1. Java的起源
2. Java是1995由詹姆斯·高斯林等人一起设计的第三代面向对象的开发语言，他被称为“Java之父”。
3. JDK的安装
4. 下载exe或者zip文件然后安装
5. 配置环境变量
   1. Path： .;%JAVA\_HOME%\bin
   2. JAVA\_HOME: jdk的安装目录
   3. Classpath(5.0后不用配置，也可以配置)
6. JDK、JRE、JVM三者之间的关系
   1. JDK包含JRE，JRE包含JVM
7. Java程序运行过程
   1. 编译：通过编译器将Java的源代码编译成.class字节码文件
   2. 运行：通过JRE和JVM运行.class文件
   3. 标识符(类、变量、方法起名)
      * 1. 以字母(Unicode字符集)、数字、下划线开头
        2. 其余部分可以是：字母、数字、下划线、$
        3. 不能是关键字
8. 数据存储单位
   1. 位: (bit, 比特),二进制位，每位存储0或1
   2. 字节: (byte, B拜特)

1 byte = 8 bit

千字节：1KB=1024byte=2^10byte

兆字节：1MB=1024KB=2^10KB

吉字节：1GB=1024MB=2^10MB

* 1. 两个bit能表示多少种状态？ 4种：00 01 10 11
  2. 11011110有几个bit? 8个bit, 占1个字节
  3. 一台计算机的字长是2个字节，那就是CPU作为一个整体加以传递的二进制代码是多少位？ 16位 = 2\*8

1. Java中8种基本数据类型以及其他数据类型

|  |  |  |  |
| --- | --- | --- | --- |
|  | 类型 | 占用存储空间 | 表数范围 |
| 布尔型 | boolean | 1bit | 0/1 |
| 字符型 | char | 2字节 | 0-65536之间的数  Unicode编码表中的字符  运行时可以直接当整数来运算, Java中允许使用转义符将字符转变为其他含义如：char a = ‘\n’代表回车 |
| 数值型 | byte | 1字节 | -128 - 127 |
| Short | 2字节 | -2^15-2^15-1(-32768-32767) |
| int | 4字节 | -2^31-2^31-1 约21亿 |
| long | 8字节 | -2^63 – 2^63-1 |
| float | 4字节 | -3.403E38 – 3.403E38 |
| double | 8字节 | -1.798E308-1.798E38 |
| 引用数据类型 | 类，数组，接口等 | 4字节 |  |

1. 类型转换：
   1. 自动类型转换：容量小的可以自动转为容量大的，可能存在丢失精度

byte –>short-->int -->long-->float-->double

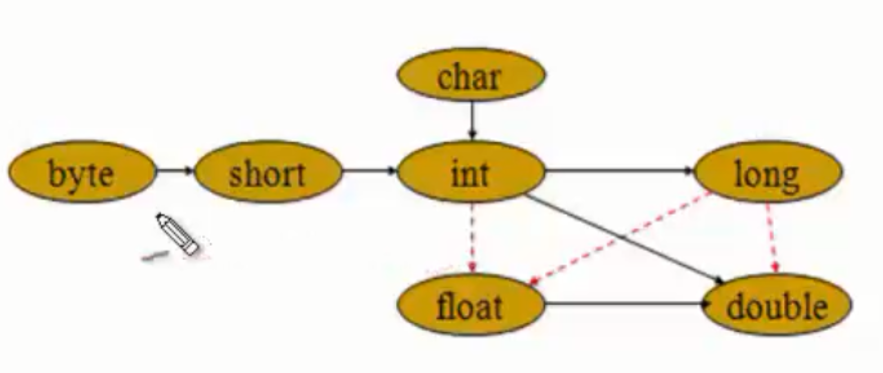
char-->int

int-->float

int-->double

long-->double

int-->float, long-->float, long-->double都可能存在精度丢失



* 1. 强制类型转，有可能导致溢出/精度损失得到一个未知的值
  2. 运算时注意类型自动提升
  3. JDK7新特性：ob开头代表二进制，整数可以用下划线分割

