3. Do-while(周五的内容)

# 看API文档

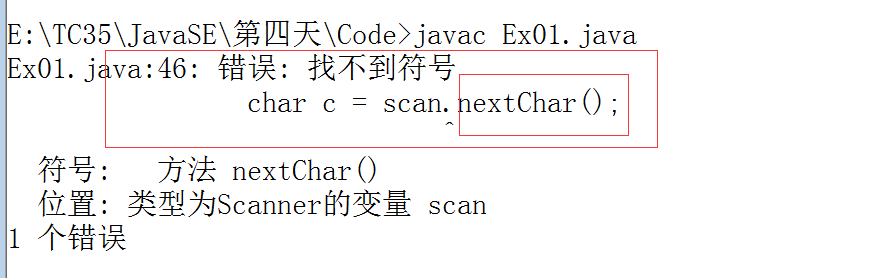
1. 测试8种原生数据类型的键盘录入

API文档，Class Scanner，没有char类型，不支持

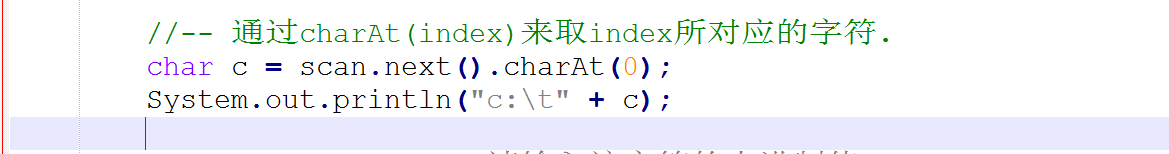
# Scanner的功能介绍

用于从指定的输入中获取数据.这些指定的输入可以是文件.可以是标准输入(从键盘录入).

它针对原生数据类型的8种提供了7种方法.唯一不支持的就是char类型的录入



如果需要获取从键盘录入的字符可以使用charAt()函数



# 循环结构

循环在日常生活中很常见.比如日复一日,年复一年描述的就是循环.

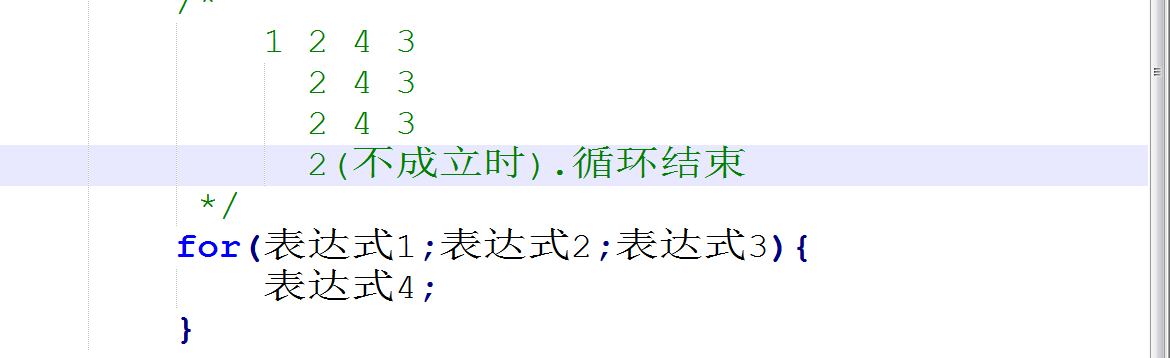
简单说循环就是重复在做同一件事情.Java中提供了三种基本循环结构.这三种循环结构也通用语其它语言.

* For
* While
* Do-While

# For

For是所有循环中最基本的一套写法,并且它的循环思路很明确.因为它拥有完整的循环4个表达式

For的格式如下



这四个表达式的功能分别是

* 表达式1:用于声明一个变量来记录循环了多少次
* 表达式2:是一个条件结构,用于控制循环是否继续
* 表达式3:递增或递减语句.来改变循环变量.起到改变循环条件的目的.
* 表达式4:需要重复执行的内容

## While

# 关键字

## Break

跳出循环

## Continue

跳过当前循环进入下一次循环.

# 循环要求的掌握程度

1. 熟练使用单层循环
2. 熟练使用循环嵌套条件
3. 熟练使用循环嵌套循环

# 思考

* 死循环(永远不会结束的循环)是怎么产生的.
* 怎么解决
* 如果循环不是按照想象那样去执行的,可能的原因有几个
* 循环的执行顺序(必须熟知)

# 作业

1. ~~家庭作业Word中的第一题和第二题.~~10-11 20:30
2. ~~课堂练习Word中的第十六题~~和第二十二题.10-11 11:51

# 预习

* While
* Do-While

# 开发流程

1. 产品经理提出需求
   1. 甲方需求(为自己开发)
   2. 乙方需求(外包到别人的公司为别人开发)
2. 交给UI
   1. 制作静态页面.(简单说就是产品的效果图,是图片)
3. 交给前端
   1. 负责把UI制作的图片转成实际的静态页面.因为我们做的是Web那就是HTML页面.
4. 交给后端
   1. 开始写后端的业务逻辑和数据库的操作
   2. 写测试文档
      1. 给测试人员看的
5. 交给测试
   1. 测试没有问题
6. 交个上线人员上线
   1. 凌晨上线
7. 教给客户或者交给运维

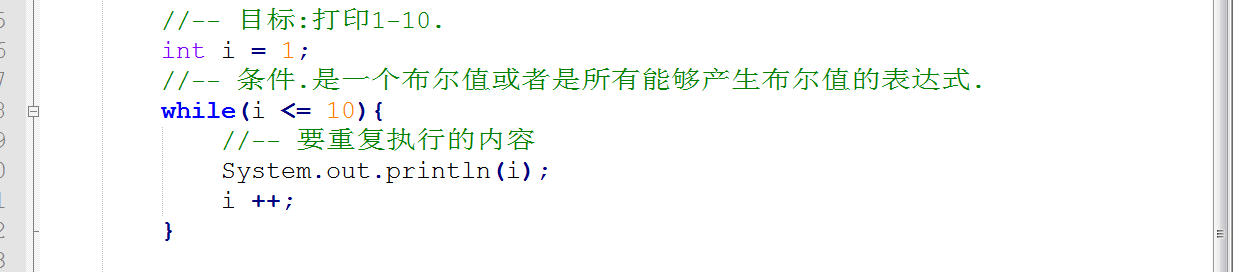
# 其它循环介绍

## While循环

While的写法和If的写法有些类似.都需要在小括号内提供条件.

### 可以构成条件的运算符

1. 关系运算符
2. 逻辑运算符
   1. 逻辑运算符存在短路现象
      1. 但部分表达式可以确定整体的真假时,后续的不运算.
      2. true&&false||true 和true&&false&&true
3. 针对布尔值的赋值运算
4. 直接提供布尔值常量.



