[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

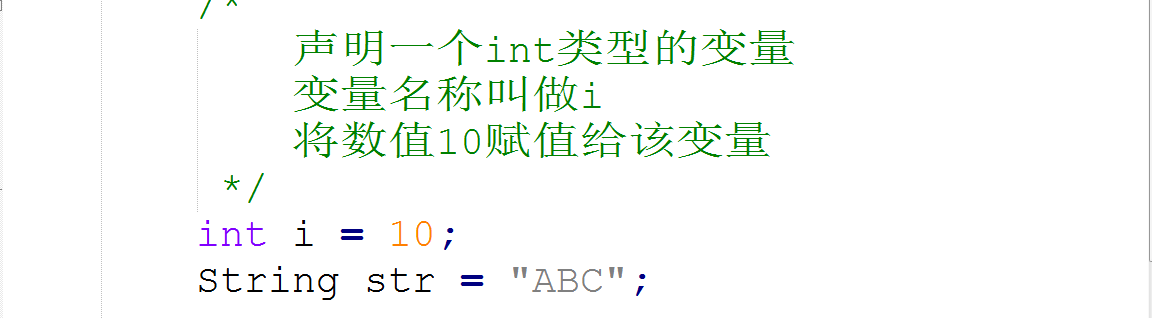
数据类型先教大家两种:

* 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.

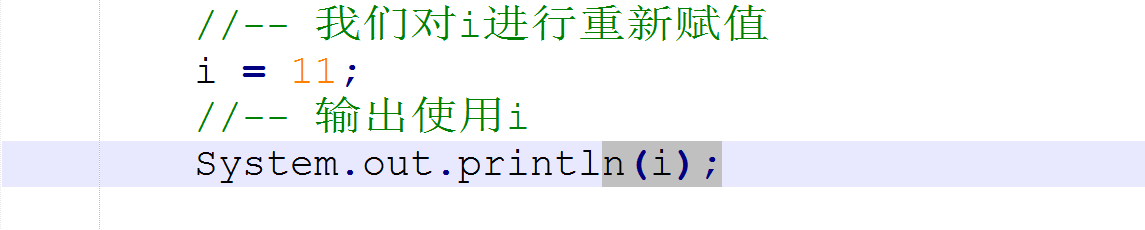
## 变量的声明



## 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

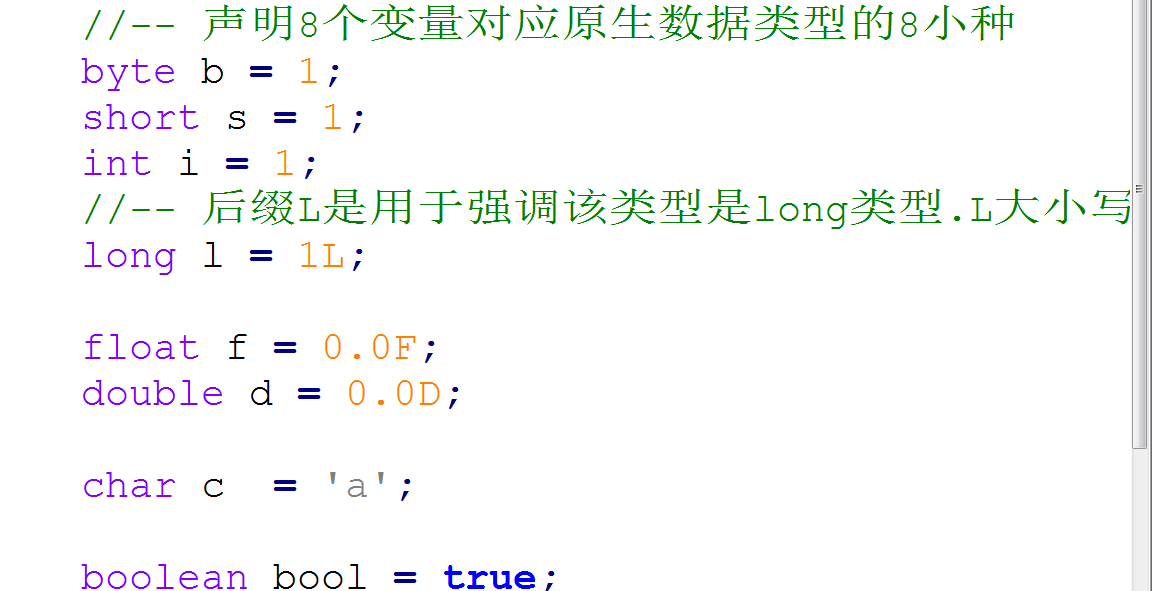
在Java中有两大类数据类型:

* 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 |  |



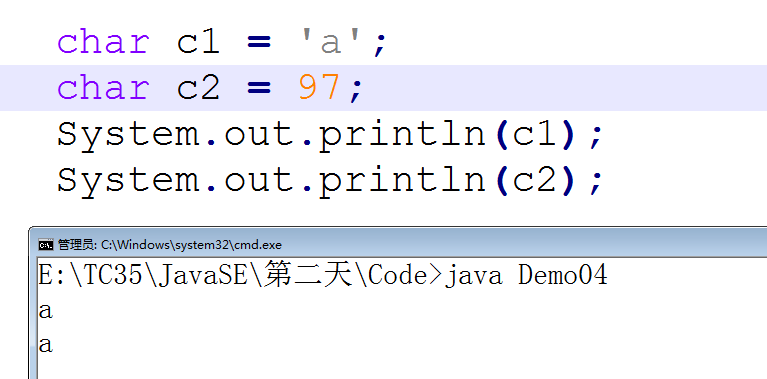
## 不同分类的默认类型

|  |  |
| --- | --- |
| 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.

