# 变量

Java中有八种基本类型，都是Java语言预先定义好的，并且是关键字。

这八种基本类型分别是：

整型 （4种）

字符型 （1种）

浮点型 （2种）

布尔型（1种）

## 类型

### 整型

整型变量用于存放整数：

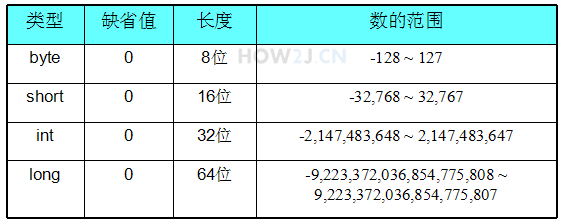
byte

short

int

long

区别在于不同类型的整型，最大值，最小值不一样。

如果试图给byte类型的变量赋予超出其范围的值，就会产生编译错误，这个时候就需要进行类型转换。

### 字符型

char类型用于存放一个字符，值用单引号'表示 (双引号表示字符串)，其长度和short一样，也是16位的，只能存放一个字符，超过一个字符就会产生编译错误。

### 浮点型

浮点数类型有两种：

float 长度为32位

double 长度为64位

注意：默认的小数值是double类型的

所以 float f = 54.321会出现编译错误，因为54.321的默认类型是 double，其类型 长度为64，超过了float的长度32

在数字后面加一个字母f，直接把该数字声明成float类型：

float f2 = 54.321f,

这样就不会出错了

### 布尔型

布尔型用于表示真假，其长度为1。

boolean b1 = true;

boolean b2 = false;

分别代表真假

虽然布尔型真正存放的数据是0(false) 1(true)，但是，不能直接使用0 1 进行赋值。

### String类型

String类型其实并不是基本类型，但是它是如此广泛的被使用，常常被误以为是一种基本类型。

String类型是Immutable的，一旦创建就不能够被改变。

## 字面值

创建一个Hero对象会用到new关键字，但是给一个基本类型变量赋值却不是用new. 因为基本类型是Java语言里的一种内置的特殊数据类型，并不是某个类的对象。

给基本类型的变量赋值的方式叫做 字面值，如下所例：

float hp = 313f;

int armor = 24;

### 整数字面值

当以l或者L结尾的时候，一个整数字面值是long类型，否则就是int类型。 建议使用大写的L而非小写的l，因为容易和1混淆。

byte,short,int和long的值都可以通过int类型的字面值来创建。整数的字面值可以用如下四种进制来表示：

十进制: 基 10, 包含从0-9的数字，平常用的就是这种

十六进制: 基 16, 包含从0-9的数字，和从A-F的字母。

八进制: 基 8, 包含从0-7的数字

二进制: 基 2, 包含0和1。（从 JAVA7开始就可以创建 二进制的字面值了）

### 浮点数字面值

当以f或者F结尾的时候，就表示一个float类型的浮点数，否则就是double类型（以d或者D结尾，写不写都可以）。

浮点数还可以用E或者e表示（科学计数法）

e2表示10的二次方，即100

1.234e2 = 1.234x100

### 字符和字符串字面值

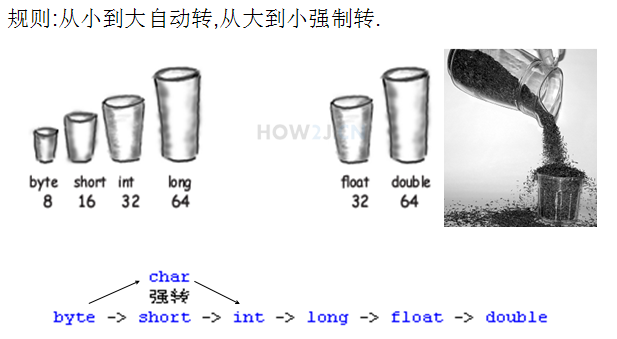
字符的字面值放在单引号中

字符串的字面值放在双引号中

需要注意的是，\表示转义，比如需要表示制表符，回车换行，双引号等就需要用 \t \r \n \" 的方式进行

## 类型转换

### 转换规则

转换规则如图所示 ：

精度高的数据类型就像容量大的杯子，可以放更大的数据

精度低的数据类型就像容量小的杯子，只能放更小的数据

小杯子往大杯子里倒东西，大杯子怎么都放得下

大杯子往小杯子里倒东西，有的时候放的下，有的时候就会有溢出

虽然short和char都是16位的，长度是一样的，但是彼此之间，依然需要进行强制转换。

### 低精度🡪高精度

long l = 50;

int i = 50;

l 是long类型的，其类型长度是64位

i 是int类型的，其类型长度是32位

所以l的精度，比i的精度要高

l = i;

把i的值赋给l， 首先l和i彼此的类型是不一样的，那么能否转换就取决于彼此的精度

这个例子，是低精度向高精度转换 是可以正常转换的

换句话说，int比较小，要放进比较大的long,随便怎么样，都放的进去

### 高精度🡪低精度

byte b = 5;

int i1 = 10;

int i2 = 300;

b = i1;

b=i2;

b的类型是byte,其长度是8，最大只能放127

i1 的类型是int, 其长度是32,最大，反正就是很大了，超过127

所以， 把int类型的数据转成为byte类型的数据，是有风险的

有的时候是可以转换的，比如 b = i1 (i1=10);

有的时候不可以转换 比如 b= i2 (i2=300) 因为放不下了

编译器就会提示错误

这个时候就只能采用强制转换，强制转换的意思就是，转是可以转的，但是不对转换之后的值负责。风险自担，后果自负。

## 作用域

## final

final 修饰一个变量，有很多种说法，比如不能改变等等

准确的描述是 当一个变量被final修饰的时候，该变量只有一次赋值的机会

# 数组

## 创建数组

### 声明数组

int[] a; 声明了一个数组变量。

[]表示该变量是一个数组

int 表示数组里的每一个元素都是一个整数

a 是变量名

但是，仅仅是这一句声明，不会创建数组

有时候也会写成int a[]; 没有任何区别，就是你看哪种顺眼的问题

### 创建数组

创建数组的时候，要指明数组的长度。

new int[5]

引用概念：

如果变量代表一个数组，比如a,我们把a叫做引用

与基本类型不同

int c = 5; 这叫给c赋值为5

声明一个引用 int[] a;

a = new int[5];

让a这个引用，指向数组

### 访问数组

### 数组长度

.length属性用于访问一个数组的长度

数组访问下标范围是0到长度-1

一旦超过这个范围,就会产生数组下标越界异常

## 初始化数组

## 排序

## 增强for循环

## 复制数组

### 二维数组

### 数组Array

# 数字

# 字符串

# 日期

# 类和对象

# 接口

# 继承

# ----

# 异常处理

# I/O

# 集合

# 泛型

# Lambda

# 多线程

# 网络编程

# JDBC

# ----

# 反射

# 注解