复习：

1. Java技术结构：j2se,j2ee,j2me
2. Java的跨平台：JVM是Java能够跨平台的前提---Java语言是跨平台，JVM不是跨平台的
3. 环境变量的配置：JAVA\_HOME=JDK的安装目录 Path=%JAVA\_HOME%\bin;
4. javac -d 路径 要编译的Java文件--编译
5. java -cp 路径 包名.类名---运行
6. 关键字：53个---const，goto---关键字都是小写---System,String
7. 标识符：字母、数字、\_、$组成，数字不能开头，不能使用关键字，见名知意，大小写敏感---Windows系统区分大小写么？---不区分

### 注释

在代码中解释程序的文字

// 注释文字 --- 单行注释

/\*注释文字\*/ --- 多行注释

/\*\*注释文字\*/ --- 文档注释---可以利用javadoc将注释内容来提取处出来形成文档---只能放在类或者函数上，javadoc命令只能用于提取公共类

### 计算机常量

本身不可改变的量

整数常量：所有的整数 3，4，100

小数常量：所有的小数 1.02,5.36，8.7209

字符常量：将一个符号用 ‘’ 标志起来就构成一个字符常量 ‘a’ ‘+’ ‘1’ ‘ ’

字符串常量：将多个字符用 “”标志起来就构成了一个字符串常量 “abc” “12w” “a” “”

布尔常量：只有两个值---true/false---用于表示逻辑值

空常量：null

考虑：2--整数, 2.0---小数 ‘2’--字符, “2”---字符串, ‘2.0’--非法, “2.0”--字符串 分别是什么常量？

### 进制

计数方式

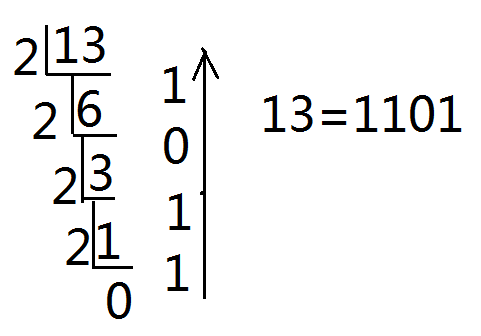
二进制:0-1，满2进1 在JDK1.7以前Java代码中不允许直接表示二进制数字，从jdk1.7开始允许在代码中使用二进制数字，要求以0b/0B开头 1+1=10 11+1=100 0b100100 0B10111

八进制：0-7，满8进1 以0开头标志八进制数字 7+1=10 16+1=17 17+1=20 015 07 012 027

十进制：0-9，满10进1

十六进制：0-9，A-F/a-f，满16进1 以0x/0X开头标志十六进制数字 9+1=a a+1=b f+1=10 0x23 0xa 19+1=1a 1f+1=20

十进制转换成二进制：不断地除以2取余数，将余数倒排



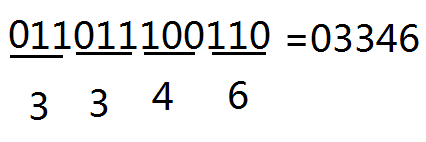
二进制转化成十进制：从低位次开始，每一位乘以2的位次之幂，然后将积求和



十进制转化为其他进制：就是除以对应的进制然后取余倒排

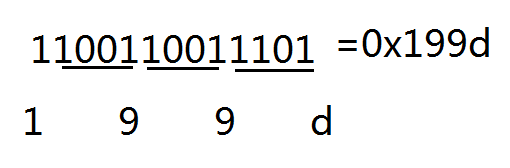
其他进制转化为十进制：乘以对应进制的位次次幂然后求和

二进制转换成八进制：从低位次开始，每三位化为一组，产生一个八进制数字，最高位如果不足三位，则补0---三变一



八进制转换为二进制：一变三

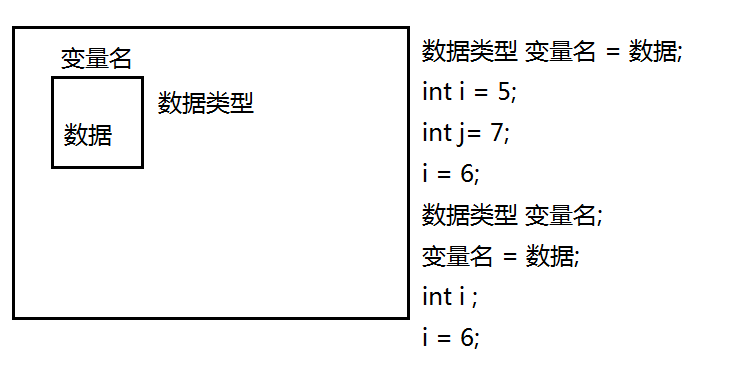
二进制转化为十六进制：四变一



十六进制转化为二进制：一变四

### 变量

存储数据的



变量名不能重复。

int i, j,k;---允许一次性定义多个同一类型的变量，用,隔开

int i;

double j;