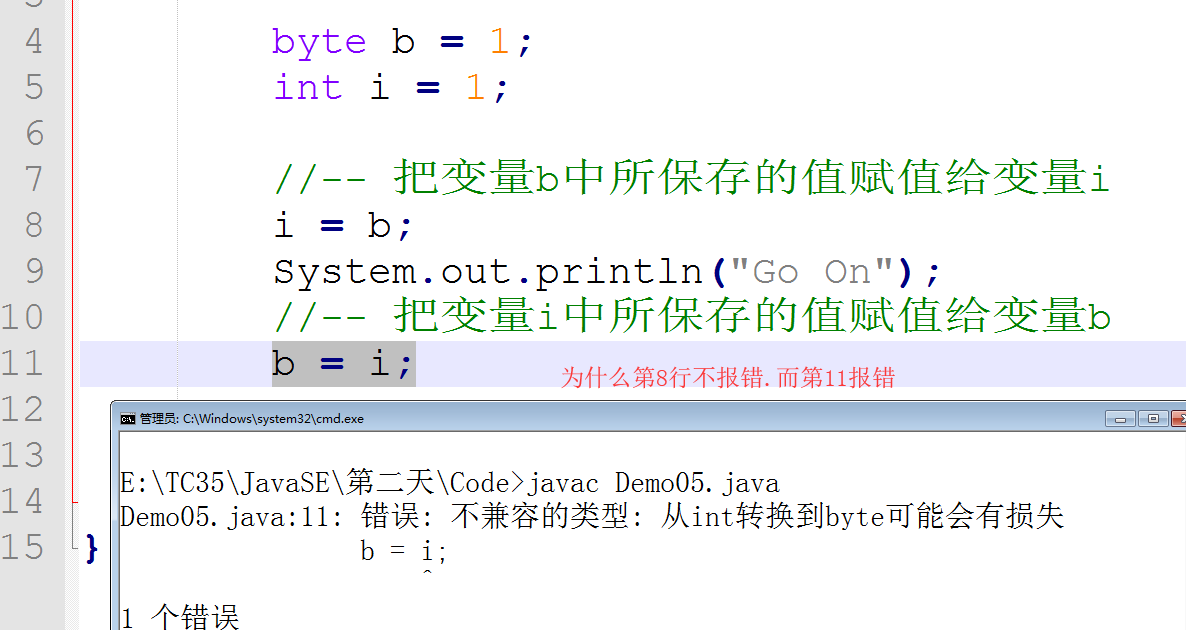
## 原生数据类型的具体大小

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:

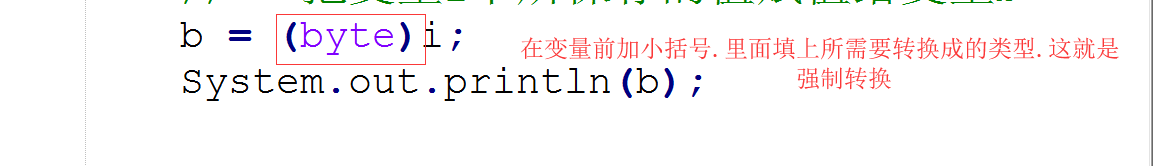
* 强制转换(把大的赋值给小的)
  + 因为看的见,所以也叫做显式转换.
* 自动转换(把小的赋值给大的)
  + 因为看不见,所以也叫做隐式转换.



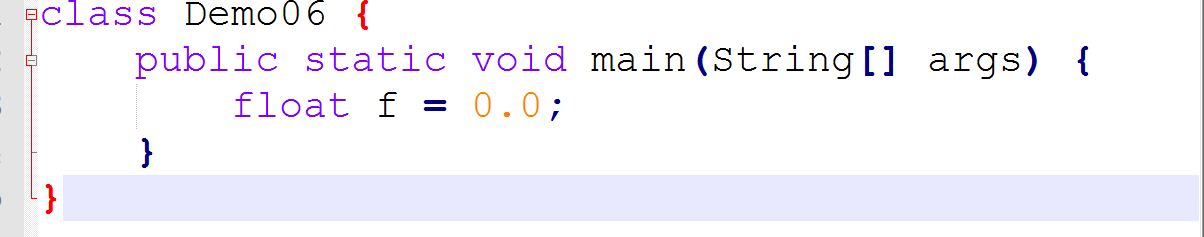
举一个例子.有俩瓶子一个500ML一个1000ML.问把那个瓶子中的内容导入那个瓶子会产生溢出现象.只有把10000ML的倒入500ML的会产生.

