# 计算机语言分类

**①　机器语言**：直接用二进制指令表达，指令是用0和1组成的一串代码，它们有一定的位数，并分成若干段，各段的编码表示不同的含义（如0000 代表 加载（LOAD）,0001 代表 存储（STORE））。

**②　汇编语言**：使用一些特殊的符号来代替机器语言的二进制码(又称符号语言)，计算机不能直接识别，需要用一种软件将汇编语言翻译成机器语言，汇编语言依赖于硬件体系，开发难度大（如加法指令ADD/ADC、减法指令SUB/SBB）。

**③　高级语言**：使用一定格式的自然语言进行编写源代码，通过编译器将源代码翻译成计算机直接识别的机器语言，之后再由计算机执行，不直接操作硬件，把繁琐的翻译操作交给编译器完成。

# JAVA描述

历史

美国SUN(Stanford University Network)公司，在中国大陆正式中文名为“太阳计算机系统（中国）有限公司”。

1982年，SUN公司诞生于美国斯坦福大学校园，并于1986年上市，在NASDAQ（纳斯达克：是全美证券商协会自动报价系统）的标识为SUNW，2007年改为JAVA。

2009年4月20日19点40分，美国数据软件巨头甲骨文公司（Oracle）宣布以74亿美元收购SUN公司，从此Java也有“干爹”了，在这个拼爹的时代，Java的发展前景不容置疑。

Java的三大平台

1999年SUN公司发布了基于Java的三个平台技术标准：J2SE,J2EE,J2ME,2005年开始分别改名为：JavaSE，JavaEE，JavaME。随着J2EE的发布，Java正式的迎来了属于自己的“春天”。

①　Java SE（Java Platform Standard Edition）：Java标准平台，开发和部署在桌面、服务器、嵌入式环境和实时环境中使用的Java 应用程序，如做一个桌面版的QQ，JavaSE能为Java EE和JavaME提供开发基础。

②　Java EE（Java Platform Enterprise Edition）：Java企业平台，开发和部署可移植、健壮、可伸缩且安全的服务器端Java应用程序，如企业的应用系统ERP等。Java EE是在Java SE的基础上构建的，它提供Web服务、组件模型、管理和通信API，可以用来实现企业级的面向服务体系结构（SOA）和Web等应用程序。

③　Java ME（Java Platform Micro Edition）：Java微型平台，也叫K-JAVA，开发移动设备和嵌入式设备(比如手机、PDA、电视机顶盒和打印机等)。随着Android迅猛发展，JavaME使用越来越少了，Android会逐渐的取代JavaME的市场份额。

以Java作为开发语言的其他设备有很多，如：如日中天的Android智能设备。

学习Java，我们都是从JavaSE开始的，只有学好JavaSE才能很好的掌握JavaEE和Android的知识。对于Java的就业，很少有人从事单纯的JavaSE开发的，这不是Java的强势之处，我们应该面向市场更广阔的JavaEE和Android就业方向。

Java虚拟机

JRE(Java Runtime Environment)：Java运行环境，如果要运行Java程序，就需要JRE的支持，JRE里包含JVM。

JDK(Java Development Kit)：Java开发工具，包含开发Java程序的所有工具,如javac和java等，JDK里包含JRE。

Java虚拟机（Java Virtual Machine）简称JVM:它是运行所有Java程序的虚拟计算机，好比是街机游戏的模拟器。

JVM是Java语言的运行环境，也是Java 最具吸引力的特性之一。JVM用于读取并处理编译过的与平台无关的字节码(class)文件，从而实现Java的可移植性。但是值得注意的是Java虚拟机是不跨平台的。也就是说在Win下得装Win版的JVM，在Linux下装Linux版的JVM。