## Do-While循环

针对一种特殊情况而创建的循环:至少要执行一次的.因为Do-While 是先执行再判断



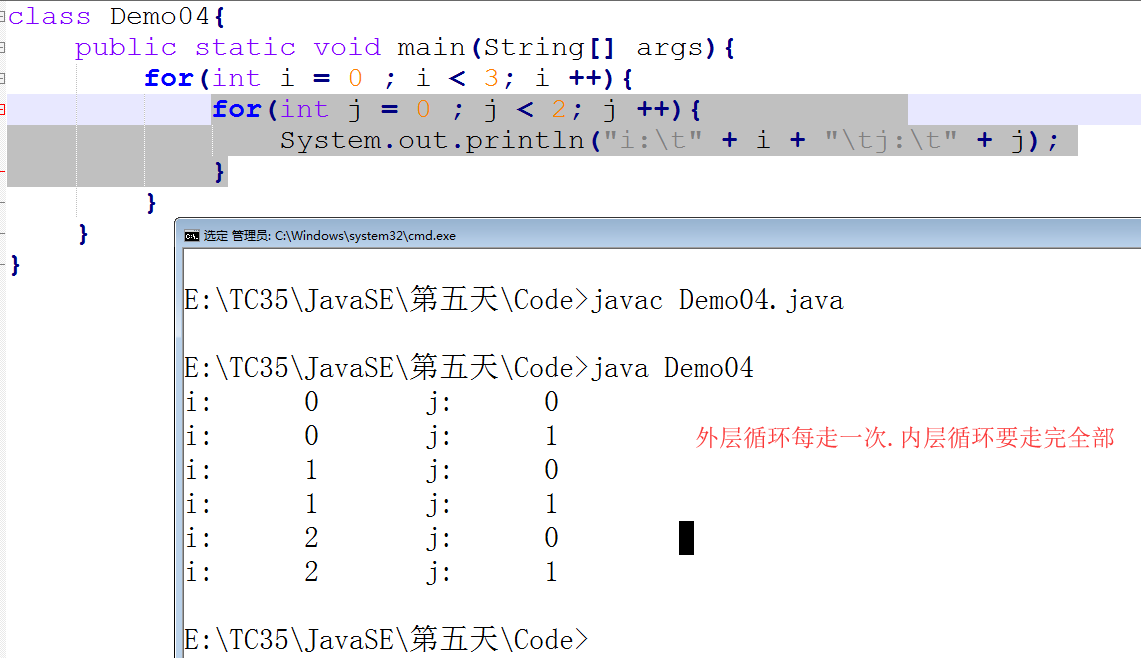
## 三种循环对比

|  |  |
| --- | --- |
| 循环 | 特点 |
| For | 结构工整(四个表达式),擅长在直到循环次数的情况下执行. |
| While | 擅长在不知道循环次数的情况下执行 |
| Do-While | 至少会执行一次. |

# 循环和条件的组合使用

## 循环嵌套循环

循环嵌套循环.就是嵌套循环.需要注意的是(假设嵌套一层).有外层循环A和内层循环B.B是作为A的循环体存在的.当B执行完成后才会执行下一次循环.



我们可以为内外两层循环(A和B)四个表达式分别命名A1,A2,A3,A4和B1,B2,B3,B4.那它们的执行流程(假设A循环3次.B循环2次)如下

* A第一次
  + A1 A2 A4(直接转到B)
    - B1 B2 B4 B3
    - B2 B4 B3
  + A3
* A 第二次
  + A2(判断条件是否成立.成立继续.不成立则直接退出循环) A4(直接转到B)
    - B1 B2 B4 B3
    - B2 B4 B3
  + A3
* A 第三次
  + A2(判断条件是否成立) A4
    - B1 B2 B4 B3
    - B2 B4 B3
  + A3

### 嵌套循环的循环跳出

因为是嵌套循环就必须要清除的知道跳出的是那一层循环.并且还要清除到底是跳出还是跳过.

比如循环1-7.代表周一到周七.一个月假设有31天.问那些天是放假的.要求代码实现..

我们一般使用循环条件或者直接使用条件表达式或者是布尔值来在内层循环进行控制是否跳出或者跳过.

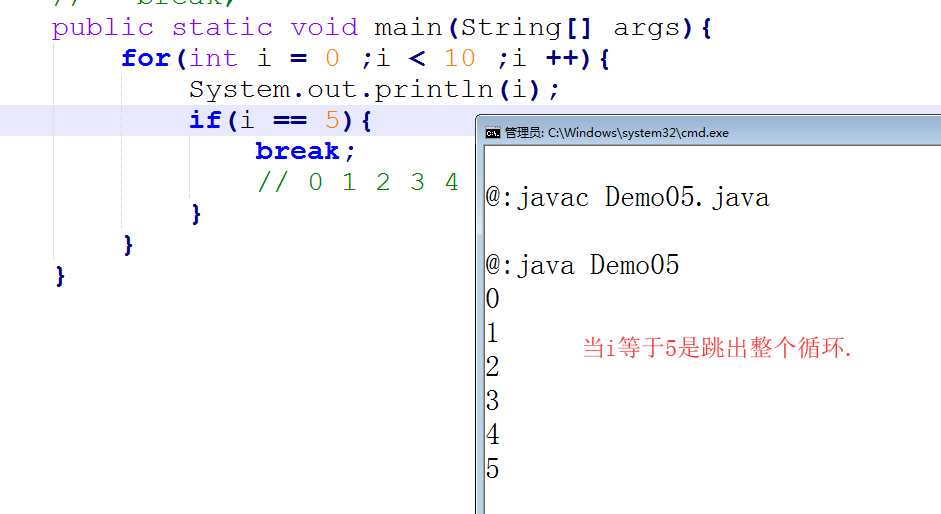
(等到二维数组的时候会再说!)