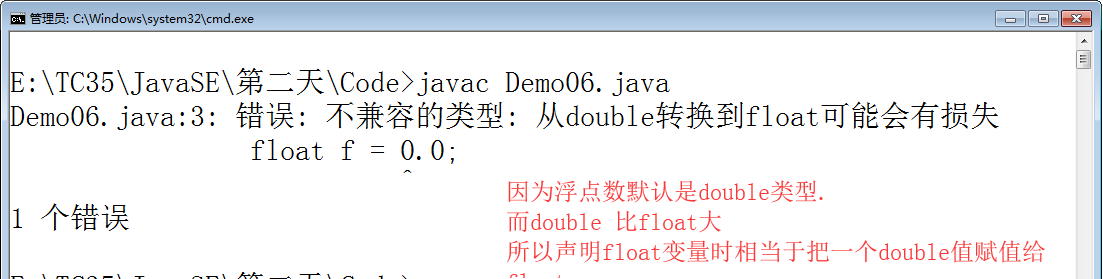
同理把这个例子理解到上面的赋值语句中.赋值语句中涉及两个类型:byte和int.int大.所以我们把byte赋值给int,对于int来说可以容纳下byte的数值.但是反过来把int赋值给byte.byte装不下int的内容.比如把+128赋值给byte.byte就存储不了.所以到时赋值失败!

遇到这种情况我们就需要使用强制转换.写法如下



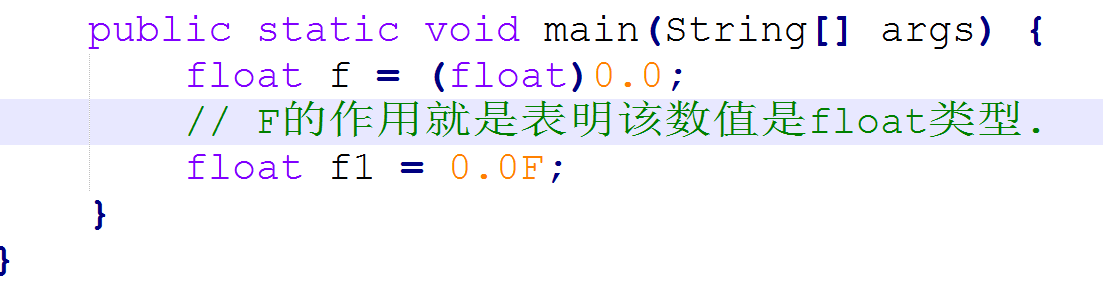
## 浮点数的注意点





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

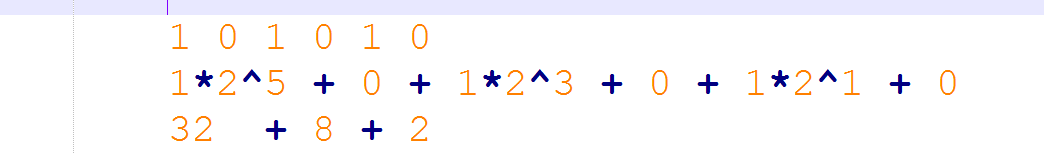
十六进制的取值是0-9和A-F

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)

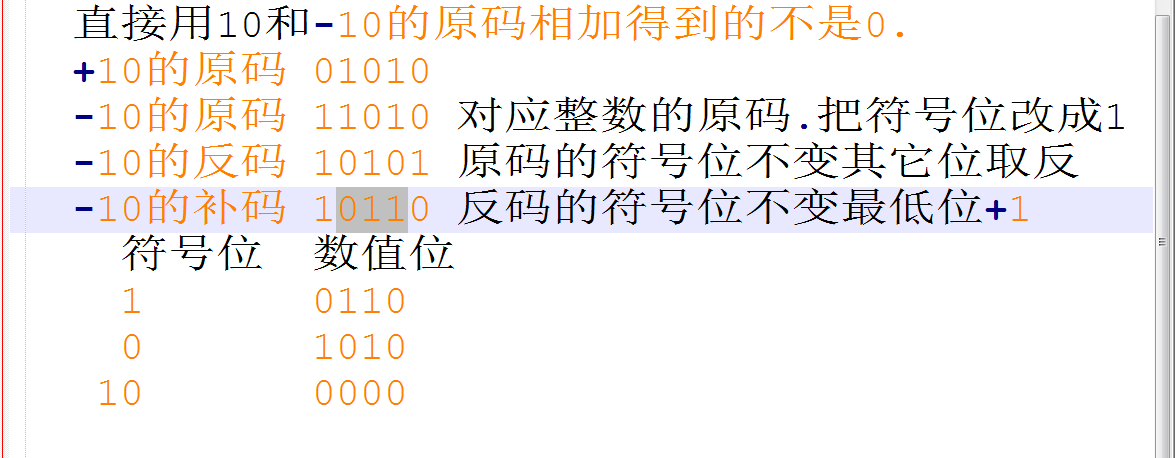
原码反码补码其实是针对负数的.因为正数原反补都是自身.而计算机在处理数据时都是按照补码来处理的.