JVM是Java程序的解释和执行器。

Java语言特性

简单、面向对象、安全、跨平台、多线程、健壮、分布式等

# JAVA基础知识

## JDK目录

bin:存放了Java的操作工具，比如编译工具javac,启动JVM的java等。

db:存放了Java测试的数据库derby，企业不用。

include：存储C++的头文件。

jre:Java的运行环境，里面有JVM。

lib:Java运行和以来的核心库。

src.zip：Java的源代码。

## 环境变量

JAVA\_HOME:指定JDK的安装根目录

D:\java\jdk1.8.0

PATH:指定java开发工具的位置

D:\java\jdk1.8.0\bin

CLASSPATH:指定jvm在运行时去哪一个目录路径去加载字节码文件

CLASSPATH顾名思义就是class文件的路径，表示JVM从哪里去寻找class文件。

CLASSPATH = c:\test就表示执行java命令时去c:\test目录中去找需要被执行的class文件。

在java5之前CLASSPATH没有默认值，要是设置成当前路径的话，得使用“.”

如CLASSPATH=.;c:\test：就表示先在当前目录找class文件，要是没找到就去c:\test目录找(当一个变量有多个值时，多个值之间用英文的分号隔开)。

从Java5开始CLASSPATH默认就是当前路径，一般情况下不需再指定。

若使用Java5以前的版本，设置CLASSPATH应该是：

.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar

dt.jar是关于运行环境的类库，主要是swing的包；tools.jar是工具类库；

后来SUN公司改进了JDK设计，JRE会自动搜索当前路径下的jar包，并自动加载dt.jar和tools.jar。那么从Java5开始，再也不必为设置CLASSPATH变量感到头疼了。

如何把编译出来的class文件存放到指定的目录:

使用javac -d C:\myclasses Hello.java

## 基本语法

1):Java语言严格区分大小写，好比main和Main是完全不同的概念。

2):一个Java源文件里可以定义多个Java类，但其中最多只能有一个类被定义成public类。若源文件中包括了public类，源文件必须和该public类同名。

3):一个源文件中包含N个Java类时，成功编译后会生成N份字节码文件，即每个类都会生成一份单独的class文件，且字节码文件名和其对应的类名相同。

4):若一个类必须运行，则必须拥有main方法,**因为main方法是程序的入口.**

注释

Java的注释信息是给程序员看的，编译器(javac)在编译的时候会忽略掉源文件中的注释信息。

为此，Java提供3种注释类型：

①　单行注释；// ； //注释信息，从//开始到本行结束的所有字符会被编译器忽略;

②　多行注释；/\* \*/；/\* 注释信息 \*/之间的所有字符会被编译器忽略

③　文档注释；/\*\* \*/；/\*\* 注释信息 \*/和多行注释一样，除此之外还可以专门生成文档信息API。

注意:多行注释和文档注释之间彼此都不能交叉嵌套.

关键字和保留字

关键字:在编程语言中有一些事先定义的，有着特殊含义和用途的单词。

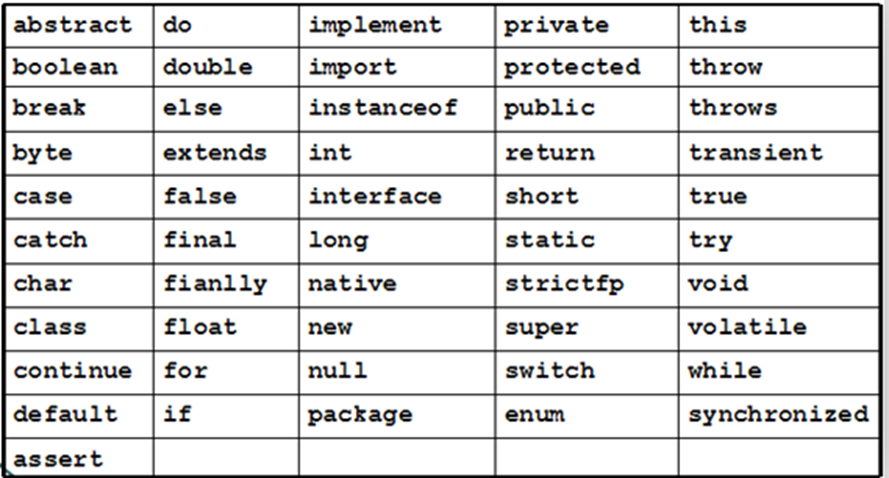
保留字:和关键字一样都是编程语言事先定义好的，只是说现在暂时没有特殊的用途，但说不定以后某天会突然被赋予意义和被使用到，因此被保留下来的单词，goto和const就是Java的保留字。

