# Java简介

1. Java版本：java se 标准版，java ee 企业版，java me 微型版
2. Java运行过程：

编写源代码 source.java 🡪 javac 编译源代码生成字节码 source.class 🡪 java 运行

# Java 基本程序设计结构

1. Java基本格式

public class Main {

public static void main(String[] args) {

System.out.println("Hello World!");

}

}

必须有一个main方法，声明为public static

1. 数据类型

8种基本类型 🡪 4种整型，2种浮点型，1种用于表示Unicode编码字符类型char，1种布尔型

1. 整型

int 4字节 -2^31 ~ 2^31-1（>20亿）

short 2字节 -2^15 ~ 2^15-1

long 8字节 -2^63 ~ 2^63-1

byte 1字节 -2^7 ~ 2^7-1

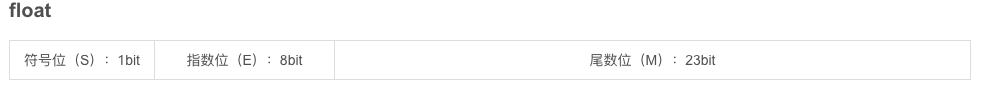
* 长整型后加L或l ，十六进制前加0x或0X，八进制前加0，二进制前加0b或0B
* 没有任何无符号形式的整形

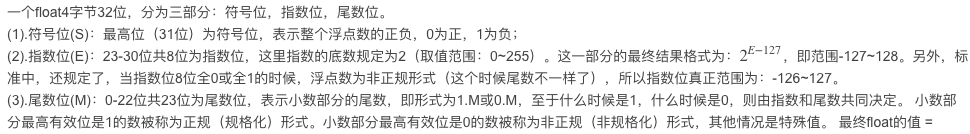
1. 浮点类型

float 4字节 -2^128 ~ 2^128 精度 6～7位

double 8字节 -2^1024 ~ 2^1024 精度 15～16位

float 数据存储结构



flost = (−1)S∗(2^E−127)∗(1.M)

细说浮点数：<https://segmentfault.com/a/1190000005015151>

* float 类型的数值由一个后缀F或f，没有的默认为double类型
* 三个特殊的浮点数值，正无穷大，负无穷大，NaN（不是一个数字）
* 使用Double.isNaN(x) 检查x是不是一个NaN

1. char类型

描述了UTF-16编码中的一个代码单元

1. boolean类型

false true，用来判定逻辑条件，整型值，布尔值之间不能进行相互转换

1. 变量
2. 变量声明

double salary;

每个变量都有一个类型（type），变量名由Java“字母”开头，由“字母”，“数字”构成的序列。

* 使用Character类的isJavaIdentifierStart ，isJavaIdentifierPart检查Unicode是否属于Java“字母”

1. 变量初始化

int vacationDays = 12;

声明一个变量后，必须用赋值语句对变量进行显示初始化

1. 常量
2. final double CM\_PER\_INCH = 2.54;

使用final指示常量，只能被赋值一次，一般使用大写

1. public static final double CM\_PER\_INCH = 2.54;

定义一个常量在一个类的多个方法中使用，称为类常量，使用关键字static final

类常量定义位于main方法的外部，声明为public

1. 数学函数
2. + - \* / %

整数被0除将会产生一个异常，浮点数被0除回得到无穷大或NaN

浮点数运算时中间结果默认为允许扩展的指数，可使用严格的strictfp关键字声明，

计算时中间结果强制截断

1. Math类函数

sqrt(x) 🡪 平方根

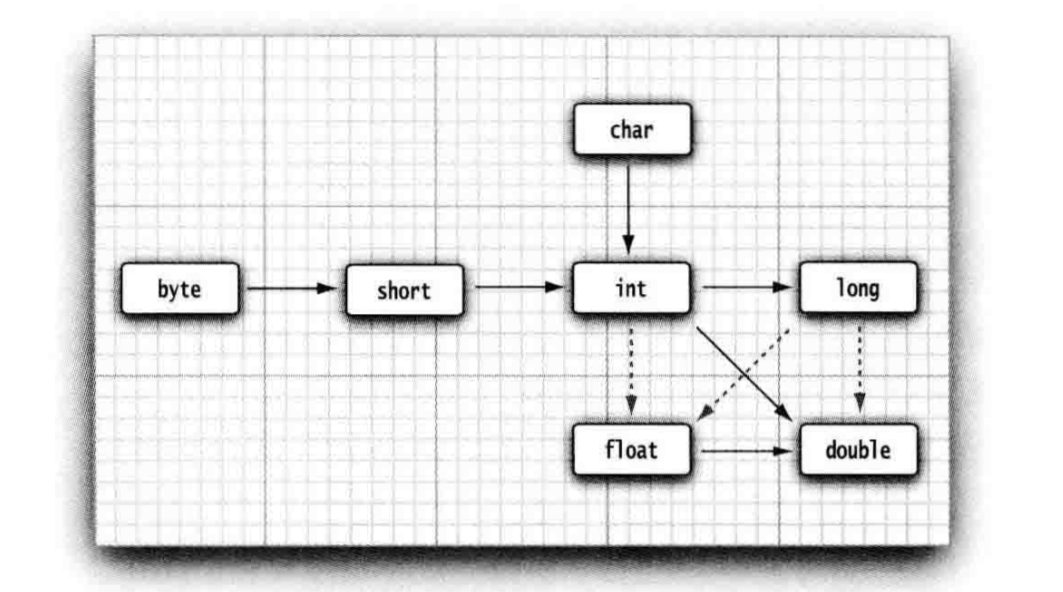
pow(x, a) 🡪 幂运算

%取余，floorMod取模：c = x/y, r = x-(c\*y)

区别：取余在计算商值向0方向舍弃小数位，取模在计算商值向负无穷方向舍弃小数位，%(x,y), floorMod(x,y), x,y符号相同时结果相同，符号

不同时，%符号是x，floorMod符号是y，python中%是取模运算。

1. 数值类型之间的转换
2. 实线表示无精度损失转换，虚线表示精度损失转换



1. 强制类型转换

造成信息丢失的类型转换需要通过强制类型转换(cast)实现：

int 🡪 double double a; int b = (int) a

强制类型转换通过截断小数部分将浮点值转换为整型，若进行舍入运算使用：

double a; int b = (int)Math.round(a) 🡪 round()方法返回long类型

不要使用boolean类型与任何数值类型之进行强制类型转换

1. 结合赋值和运算符

x += 4 🡪 x = x + 4

若左侧类型与结果不同，就会发生强制类型转换

1. 自增与自减运算符

同C语言

不建议使用

1. 关系和boolean运算符

==; !=; <; >; <=; >=;

&&; || 短路计算方式，若第一个条件为Ture，不计算第二个条件

condition ? expression1 : expression2

位运算：&(and); |(or); ^(xor); ~(not) boolean使用位运算时不采用短路运算，两个条件都计算

>>和<< 运算符将位模式左移或右移，>>>无符号右移

详细分析：<https://www.cnblogs.com/chuijingjing/p/9405598.html>

运算优先级