### 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

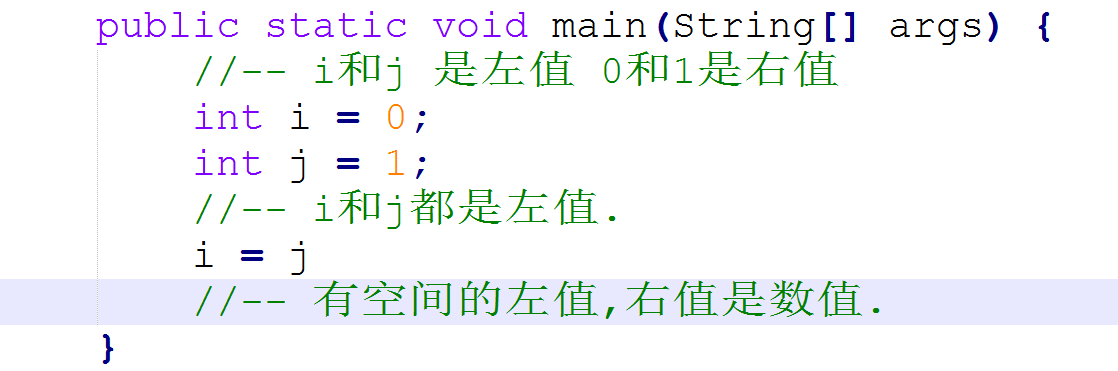
运算符左右两边的被叫做操作数.比如加法a+b.那在+号左右两边的a和b就是操作数.

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



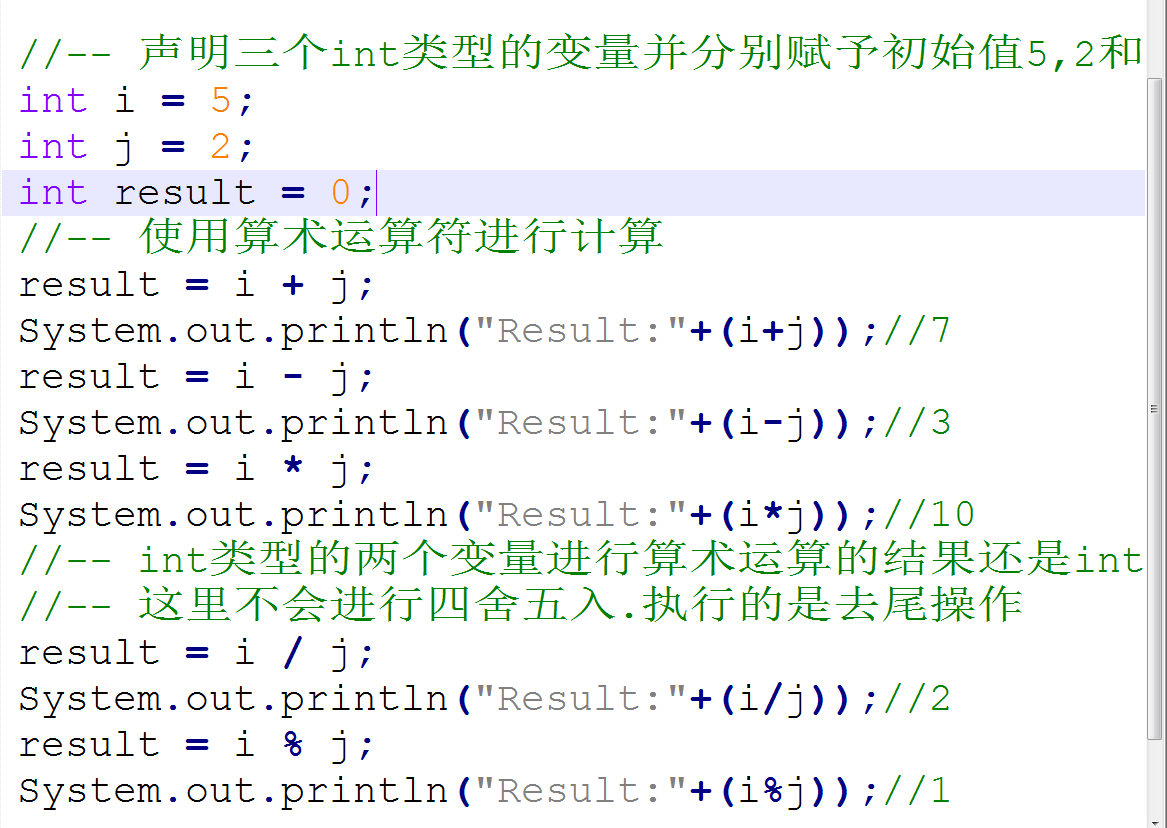
所以要求等号左边必须是左值.至于等号右边.左右值都可以

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.