注意关键字和保留字都是由小写组成，关键字不要去记，我们学一个记一个。

注意：java 无sizeof 、goto、 const 关键字

----------------------------------------

有人认为，true,false，null.属于字面量，直接量，其实无所谓，我们不纠结这些学术性。



Java语言分隔符:

分号（;）：语句的分割，表示一句话结束，好比咱们使用的句号。

花括号（{}）：表示一个代码块，是一个整体，花括号要成对使用。

方括号（[]）：定义数组和访问数组元素时使用。

圆括号（()）：使用很广泛，具体用到细讲。

圆点（.）：类和对象访问它的成员时使用。

空格（ ）：把一整条语句分割成几段，空格的次数不限制，好比一句英文里单词都要分开写一样。

注意：必须都是半角下的英文符号。；ab;ａｂ

标识符:

在写代码的时候为了增强代码的阅读性会自定义很多名字，比如：类名，方法名，变量名等。

在编程的里我们把这种为了增强程序阅读性而自定义的名称,称为标识符。

标识符命名规则：

①　由字母、数字、下划线、$组成，但不能以数字开头（注：此处的字母可以是中文、日文等）。

②　大小写敏感。

③　不得使用java中的关键字和保留字。

④　不用Java中内置的类名作为自己的类名。

## 常量

常量概述

在程序执行的过程中其值不可以发生改变

Java中常量分类

字面值常量

字符串常量： 用双引号括起来的内容

整数常量： 所有整数 12,23

小数常量 所有小数 12.34,56.78

字符常量 用单引号括起来的内容 ‘a’,’A’,’0’

布尔常量 较为特有，只有true和false

空常量 null

自定义常量

Class定义常量方法（推荐方法）

采用“类.常量名”方法调用。需要私有化构造方法，避免创建该类的实例。同时不需让其他类继承该类。

如果多处要访问工具类中定义的常量，可通过静态导入（static import）机制，避免用类名来修饰常量名

public final class Constants {

//私有构造方法

private Constants() {}

public static final int ConstantA = 100;

public static final int ConstantB = 100;

......

}

Interface定义常量方法

在interface中声明的字段，虚拟机在编译时自动加上public static final修饰符。使用方法一般是“接口.常量名”。也可以通过实现该接口，直接访问常量名，即常量接口模式。

常量接口：即接口中不包含任何方法，只包含静态的final域，每个域都导出一个常量。使用这些常量的类实现这个接口，以避免用类名来修饰常量名。

public interface Constants {

int ConstantA = 100;

int ConstantB = 100;

......

}

区别

上述两种方法对比，interface中定义常量方法生成的class文件比第一种方法的class文件更小, 且代码更简洁, 效率更高.