## 循环嵌套条件

# 关键字

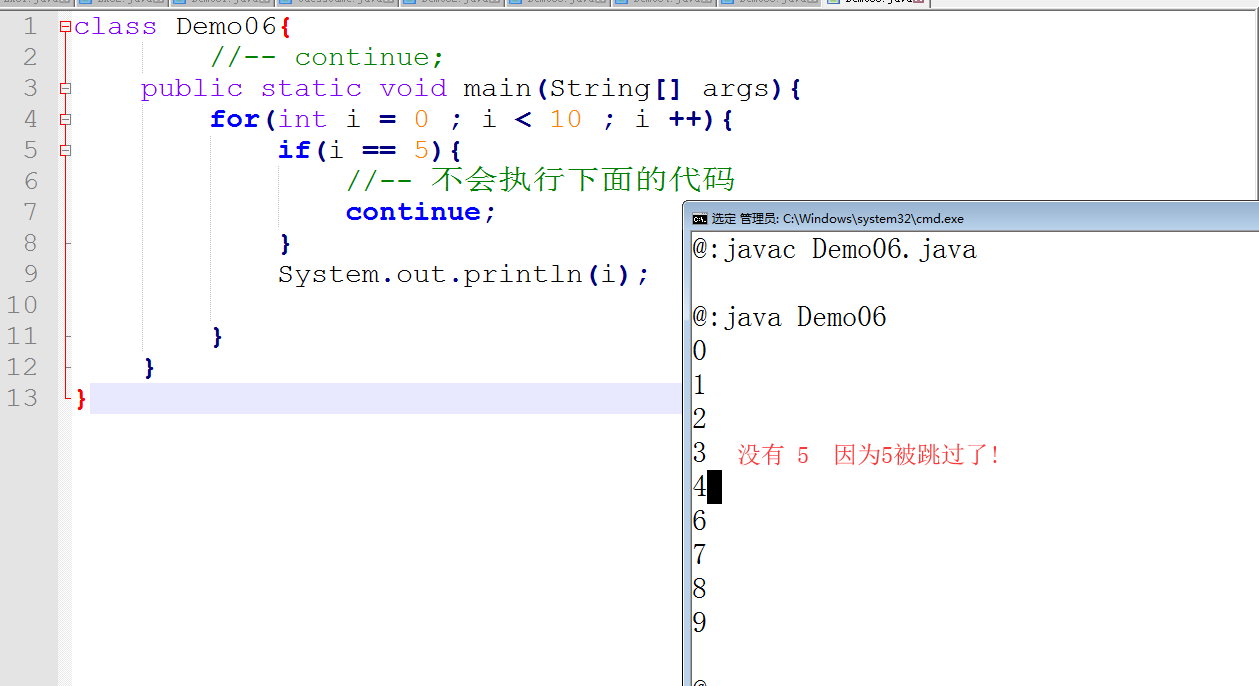
## Break

脱离距离Break最近的一层循环.

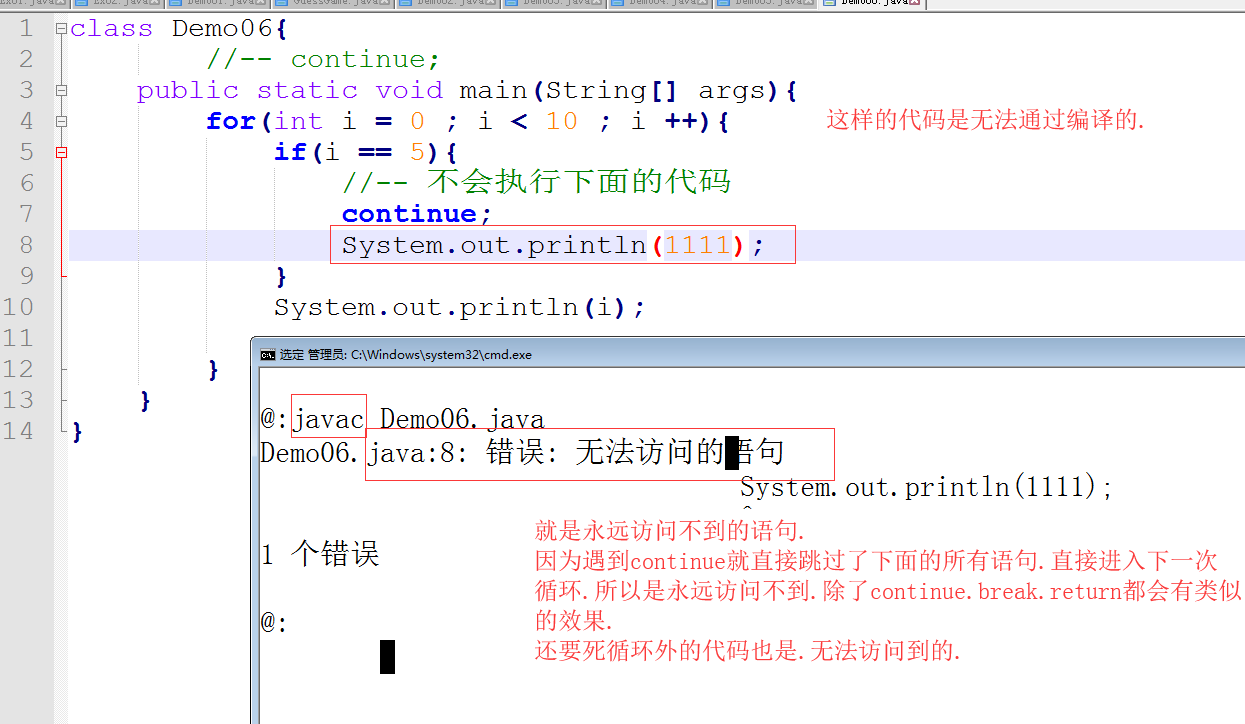


## Continue

跳过当前进入下一次



UnReachable Statement

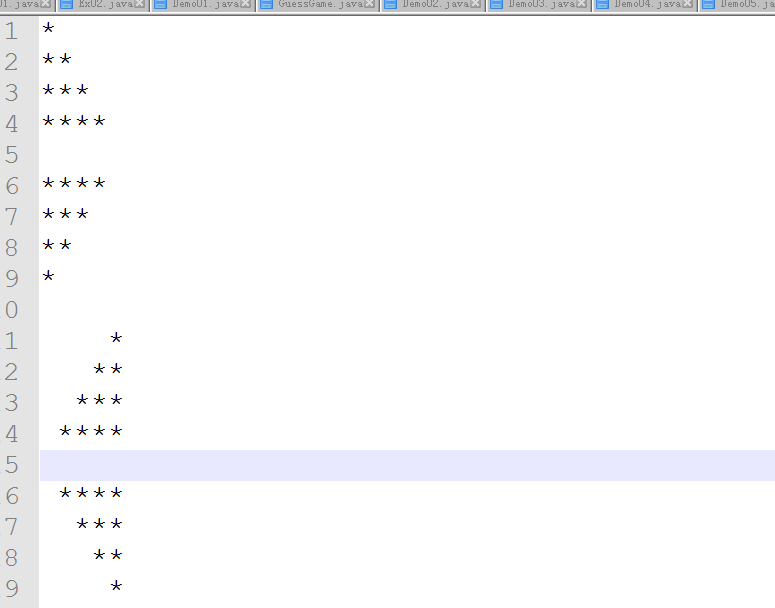


# 预习

1. 数组的三种声明方式
   1. 注意区别
2. 数组的取值和赋值(非常重要)
3. 排序算法
   1. 冒泡排序
   2. 快速排序
4. 查找算法
   1. 二分查找法
5. 二维数组的创建
6. 二维数组的取值和赋值

# 家庭作业

1. 打印四个三角形，30分钟（笨方法），10.13



三角形的行数是由键盘录入的.