1. Java中的运算符
   1. Java中运算符总结

|  |  |
| --- | --- |
| 分类 | 操作符 |
| 算术运算符 | +、-、\*、/、%、++、-- |
| 关系运算符 | ==、!=、>、<、>=、<=、instanceof |
| 位运算符 | &、|、^、~、<<、>>、>>> |
| 逻辑运算符 | &&、||、! |
| 赋值运算符 | =、+=、-=、\*=、/=、%=、<<==、>>=、&=、^=、|=、 |
| 其他运算符 | ?: |
| 运算符的优先级做基本了解即可。实际开发中多用()来处理运算的优先级问题 | |

* 1. 位运算符补充

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 例子 |
| ＆ | 如果相对应位都是1，则结果为1，否则为0 | （A＆B），得到12，即0000 1100 |
| | | 如果相对应位都是0，则结果为0，否则为1 | （A | B）得到61，即 0011 1101 |
| ^ | 如果相对应位值相同，则结果为0，否则为1 | （A ^ B）得到49，即 0011 0001 |
| 〜 | 按位取反运算符翻转操作数的每一位，即0变成1，1变成0。 | （〜A）得到-61，即1100 0011 |
| << | 按位**左移**指定的位数，空缺补0。相当于**A\*2^b** | A << 2得到240，即 1111 0000 |
| >> | 按位**右移**指定的位数，空缺补最高位数。相当于**A/2^b** | A >> 2得到15即 1111 |
| >>> | **无符号右移**，按位右移指定的位数，空缺补0。相当于**A/2^b** | A>>>2得到15即0000 1111 |

* 1. >> 1： 相当于除以2， <<1：相当于乘以2, >>>1: 相当于除以2
  2. 补充知识：原码，反码，补码,见：原码反码补码.docx文件

1. 控制语句
   1. 顺序
   2. 选择(单选，双选，多选)
      1. If else if.. else
      2. Switch case 注意break穿透
   3. 循环
      1. While(){…}
      2. Do…while(…)
      3. For(…){…}
2. 补充知识：Unicode编码

编码：

ASCII: ASCII 码一共规定了128个字符的编码，比如空格SPACE是32（二进制00100000），大写的字母A是65（二进制01000001）。这128个符号（包括32个不能打印出来的控制符号），只占用了一个字节的后面7位，最前面的一位统一规定为0。

Unicode: Unicode 当然是一个很大的集合，现在的规模可以容纳100多万个符号。每个符号的编码都不一样，

编码实现方式：UTF-8, GB2312等

互联网使用最广的是UTF-8

**UTF-8 是 Unicode 的实现方式之一**

UTF-8 最大的一个特点，就是它是一种变长的编码方式。它可以使用1~4个字节表示一个符号，根据不同的符号而变化字节长度。

UTF-8 的编码规则很简单，只有二条：

1）对于单字节的符号，字节的第一位设为0，后面7位为这个符号的 Unicode 码。因此对于英语字母，UTF-8 编码和 ASCII 码是相同的。

2）对于n字节的符号（n > 1），第一个字节的前n位都设为1，第n + 1位设为0，后面字节的前两位一律设为10。剩下的没有提及的二进制位，全部为这个符号的 Unicode 码。

下表总结了编码规则，字母x表示可用编码的位。

Unicode符号范围 | UTF-8编码方式

(十六进制) | （二进制）

----------------------+---------------------------------------------

0000 0000-0000 007F | 0xxxxxxx

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

根据上表，解读 UTF-8 编码非常简单。如果一个字节的第一位是0，则这个字节单独就是一个字符；如果第一位是1，则连续有多少个1，就表示当前字符占用多少个字节。

以汉字严为例，演示如何实现 UTF-8 编码。

严的 Unicode 是4E25（100111000100101），根据上表，可以发现4E25处在第三行的范围内（0000 0800 - 0000 FFFF），因此严的 UTF-8 编码需要三个字节，即格式是1110xxxx 10xxxxxx 10xxxxxx。然后，从严的最后一个二进制位开始，依次从后向前填入格式中的x，多出的位补0。这样就得到了，严的 UTF-8 编码是11100100 10111000 10100101，转换成十六进制就是E4B8A5。

参考：https://www.ruanyifeng.com/blog/2007/10/ascii\_unicode\_and\_utf-8.html