但是在java中会产生问题，主要是java的动态性，java中一些字段的引用可以在运行期动态进行。某些场景下，部分内容改变可只进行部分编译。

推荐使用Class定义常量，采用private修饰符，通过get方法获取常量。这种方案可以保证java的动态性。

public class A{

private static final String name = "bright";

public static String getName(){

return name;

}

常量相加与变量相加的区别

/\*

面试题： byte b1=3,b2=4,b;

b=b1+b2;

b=3+4;

哪句是编译失败的呢？为什么呢？

b = b1 + b2;是有问题的。

因为变量相加，会首先看类型问题，最终把结果赋值的也会考虑类型问题。

常量相加：在编译成class先把结果计算出来，然后看是否在byte的范围内，如果在就不报错。

\*/

class DataTypeDemo6 {

public static void main(String[] args) {

//定义了三个byte类型的变量，b1，b2，b3

//b1的值是3，b2的值是4，b没有值

byte b1 = 3,b2 = 4,b;

//b = b1 + b2; //这个是类型提升为了int类型，所以有问题

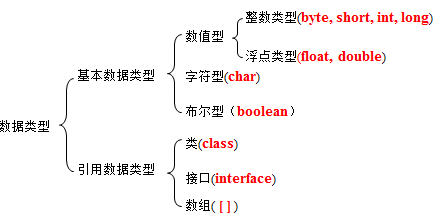
b = 3 + 4; //反编译class文件查看到是b=7;也就是它把常量结果先计算好编译成class文件

}

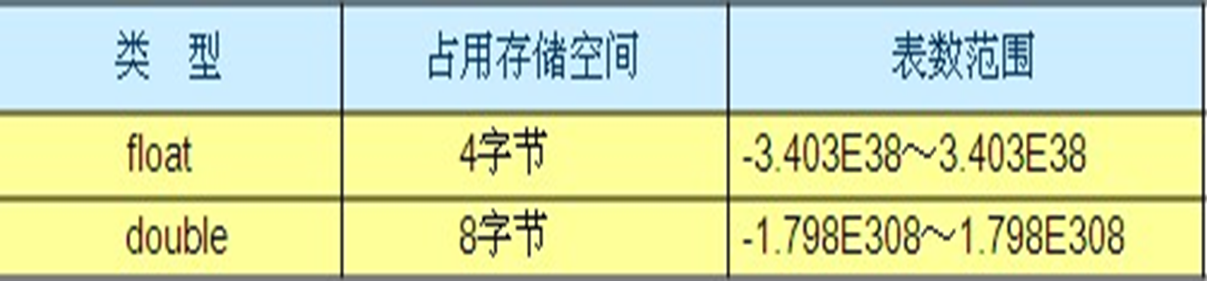
}

## 数据类型

Java语言是强类型语言，对于每种数据都定义明确的具体数据类型，在内存总分配了不同大小的内存空间







### INT

Java整型常量默认为int型，声明long型常量可以后加小写 l 或 大写L

int a= 600; //正确 long a= 88888888888L; //必须加 L否则会出错

面试题：byte b1=3,b2=4,b;

b=b1+b2;

b=3+4;

哪句是编译失败的呢？为什么呢？

答案：b = b1 + b2;是有问题的。因为变量相加，会首先看类型问题，最终把结果赋值的也会考虑类型问题。

b=3+4;常量编译时先相加经反编译看到：b=7,jvm编译时帮你先进行相加，然后判断是否在byte的范围内，不在就报错。

### double

Java 浮点型常量默认为 double 型，要声明一个常量为 float 型，则需在数字后面加 f 或 F

double d = 12345.6; //正确 float f = 12.3f; //必须加f否则会出错

面试题 看看下面两个定义有没有区别呢?

float f1 = (float)12.345;

float f2 = 12.345f;

答案

f1其实是通过一个double类型转换过来的，强制类型如果不在float范围内会损失精度造成不一样的结果。

而f2本身就是一个float类型。

### char

char 型数据用来表示通常意义上的“字符”

字符常量为用单引号括起来的单个字符，例如：char ch1= 'a'; char ch2='中';