变量先定义后使用，先给值后使用，在哪儿定义在哪儿用

扩展：每一个二进制数字称之为一位(bit---b)--->字节（Byte---B）---1B=8b 1KB=1024B 1MB=1024KB -> GB -> TB

## 数据类型

### 基本数据类型

数值型

整数型

byte---字节型---1个字节 --- -27~27-1 - -128~127

byte b = 25; byte b2 = 128;---false

short---短整型---2个字节 --- -215~215-1 --- -32768~32767

short s = 3; short s = -109;---几乎不用

int---整型---4个字节--- -231~231-1 --- 2.1\*1010~2.1\*1010---整数默认为int类型

System.out.println(6); int i = 129;从JDK1.7开始int j = 3\_186\_485\_318; ->编译完成之后就是int j = 3186485318;

long---长整型---8个字节--- -263~263-1 --- -9\*1018~9\*1018---后边添加L/l标志是一个长整型

long l = 31864853188L;

浮点型

float---单精度---4个字节 --- -1038~1038---要求必须以F/f标志float类型的数字

float f = 3.23f;

double---双精度---8个字节--- -10308~10308 ---小数默认为double

System.out.println(3.4); double d = 4.2; double d2 = 5.6D; double d3 = 6.01d;

double d = 3e3; --- 科学计数法 3\*103=3000.0

double d = 0x3p3;---科学计数法 3\*23=24.0

char---字符型---2个字节--- 0-216~1 --- 0~65535

char c = ‘中’; char c2 = ‘\u52fa’;

‘a’ --- 97 ‘A’---65 ‘0’---48

‘中’ gbk---2 utf-8--3 utf-16---2

‘a’ iso---1 gbk---1 u8-1 u16-1

#### 转义字符

‘\t’ ---table 制表符 ‘\r’---return 回车 ‘\n’---next/newLine 换行

‘\\’ ---\ ‘\’’ --- 单引号 ‘\”’ --- 双引号

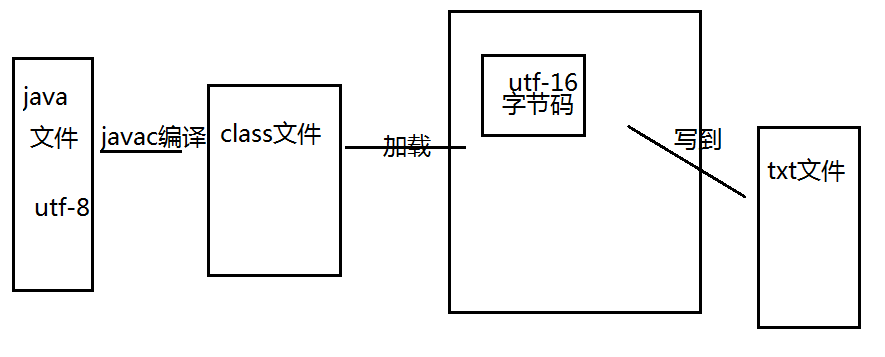
编码：将数据按照指定的规则转化为数字---编码表---码表---ASCII

规定所有的码表默认兼容西欧码表

ISO-8859-1---西欧码表---1个字符1个字节

gb2312---国标码---2个字节一个字符---收录了常见的简体汉字以及一部分的常见繁体汉字---gbk

Unicode编码体系---收录了常见语言的常见字符---Java采用了Unicode编码体系--- utf-8---三个字节一个字符，Java文件默认是使用utf-8；Java文件编译完成之后会产生对应的class文件，class文件加载到内存之后是以utf-16形式存储的。utf-16规定一个字符2个字节



boolean ---布尔型---true/false---内存大小根据jdk版本和操作系统来确定

boolean b = true;

引用数据类型

数组[] 类class 接口interface

### 数据类型的转换

#### 自动类型转换/隐式转换

byte b = 120;

int i = b;

float f = 3.2f;

double d = f;

规律一：小的类型能够自动转化为大的类型

int i = 6;

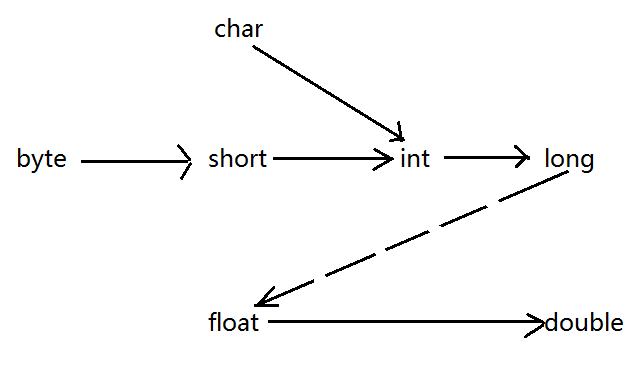
float f = i;