1. 使用For和While两种方式实现九九乘法表
   1. 练习双重循环
2. 使用While和do-while来重写猜拳
3. 使用For和While来重写万年历
   1. 1900年一月一日是星期一
   2. 求从1900年1月1日到今天的所有天数和7取余数.
   3. 前面补空格
   4. 一行是七天.第一行就是数字+空格一共为7
4. 归纳客户关系管理系统中的模块和模块下的功能.
5. 看API文档.
   1. Lang(java.lang)包下有哪些原码文件(知道).

[数组 1](#_Toc527386007)

[数组的概念 1](#_Toc527386008)

[数组的声明 2](#_Toc527386009)

[通用的三种声明格式 3](#_Toc527386010)

[原生类型的数组声明 3](#_Toc527386011)

[引用类型的数组声明 3](#_Toc527386012)

[数组的取值和赋值 3](#_Toc527386013)

[数组中元素的遍历 4](#_Toc527386014)

[排序算法 4](#_Toc527386015)

[冒泡排序 4](#_Toc527386016)

[二维数组 5](#_Toc527386017)

[课堂练习 5](#_Toc527386018)

[家庭作业: 6](#_Toc527386019)

[参考资源 6](#_Toc527386020)

# 数组

## 数组的概念

数组是一个容器.用于存储同类型的一组数据.容器一般都是有大小的.所以数组在声明时是必须要指明数组的大小.即数组中有多少个元素.(数组一旦声明成功,大小不可以改变).

## 数组的声明

在Java中有三种声明数组的方式.有各自的使用场景.

* 使用大括号声明(必须掌握)
  + 这种声明也叫做遍历初始化.只能用于声明新数组时使用.不可以给已有数组赋值.
* 指明长度声明(必须掌握)
* 指明初始值声明.(了解)

### 通用的三种声明格式

### 原生类型的数组声明

### 引用类型的数组声明

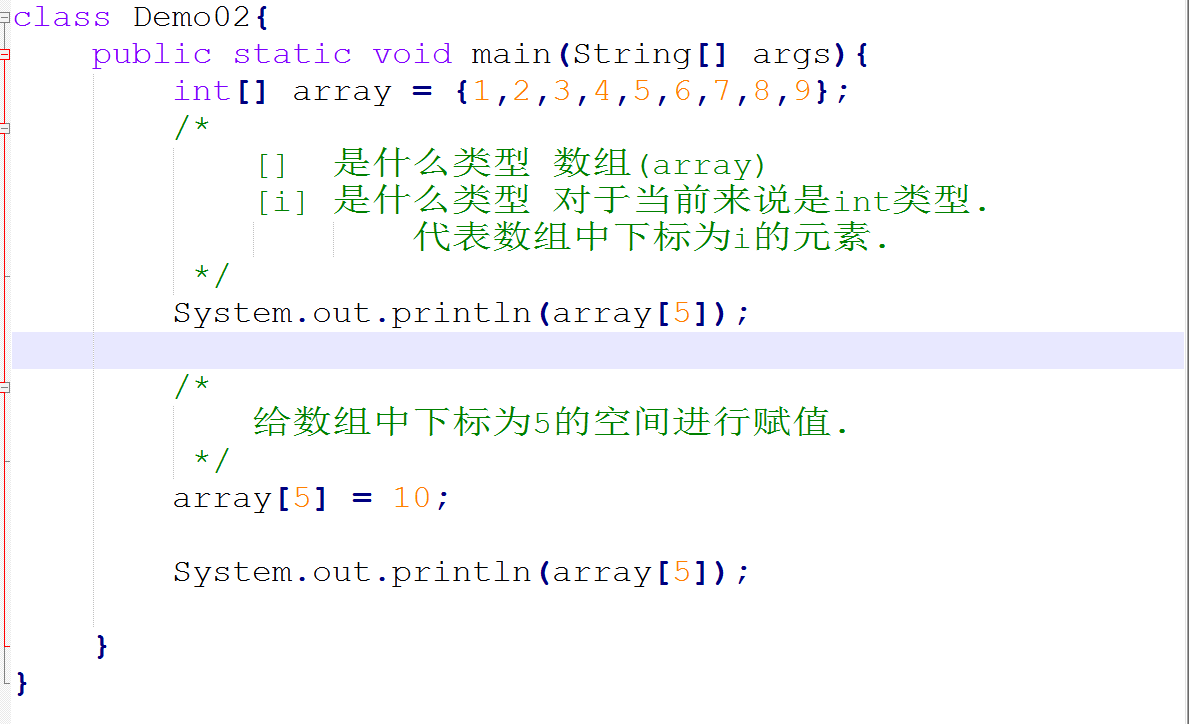


## 数组的取值和赋值

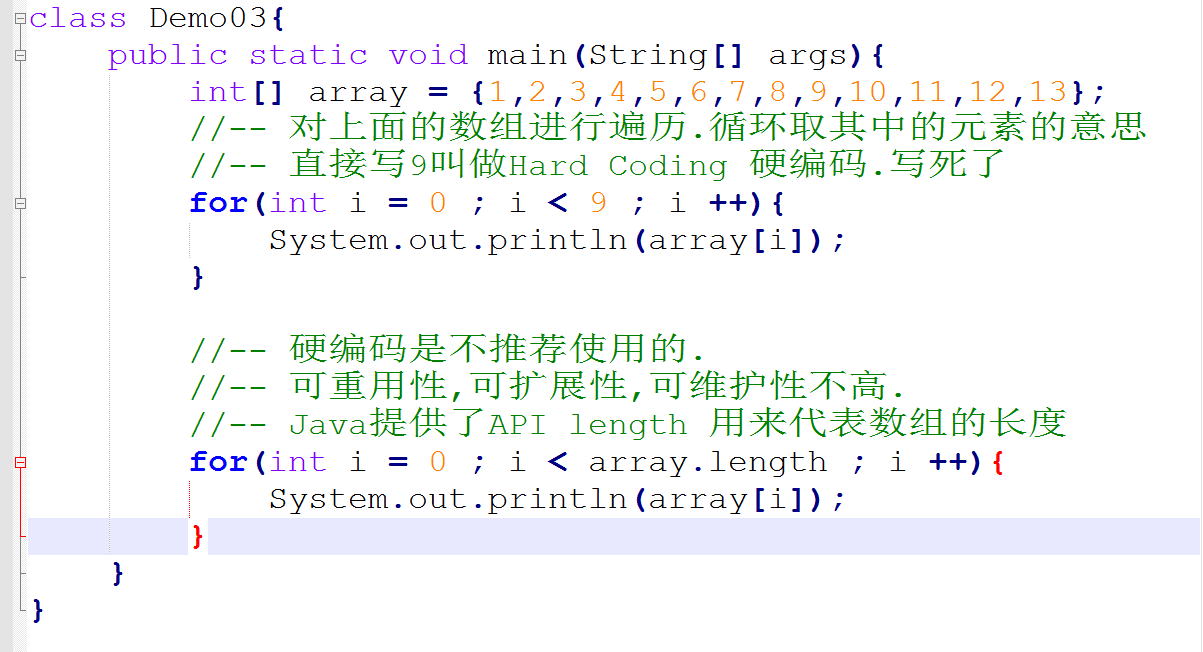
对数组中的元素进行操作是通过下标来完成的.基本上所有的语言中,凡是涉及下标的都是从0开始(极个别除外).