Java字符采用 Unicode 编码，每个字符占两个字节，因而可用十六进制编码形式表示。注：Unicode是全球语言统一编码

字符参与运算是查找ASCII里面的值，记住：a在ASCII中是97，A是65，0是48

通过字符和一个整数相加，我们给出一张表：ASCII码表。

通过看完这张表以后，我们要记住三个值： 'a':97,'A':65,'0':48

class DataType {

public static void main(String[] args) {

//直接输出一个字符

System.out.println('a'); //a

//输出一个字符和一个整数做加法

System.out.println('a'+1); //98

}

类型转换

注意： boolean类型不能转换为其他的数据类型

默认转换(从小到大的转换)

A:byte,short,char：—>int—>long—>float—>double

B:byte,short,char相互之间不转换，他们参与运算默认转换为int类型

强制类型转换

从大的数据类型到小的数据类型。

格式：目标数据类型 变量 = (目标数据类型) (被转换的数据);

注意：不要随意的去使用强制转换，因为它隐含了精度损失问题。

例子：

public static void main(String[] args) {

byte a = 3;

int b = 4;

int c1 = a + b; //这个肯定没有问题(默认转换为int类型)

byte c = a + b; //错误：因为a和b是变量，因为变量的值会变化，不确定具体的值，所以默认使用int类型进行存储

byte c = (byte) (a + b); //用强制类型转换改进

System.out.println(c);

}

上面例子内存中是如何转换的呢：请看图



## 进制

Java针对整数常量提供了4种表现形式：二进制 、8进制、十进制、16进制

计算机存储单位：bit(位) ,byte(字节),KB,MB,GB,TB

bit(位): 二进制表现方式：00000000

1byte(字节)=8bit(位)

1TB=1024GB,1GB=1024MB,1MB=1024KB ,1KB=1024byte

二进制 只有0和1两个数码来表示的数。进位规则是“逢二进一”，借位规则是“借一当二”

java以0b开头都是二进制

1字节的二进制：0=00000000；1=00000001；2=00000010；3=00000011；4=00000100

十进制

1字节的十进制：0=00000000；1=00000001；2=00000002;………10=00000010

八进制 0,1,2,3,4,5,6,7,没有8和9。进位规则是“逢八进一”

java以0开头都是八进制

1字节的八进制：0=00000000，1=00000001……….. 7=00000007, 8=00000010,9=00000011

十六进制 除了0-9数字还有： a，b，c，d，e ,f分别代表：10,11,12,13,14,15

java以0x开头的都是16进制

1字节的十六进制:1=00000001,…..10=00000000a … 15=00000000f, 16=000000010, 17=0000 00011

计算机存储格式与原码、反码、补码

计算机存”都是二进制形式进行存储的，计算机只认二进制

十进制的整数1在计算机存储格式： 00000001

十进制的整数2在计算机存储格式： 00000010

负数：1字节8位数最前面(符号位)变1 就是负数,int类型4字节32位最前面（符号位）变1就是负数

符号位:即八位字符的第一位，1为负，0为正。

十进制的整数-2在计算机存储格式：10000010

这样导致了问题：在二进制 2+(-2)就不正确了：衍生了：原码、反码、补码

在计算机内，有符号数有3种表示法：原码、反码和补码,所有数据的运算都是采用补码进行的

正数：原码= 反码=补码

负数：需要最后的补码来进行操作

反码=符号位不变，其他位取反

补码=反码+1

正数5:

5的二进制是：00000101

原码：00000101

反码：00000101

补码：00000101

------------------------------------------------------

负数：-5:

5的二进制是：00000101

原码：10000101

反码：11111010

补码：11111011

二进制、八进制、十六进制转十进制：记住公式即可(很简单)