规律二：整数能够自动转化为小数，但是可能会有精度损失

char c = ‘a’;

int i = c;

规律三：字符可以自动转化为整数



float f = 23;---true---23是整数，默认为int

float f = ‘a’;---true

long l = 231;---true

short s = ‘a’;---true--- ‘a’是一个常量，因此在编译的时候值是确定的，所以在编译的时候jvm会去检查 ‘a’是否在short的取值范围内，如果范围符合，则可以转换

char c = 97;---true---97是一个常量，因此在编译的时候取值也是确定的，所以在编译的时候jvm会发现97在char类型的范围内，则可以转换

short s = 97;

char c = s;---false---s是一个变量，在编译的时候只能确定一件事儿---只能确定s的类型short类型，就只能比较short和char类型的范围是否符合---由于char不能完全包含short的取值范围，所以会产生范围的冲突---不允许转换，编译不通过

char c = ‘a’;

short s = c;---false

#### 强制类型转换/显式转换

double d = 3;

int i = (int)d;

double d = -3.9;

int i = (int)d; --- i = -3;

注意：小数强转为整数的时候会舍弃小数部分

int i = 128;

byte b = (byte)i;

### 数据的原反补三码

任意一个数据都有原码、反码、补码这么三种形式。

数据在计算机中是以补码形式来存储的，因此实际上计算的也是数据的补码。

对于正数，原反补三码一致。

int i = 5;

原码：00000000 00000000 00000000 00000101

反码：00000000 00000000 00000000 00000101

补码：00000000 00000000 00000000 00000101

对于负数，反码是在原码的基础上最高位不变，其余位1->0,0->1;补码在反码的基础上+1---最高位是符号位，0表示正数，1表示负数

int i = -3;

原码：10000000 00000000 00000000 00000011

反码：11111111 11111111 11111111 11111100

补码：11111111 11111111 11111111 11111101

## 运算符

### 算术运算符

+ - \* / % ++ --

注意：

1. byte/short/char在运算的时候会自动的提升为int类型
2. 整数运算完成之后的结果一定是整数
3. 当小类型和大类型一起运算的时候结果一定是大的类型
4. byte b = 4 + 1; --- 4 和1都是常量，值确定，所以在编译的时候进行优化，编译完成之后结果就是byte b = 5;
5. 整数/0 -- ArithmeticException---算术异常 任意一个非零小数/0;任意一个非零数字/0.0---Infinity 0.0/0 0/0.0 0.0/0.0---NaN---Not a Number---非数字

% 取模---取余

7%3=1 2%5=2

-18%5=-3 -56%11=-1

18%-5=3 56%-11=1

-18%-5=-3 --- 对于负数的取余，先忽略符号按照正数取余来运算，运算完成之后看%左边数字的符号，如果左边为正，结果为正，左边为负，结果为负。

3%1.2=0.6 4.73%1.54=0.11

注意：小数无法精确运算---绝大部分小数在内存中无法精确存储

++/-- 自增/自减

如果++/--在前，先自增/自减然后参与后续运算

如果++/--在后，先将值取出来参与后续运算然后再自增/自减

int i = 5;

int j = ++i; -> 先将i自增为6，然后将6赋值给j，所以j的值也是6；

int i = 5;

int j = i++;-> 先将i的值5取出来，然后i自增为6，最后将5赋值给j

int i = 3;

int j = ++i + 1; -> 先将i自增为4，然后将i的值4取出来参与+1，最后将4+1赋值给j，所以j的值为5

int i =3;

int j = i++ + 1; -> 先将i的值3取出来参与+1运算，i再自增为4，然后将3+1的值赋值给j，所以j的值是4

int i = 5;

int j = ++i \* 2; -> j = 12

int i = 5;

int j = i++ \* 2; -> j = 10

int i = 9;

int j = i++ + ++i; 9 + 11 -> j = 20

int i = 9;

int j = ++i + i++; 10+10->j=20

注意：byte/short/char可以参与++/--运算，结果依然是原来的类型---底层做了强制转换

### 赋值运算符

= += -= \*= /= %= &= |= ^= <<= >>= >>>=

int i = 8;

i += 2; -> i = i + 2; -> i = 10

int i = 6;

i -= 4; -> i = 2;

注意：

1. 除了=以外，其余的符号要求变量必须先有值才能使用
2. byte/short/char可以参与赋值运算

int i = 5;

i += i -= i\*= i++; -> i=-15

i = 5 + ( 5 - (5 \* 5));

int i = 5;

i = i++; ---先将i的值5取出来，然后i自增为6，最后将5赋值给i -> i = 5