在Java中我们通过中括号配合下标来实现对数组中,下标所对应的元素进行读和写.

这里需要注意的一个细节就是,要区分清楚是子元素还是数组整体.



## 数组中元素的遍历



## 排序算法

### 冒泡排序

冒泡是最基本的排序算法,经常会在白板面试中出现.白板面试简单说就是手写代码.一张白纸一支笔没了.!

冒泡的核心思想我们可以总结为以下几点

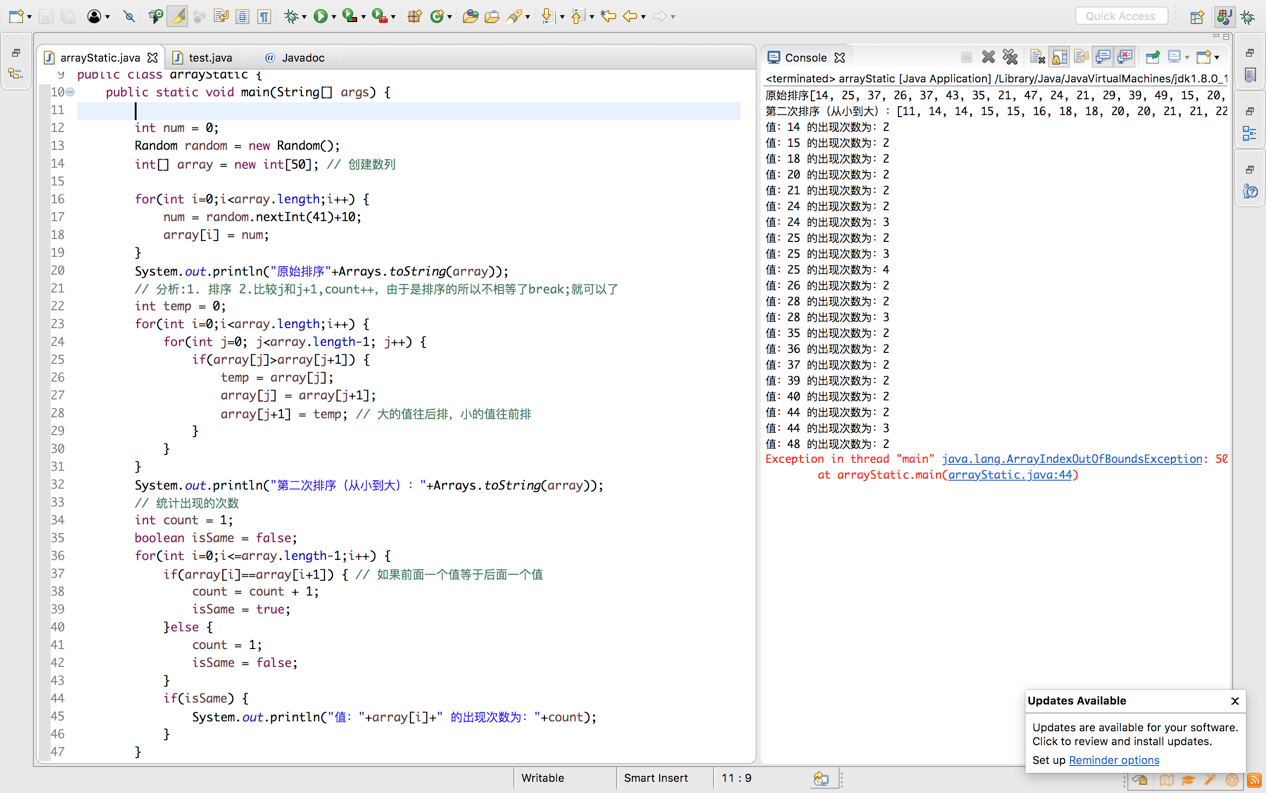
* 双循环
  + 内循环控制比较
* 相邻的两个数进行大小比较
* 每轮比较完成.会确定一个最大的
* 下一轮比较会少比较一个元素.

冒泡的写法可以总结为一句话:外层不变(或者-1),内层-1-i.两两交换.