系数\*基数的权次幂然后进行相加

给个例子：十进制到十进制的转换：先拆分相加再拆分成次方公式

12345=10000+2000+300+40+5 = 1\*10^4+2\*10^3+3\*10^2+4\*10^1+5\*10^0

系数：就是每一位上的数据：如上面：123456

基数：进制 如上面的是十进制：基数就是10

圈：固定的，从右边开始编号：即上面次方的0、1、2、3

所以公式是：系数\*基数的权次幂然后进行相加

二进制转10进制：注意0b开头是java中代表二进制

0b100=1\*2^2+0\*2^1+0\*^2^0=4;使用乘法口诀相加后最后得到4，

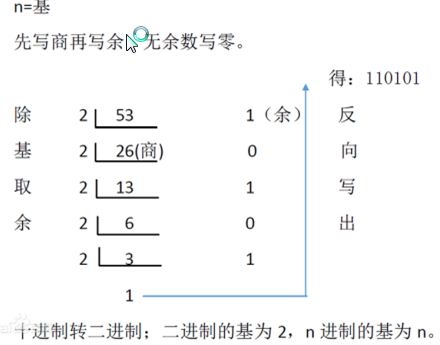
八进制转10进制：注意0开头是java中代表八进制

0100=1\*8^2+0\*8^1+0\*^8^0=64;

十六进制转10进制：注意0x开头代表16进制

0x100=1\*16^2+0\*16^1+0\*^16^0=256;

十进制转二进制、八进制、十六进制：记住公式即可(很简单)



注意问题：有可能面试问到

byte by = 130;有没有问题?有问题如何解决?结果是多少呢? byte 的数值范围是 126~-127 130已经超出byte取值范围 运算是会砍掉前面的 三个8位结果会变成-126

byte b1=3,b2=4,b;

b=b1+b2;

b=3+4;

哪句是编译失败的呢？为什么呢？

//b=b1+b2出错在JAVA虚拟机中默认int类型转换，虚拟机不知道b1,b2中的数值是多少，所有会抛出可能损失精度的错误;

虚拟机会把 b1,b2 转换成int类型在进行运算，两个int类型的数值相加结果任然是int类型，int类型不能赋值给 byte类型

byte b = 10;

b++;//这句 java虚拟机会做一个自动转换动作相当于b=(byte)(b+1)所以不会报错

b = b + 1;//这句java虚拟机会把，b转换成int类型 相当于 int b + int 类型的1 然后赋值给Byte类型 int是4个字节 byte是1个字节 所有会抛出损失精度会报错需要强制类型转换

## 位运算符

位运算符：&,|,^,~,<<,>>,>>>7种

分析：因为是位运算，所以我们必须先把数据换算成二进制。

第一步 数字3和4在内存中二进制数据(java默认int类型)