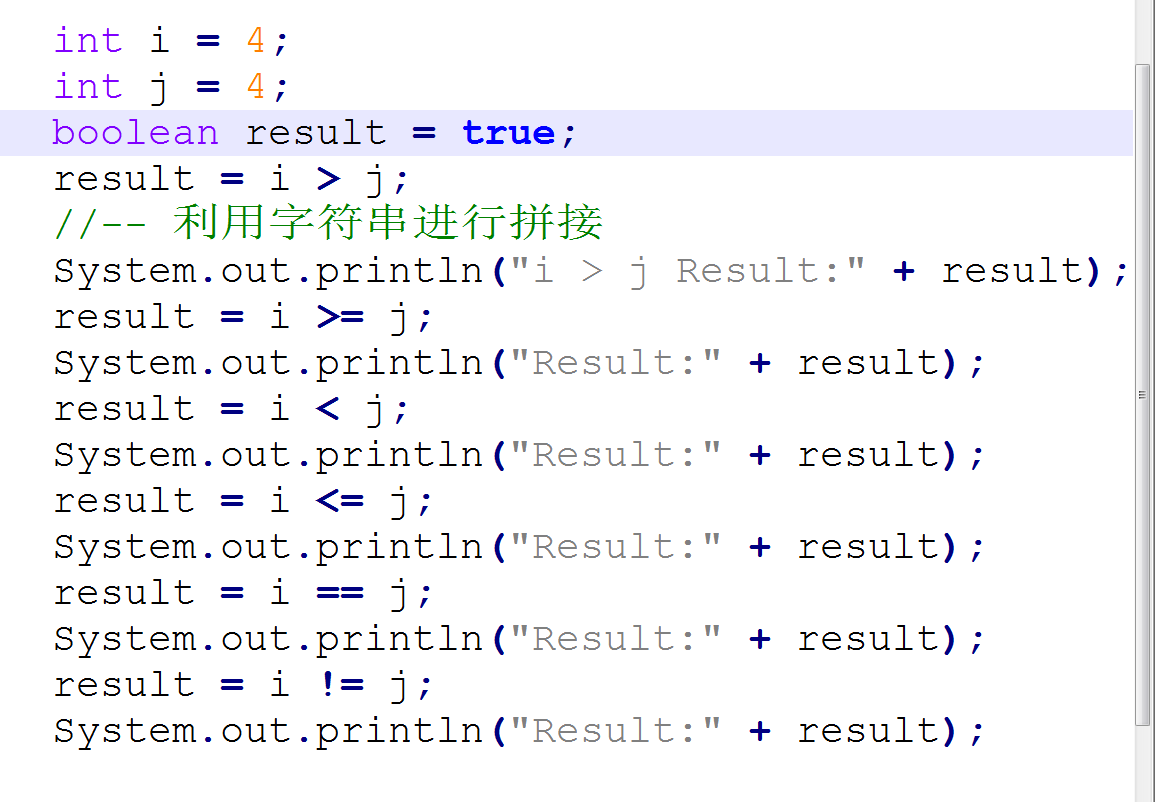
## 关系运算符

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

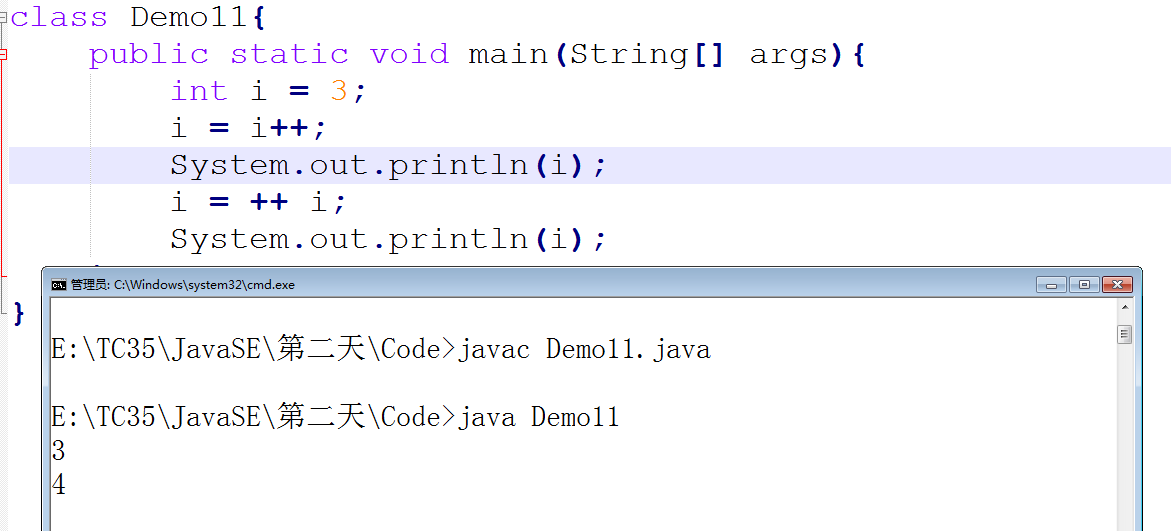
## 算术和赋值运算中的特殊点

### 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

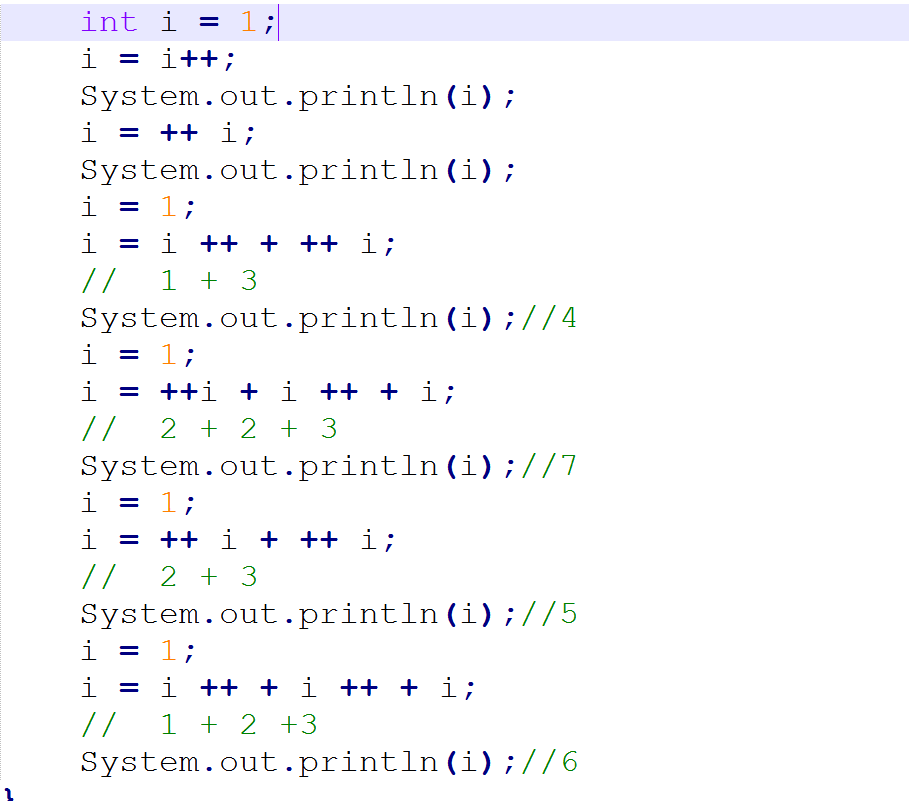