## 二维数组

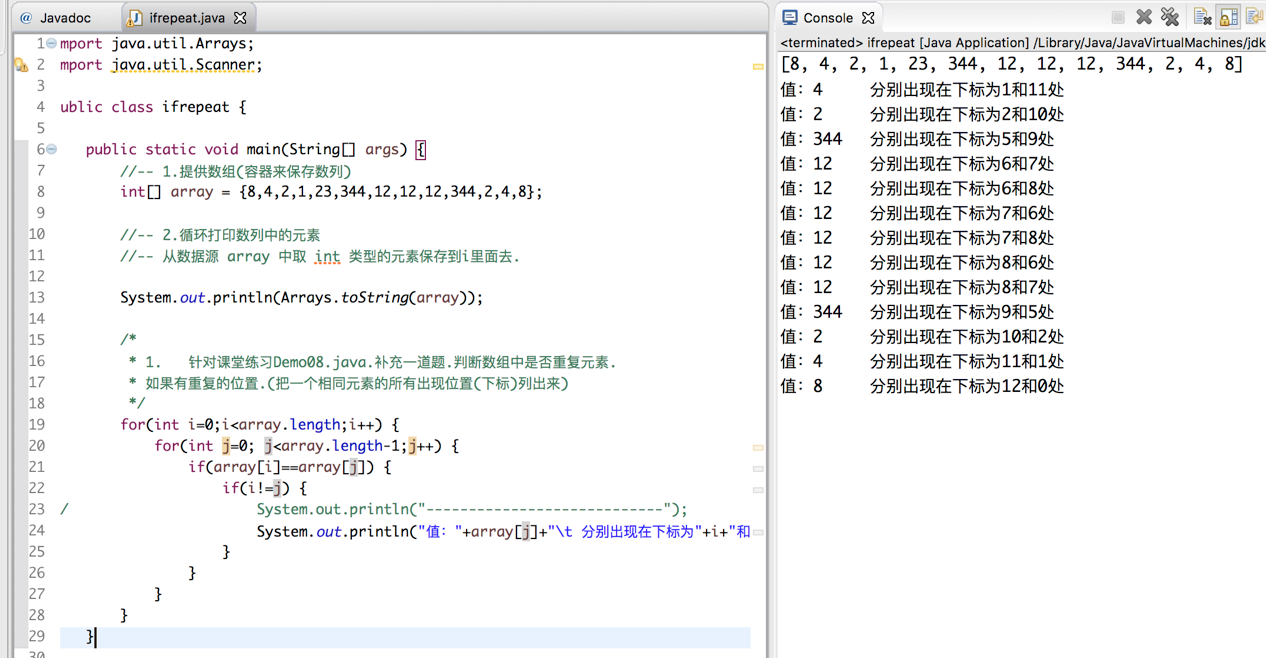
# 课堂练习

1. 原生数据类型有8小种.提供与之对应的数组.(长度都定为18)(创建方式为:方式一和二各四个.)
2. 接第一题.在使用第二种创建方式的数组上测试.在没有赋值的情况下.从数组中取值,看看这个时候数组中是否有值
3. 给int类型的数组赋值.取值随机数(随机数取值范围1-10)
4. 接第三题.对int类型数组进行遍历.
5. 接第三题.要求数组中的元素不可以重复
6. 有一个一维数组{7,3,4,9,3,2,6,10,16,12,4}，找出最大的值和最小的值并打印输出
7. 有一个数列：8，4，2，1，23，344，12.实现如下要求
   1. 循环输出所有的数列值
   2. 求数列中所有数字的和
   3. 从键盘录入一个数判断该数是否在数列从
   4. 如果数列中有重复元素.求所有重复的元素出现的位置(家庭作业第一题)
8. 随机产生50个10到50的整数，统计每个数字各出现几次，出现0次的数字不打印。

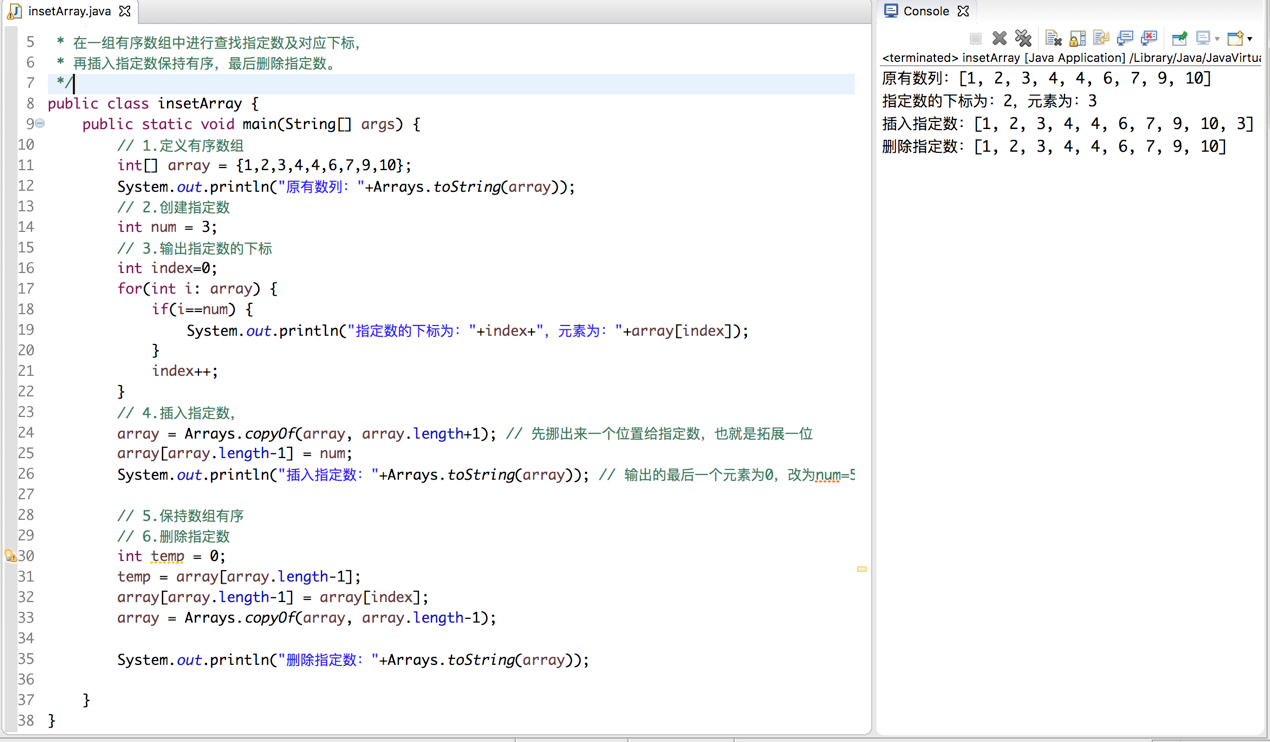


# 家庭作业:

1. 针对课堂练习Demo08.java.补充一道题.判断数组中是否重复元素.如果有重复的位置.(把一个相同元素的所有出现位置(下标)列出来)



1. 在一组有序数组中进行查找指定数及对应下标，再插入指定数保持有序，最后删除指定数。



# 参考资源

1. [排序的HTML动画](http://www.webhek.com/post/comparison-sort.html)

[二维数组 1](#_Toc527449646)

[方法 2](#_Toc527449647)

[方法的声明格式 2](#_Toc527449648)

[有参数方法 2](#_Toc527449649)

[无参数方法 2](#_Toc527449650)

[有返回值方法 2](#_Toc527449651)

[方法的调用 2](#_Toc527449652)

[方法的重载 2](#_Toc527449653)

[预习 2](#_Toc527449654)

[课堂练习 2](#_Toc527449655)

[家庭作业 2](#_Toc527449656)

[复习 2](#_Toc527449657)

# 二维数组

理论上Java中是没有多维和不规则数组的.我们一直都认为Java只有一维数组.但是如果遇到像这样的内容该怎么表达呢?

{{1,2,3},{1,2,3},{1,2,3}}

# 方法

## 方法的声明格式

## 有参数方法

## 无参数方法

## 有返回值方法

## 方法的调用

## 方法的重载

# 预习

# 课堂练习

1. 设计一个容器保存100期彩票数据.(以双色球为例)双色球一共有7个数.6个红球,1个蓝球.红球取值1-33.蓝球取值1-16.红球是不可可以有重复的.蓝球和红球的号码允许一样.
   1. 统计每个号码在100期里面出现的次数
   2. 统计每个号码出现的概率
   3. 以出现的概率排序
   4. 所有的操作都是区分红球和蓝球的.

# 家庭作业

# 复习

1. 万年历
2. 打印三角形
3. 打印99乘法表
4. 针对数组进行增删改查
5. 对数组中重复元素统计次数

[方法 2](#_Toc527557785)

[方法的声明 2](#_Toc527557786)

[方法的组成部分 2](#_Toc527557787)

[方法签名 2](#_Toc527557788)

[什么时候使用方法 2](#_Toc527557789)

[方法参数的含义 3](#_Toc527557790)

[方法返回值的含义 3](#_Toc527557791)

[特殊的返回值类型:void 4](#_Toc527557792)

[无参方法声明 5](#_Toc527557793)

[有参方法声明 5](#_Toc527557794)

[有返回值方法声明 5](#_Toc527557795)

[方法的调用 5](#_Toc527557796)

[方法的重载 5](#_Toc527557797)

[标识符 7](#_Toc527557798)

[课堂练习 8](#_Toc527557799)

[复习 8](#_Toc527557800)

[家庭作业 8](#_Toc527557801)

# 方法

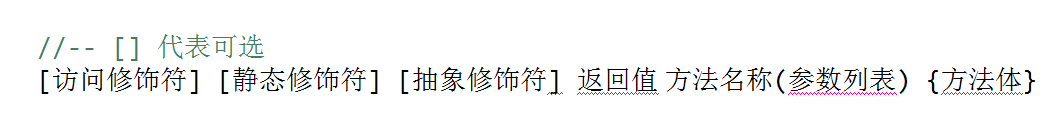
## 方法的声明

方法我们一直在使用:main.方法简单说就是一个独立的代码块.