3的二进制：11，转换为int类型4字节：00000000 00000000 00000000 00000011

4的二进制：100，转换为int类型4字节：00000000 00000000 00000000 00000100

第二步 使用为运算符的操作结果

(1) <<:左移 左边最高位丢弃，右边补齐0

//<< 把<<左边的数据乘以2的移动次幂

System.out.println(3 << 2); //3\*2^2 = 3\*4 = 12; 下面是运算原理：

注意：如果是负数需要转换为补码(正数是原码=反码=补码)来进行操作，操作完后再转换为原码

00000000 00000000 00000000 00000011 二进制 3

(00)000000 00000000 00000000 0000001100 进行左移后的二进制结果12

(2) >>:右移 最高位（符号位）是0，左边补齐0；最高为是1，左边补齐1

//>> 把>>左边的数据除以2的移动次幂

System.out.println(24 >> 2); //24 / 2^2 = 24 / 4 = 6 原理几乎和左移一样。3>>2值为0

(3) >>>:无符号右移 无论最高位是0还是1，左边补齐0

System.out.println(24 >>> 2);结果是6 原理几乎和左移一样

System.out.println(-24 >>> 2);结果是1073741818

(4)&位与运算：有0则0。

System.out.println(3 & 4);//结果为0，下面是运算原理：

00000000 00000000 00000000 00000011 二进制3

00000000 00000000 00000000 00000100 二进制4

&------------------------------------------------------- 两个运算只要一个有0最后就变为0

00000000 00000000 00000000 00000000结果是：0

(5) |位或运算：有1则1。

System.out.println(3 | 4);//结果为7,下面是运算原理：

00000000 00000000 00000000 00000011 二进制3

00000000 00000000 00000000 00000100 二进制4

|------------------------------------------------------- 两个运算只要一个有1最后就变为1

00000000 00000000 00000000 00000111 //折算为十进制为数字7

(6) ^位异或运算：相同则0，不同则1。

System.out.println(3 ^ 4); //结果为7,下面是运算原理：

00000000 00000000 00000000 00000011 二进制3

00000000 00000000 00000000 00000100 二进制4

^------------------------------------------------------- 两个运算只要相同为0，不同是1

00000000 00000000 00000000 00000111 //折算为十进制为数字7

^还有一个特点：一个数据对另一个数据位异或两次，该数本身不变。

int a = 10;int b = 20;

System.out.println(a ^ b ^ b); //结果等于10

System.out.println(a ^ b ^ a); //结果等于20

(7) ~按位取反运算符：0变1，1变0。

System.out.println(~3); 结果是：-4

00000000 00000000 00000000 00000011

~11111111 11111111 11111111 11111100 (补码) 取反后获得是补码

补码：11111111 11111111 11111111 11111100

反码：11111111 11111111 11111111 11111011 反码=补码-1

原码：10000000 00000000 00000000 00000100 原码=反码的符号位不变其它取反。所以最后结果是-4

# 类、对象和接口

## 类与对象

类：可以理解为构造对象的一个蓝图或者模版，是抽象的概念

对象：是以类为模型创建的具体实例，是对类的一种具体化

创建对象：

普通创建

类名 对象名 = new 类名();

对象名.成员变量

对象名.成员方法

匿名对象：就是没有名字的对象。

是对象的一种简化表示形式 ： new 类名();

匿名对象的两种使用情况

对象调用方法仅仅一次的时候

作为实际参数传递

构造方法

构造方法作用概述

给对象的数据进行初始化

构造方法格式

方法名与类名相同

没有返回值类型，连void都没有

没有具体的返回值

构造方法注意事项

如果你不提供构造方法，系统会给出默认构造方法

如果你提供了构造方法，系统将不再提供

构造方法也是可以重载的

### static关键字

可以修饰成员变量和成员方法

static关键字特点

随着类的加载而加载

优先于对象存在

被类的所有对象共享

这也是我们判断是否使用静态关键字的条件

可以通过类名调用

static关键字注意事项

在静态方法中是没有this关键字的

静态方法只能访问静态的成员变量和静态的成员方法

静态代码块

在类中方法外出现，加了static修饰

在类中方法外出现，并加上static修饰；用于给类进行初始化，在加载的时候就执行，并且值执行一次。

静态变量和成员变量的区别

1、所属不同

静态变量属于类，所以也称为为类变量

成员变量属于对象，所以也称为实例变量(对象变量)

2、内存中位置不同

静态变量存储于方法区的静态区

成员变量存储于堆内存

3、内存出现时间不同

静态变量随着类的加载而加载，随着类的消失而消失

成员变量随着对象的创建而存在，随着对象的消失而消失

4、调用不同

静态变量可以通过类名调用，也可以通过对象调用

成员变量只能通过对象名调用

main方法是静态的

public static void main(String[] args) {}

public 被jvm调用，访问权限足够大。

static 被jvm调用，不用创建对象，直接类名访问

void被jvm调用，不需要给jvm返回值

main 一个通用的名称，虽然不是关键字，但是被jvm识别

String[] args 以前用于接收键盘录入的

### this关键字

this:代表所在类的对象引用；记住：方法被哪个对象调用，this就代表那个对象

什么时候使用this呢?