* ++i
* i++

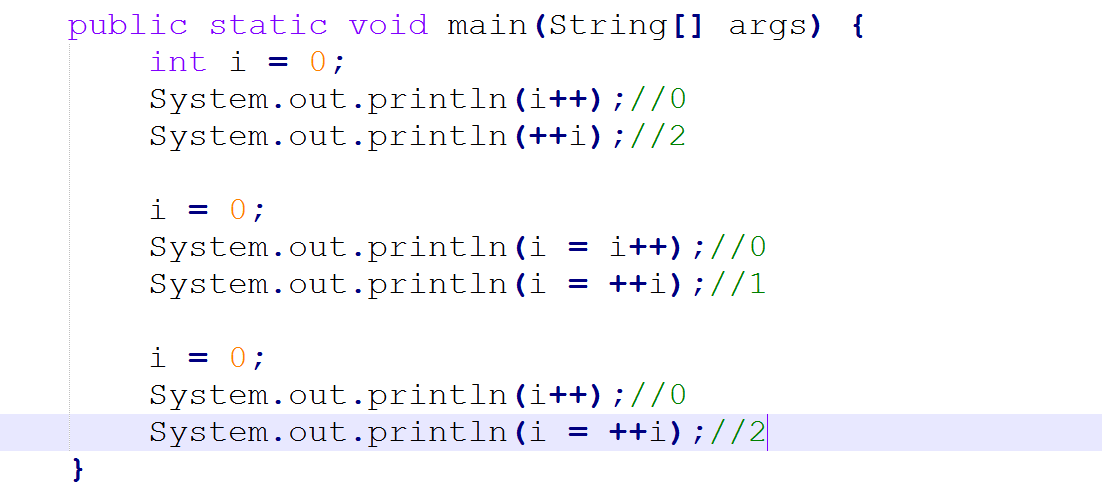
这两种写法如果单独存在都是自加1的意思.如果遇到其它运算就有区别了比如下面



++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

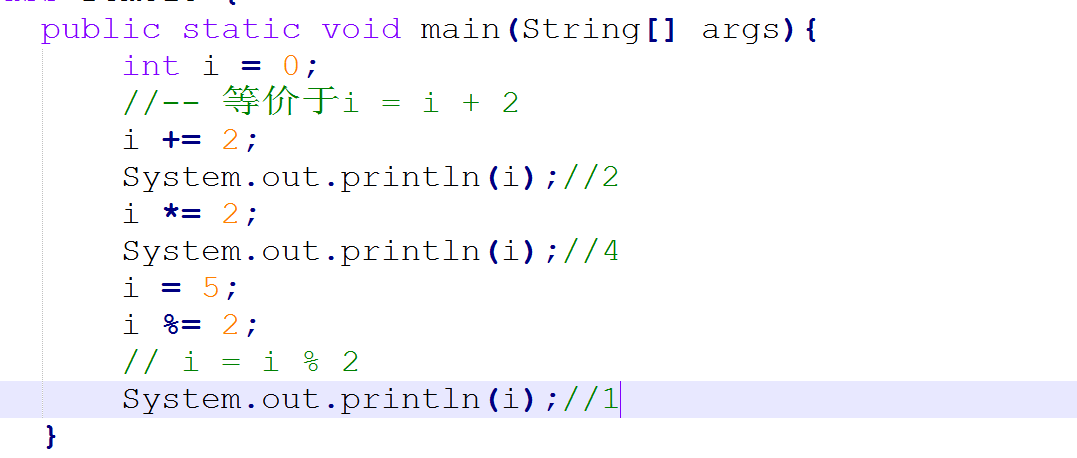
* 假设有变量i其值为1.要求使用9个加号最终结果为5.

### 算术+赋值运算

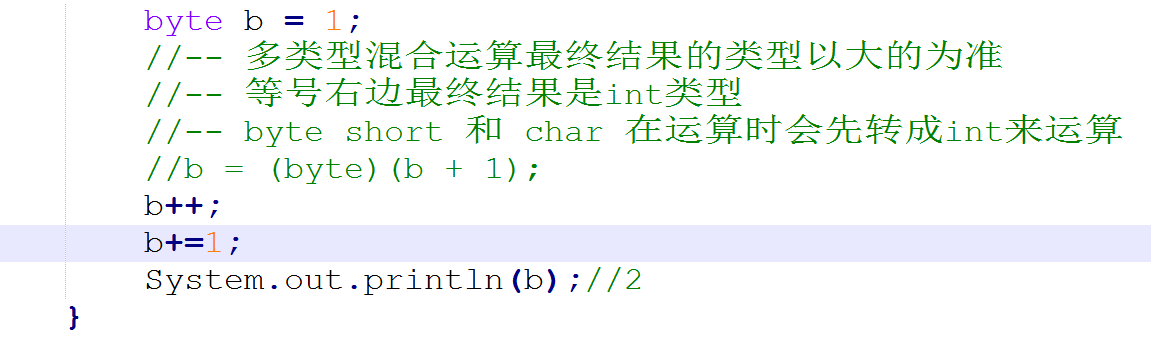