不同代码块之间可以互相调用.这就是方法间的调用.

### 方法的组成部分

方法的组成部分可以有一下几块(对比main方法进行总结)



### 方法签名(必须掌握)

方法有方法签名.它代表方法的唯一性.在一个class中不可以出现重复的方法.和不可以出现同名变量一样.

方法签名的组成有两部分组成

* 方法名称
* 参数列表

### 什么时候使用方法

当我们发现一段代码多次出现时,可以考虑把这段代码独立出来,形成一个单独的模块.这个模块就是方法.

### 方法参数的含义(必须掌握)

参数相当于一台机器设备的进料口一样.比如榨汁机.它会有进料口,这个口是有口径的.这个口径限定了可以放多大的水果进行.

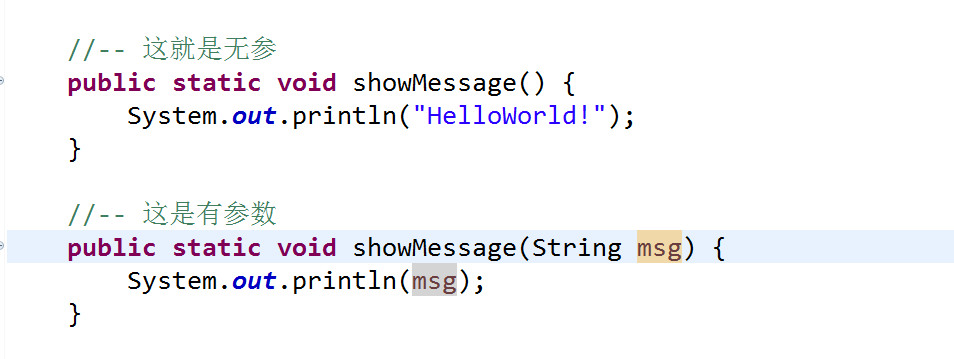
这个进料口就相当于我们方法中的参数.它有两个作用

* 规定可以接收什么类型的数据.
* 规定可以接收多少个数据.

当方法的执行所需要的数据无法内部生成时,必须由外部传入.参数就规定了,传入的数据类型和传入的数据量.

根据参数的有无我们把方法可以分为

* 有参方法
* 无参方法.



### 方法返回值的含义(必须掌握)

返回值是我们的一般叫法.准确说应该是返回值的数据类型.即当整个方法执行完成后.返回给调用方法的地方什么样的数据.

方法的返回值可以理解为.我们调用方法时,希望该方法执行结束后.能够给我们提供一个数据.

我们所需要的数据是什么类型.就把返回值类型设置成与之对应的.

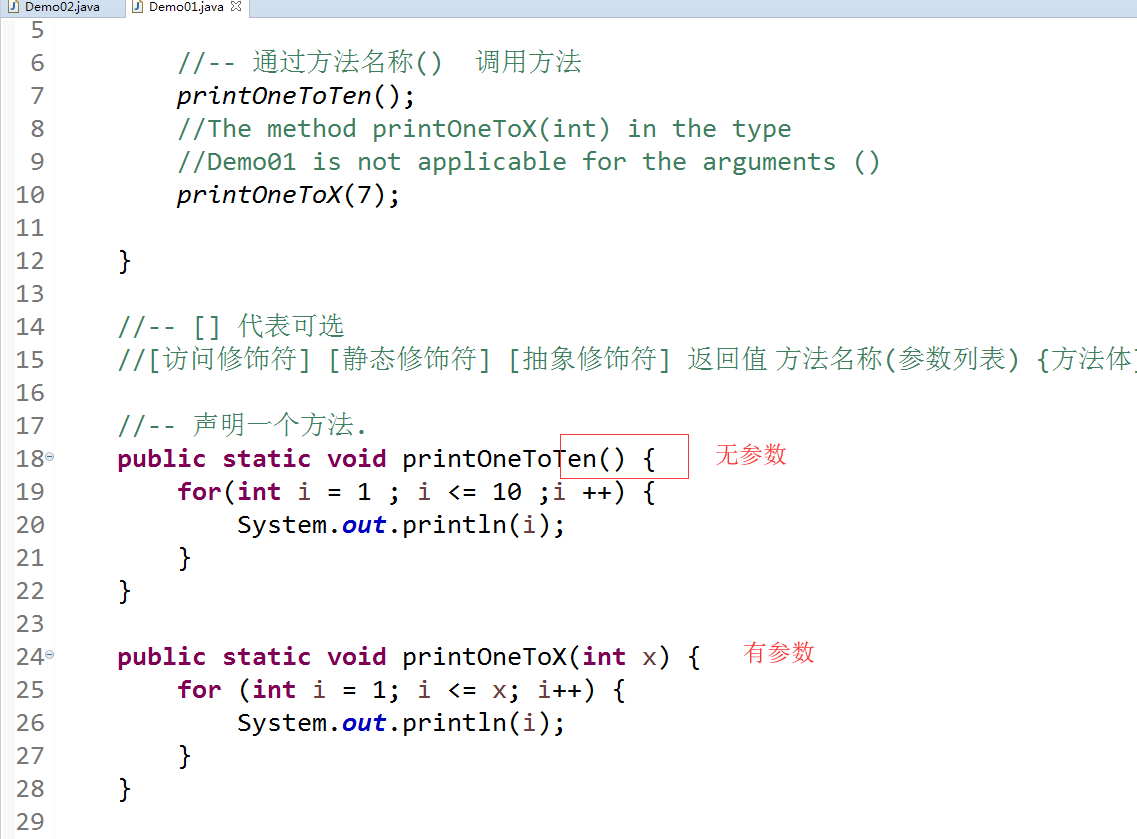
一旦有返回值类型(不是void),就必须使用关键字:return 来进行返回数据.

#### 特殊的返回值类型:void

Void代表的含义是空.即在方法结束后不需要返回值.