局部变量隐藏成员变量：局部变量和成员变量相同的变量名称

### 成员变量和局部变量的区别

1、在类中的位置不同

成员变量 类中方法外

局部变量 方法内或者方法声明上

2在内存中的位置不同

成员变量 堆内存

局部变量 栈内存

3、生命周期不同

成员变量 随着对象的存在而存在，随着对象的消失而消失

局部变量 随着方法的调用而存在，随着方法的调用完毕而消失

4、初始化值不同

成员变量 有默认的初始化值

局部变量 没有默认的初始化值，必须先定义，赋值，才能使用

### 初始化对象

类(无继承和实现接口)的初始化过程

Student s = new Student();在内存中做了哪些事情?

第一步:加载Student.class文件进内存

第二步:在栈内存为s开辟空间

第三步:在堆内存为学生对象开辟空间

第四步:对Student对象的成员变量进行默认初始化 ---- 如基本数据int默认是0

第五步:对Student对象的成员变量进行显示初始化 ---- 如成员变量中直接赋值： int i=5；

第六步:通过构造方法对Student对象的成员变量赋值

第七步: Student对象初始化完毕，把对象地址赋值给s变量

注意：一个类初始化构造方法都是最后执行的,所以一般都用构造方法为成员变量赋值

### 继承

Java只支持单继承，不支持多继承，支持多层继承(继承体系)

继承的好处

提高了代码的复用性

多个类相同的成员可以放到同一个类中

提高了代码的维护性

如果功能的代码需要修改，修改一处即可

让类与类之间产生了关系，是多态的前提

其实这也是继承的一个弊端：类的耦合性很强

Java中继承的注意事项

子类只能继承父类所有非私有的成员(成员方法和成员变量)

1其实这也体现了继承的另一个弊端：打破了封装性

2子类不能继承父类的构造方法，但是可以通过super关键字去访问父类构造方法。

3不要为了部分功能而去继承，我们到底在什么时候使用继承呢?

答：继承中类之间体现的是：”is a”的关系。

#### 继承中构造方法的关系

子类中所有的构造方法默认都会访问父类中空参数的构造方法

为什么呢?

因为子类会继承父类中的数据，可能还会使用父类的数据。所以，子类初始化之前，一定要先完成父类数据的初始化。

每一个构造方法的第一条语句默认都是：super()

#### super关键字

super的用法和this很像

this代表本类对应的引用。

super代表父类存储空间的标识(可以理解为父类引用)

用法(this和super均可如下使用)

访问成员变量

this.成员变量 super.成员变量

访问构造方法

this(…) super(…)

访问成员方法

this.成员方法() super.成员方法()

#### 继承类的初始化过程

子父类的初始化(分层初始化)

先进行父类初始化，然后进行子类初始化。

例子：A类继承B类，A类是中main方法进行初始化A类

class B {

C c = new C(); //初始化第二步：先初始化成员变量：new了一个C类，进入C类进行初始化

B() {System.out.print("B");}} //初始化第四步：初始化完成员变量：构造方法执行打印：B，然后进入子类A进行初始化

class C {

C() {System.out.print("C");}} //初始化第三步：C类没有成员变量进行构造方法初始化打印：C，C类初始完毕进入B

//调用操作：A类继承B类

public class A extends B { //初始化第一步：看到有继承类B，先进行初始化B类

C c = new C(); //初始化第五步：A类初始化成员变量：C类—>进入C类进行初始化成员变量和构造：打印C

A() { //初始化第六步：A类初始化完成员变量进行构造方法打印A

super();注意：虽然子类中构造方法默认有super()，初始化的时候，不是按代码从上至下顺序进行的。

System.out.print("A");

}

public static void main(String[] args) {

new A();//c,b,c,a

}

//最后打印：CBCA

### 封装

封装概述

是指隐藏对象的属性和实现细节，仅对外提供公共访问方式。

好处：

隐藏实现细节，提供公共的访问方式

提高了代码的复用性

提高安全性。

封装原则：

将不需要对外提供的内容都隐藏起来。

把属性隐藏，提供公共方法对其访问。

### 多态