是赋值运算的一种.可以说每一种算数运算符都可以有与之对应的赋值运算

|  |  |
| --- | --- |
| 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

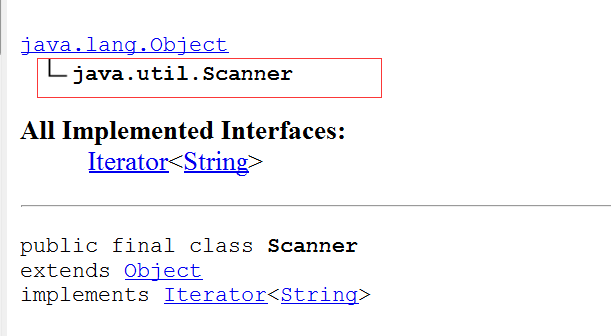
## 其它运算符(三目运算符)

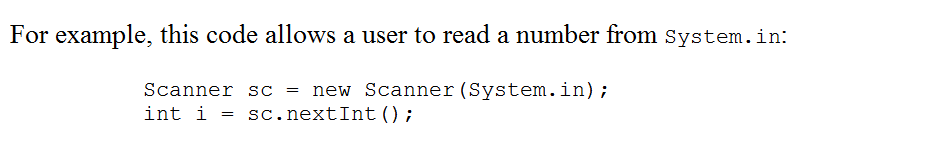
# Java中从控制台录入数据(熟练使用.大宝天天见)