### 无参方法声明

### 有参方法声明



### 有返回值方法声明

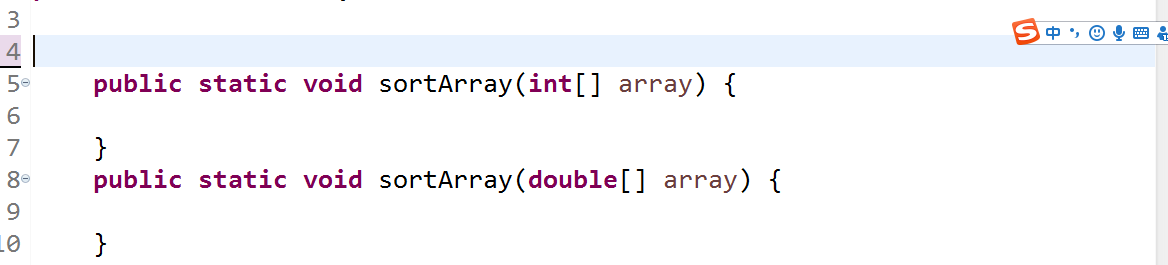
## 方法的调用(必须掌握)

通过方法签名来完成方法的调用.

## 方法的重载(必须掌握)

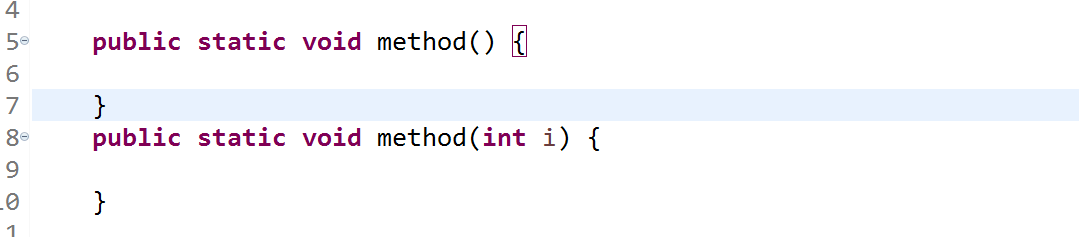
在一个class中是不可以出现同名方法的.即方法签名一样的方法出现两个.

但是可以出现同名不同参数的方法.这种写法被叫做方法重载

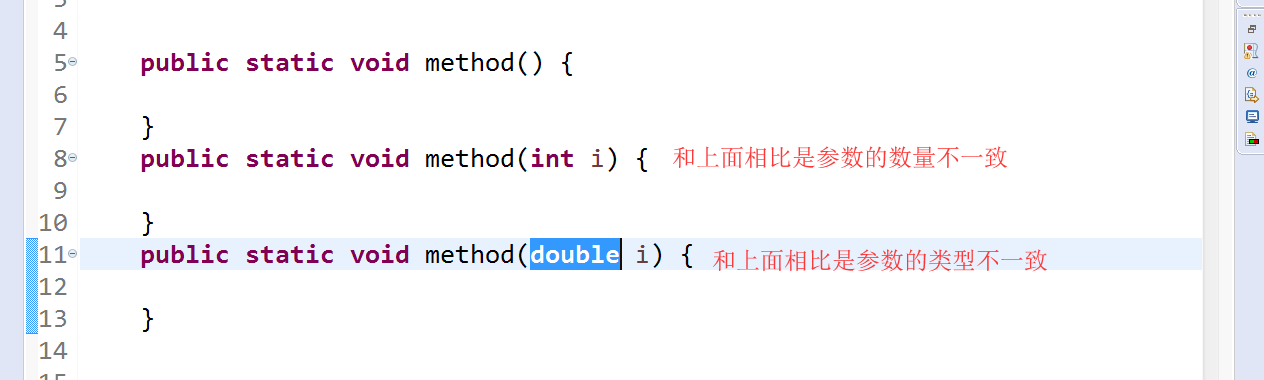


重载可以体现在一下几点上

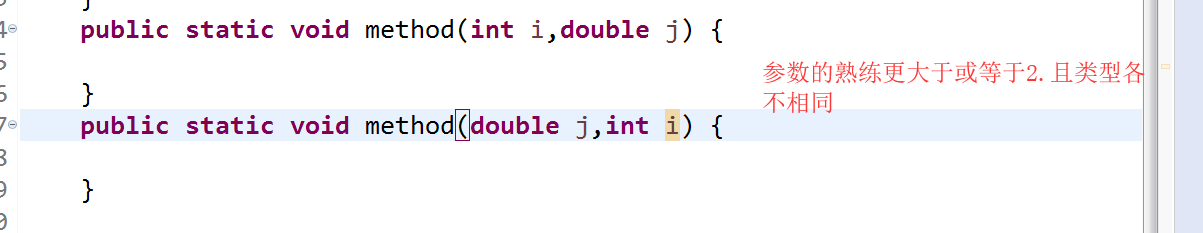
* 参数的数量不一致



* 参数的类型不一致



* 当参数数量大于等于2时,且类型各不相同时,顺序不一样也可以视为重载



平时在代码中最常见的方法重载是:构造方法的重载.

# 标识符

代码中所有用于命名使用的(给变量,给方法,给类等)有序的字符序列.Java是区分大小写的.

标识符的成员有:

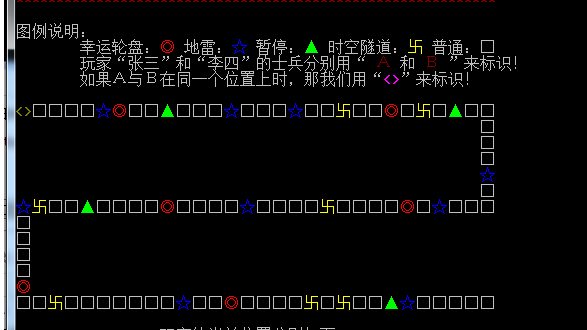
* A-Z
* a-z
* 0-9
* \_和$

在命名的时候我们需要遵守一些规范

* 只可以是字母开头,不可以是数字开头(数字开头是语法错误)
* 不推荐使用特殊符号(\_和$)开头
* 不可以使用已有的关键字来作为标识符
* 不可以使用系统已有的类名来作为标识符
* 在命名的时候要遵守驼峰式
  + 大驼峰:给类命名
    - 首字母大写
      * Student(凡是看到首字母大写的基本都是类名)
  + 小驼峰:给变量和函数命名
    - 首字母小写.如果有多个单词组成首字母还是小写后续单词首字母要大写
      * studentAge
      * studentPhoneNumber
  + 常量是全大写的.并且常量有多个单词组成,不同单词间用下划线间隔.

# 课堂练习

1. 定义一个方法.该方法可以打印1-10.(无参)
2. 定义一个方法.该方法可以给任意整型数组进行排序.(有参)
3. 设计一个方法.该方法根据数组中字符串的长度进行降序排列.（15分钟）
4. 菲波那切数列使用方法来实现.（10分钟）
5. 设计一个方法.用于把小写的字母转换成大写.
6. 绘制一张地图,地图效果如右边.(最低要求用星号打印2.家庭作业)



# 复习

# 家庭作业