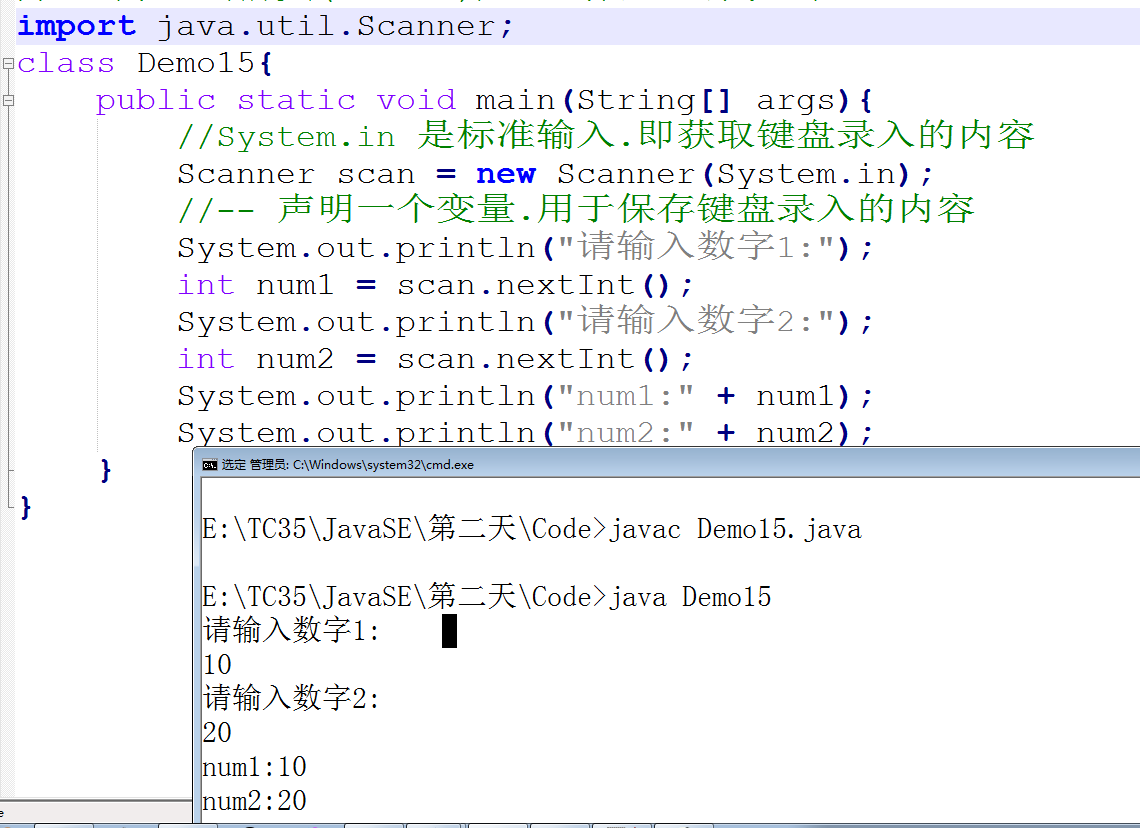
在Java中.有庞大的类库供我们使用.这些是SUN和Oracle已经为我们提供好的.只要按照他们规定的使用规范就可以直接使用.

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.

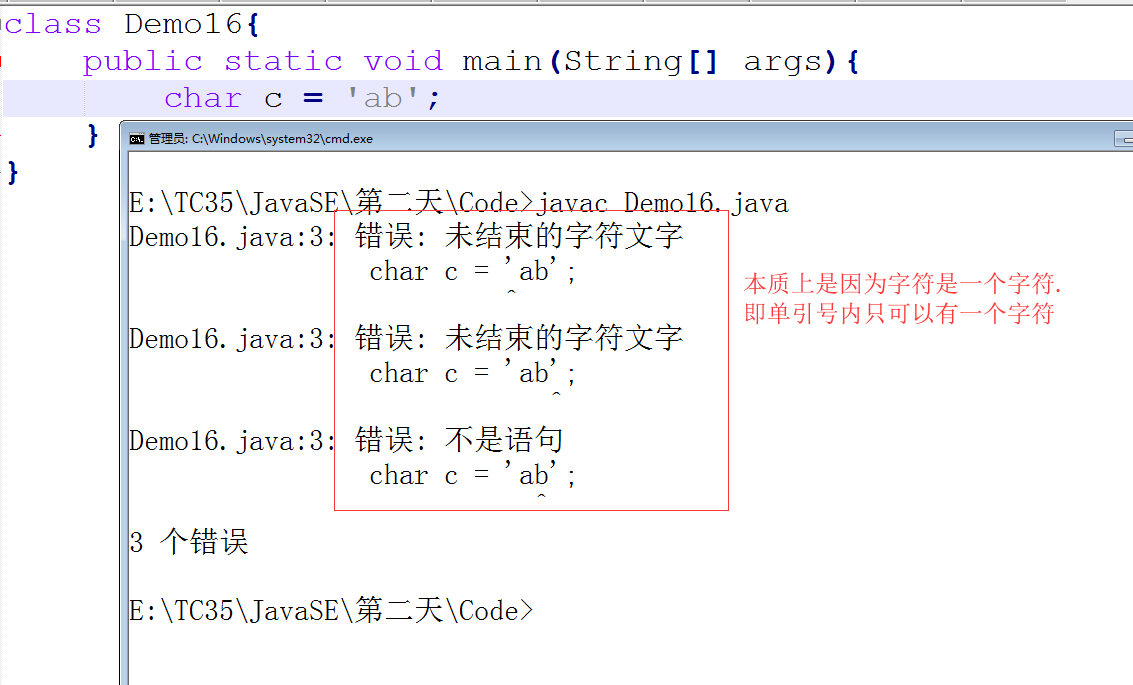






# 可能遇到的错误操作

* 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