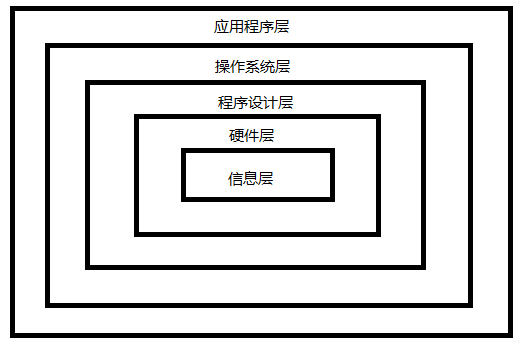
# AP 计算机科学—java

## 计算机科学导论

### 计算机全景图

#### 1.1.1 计算机系统的分层



其中，信息层是纯概念层，也是计算机的原理层。在这个层面上，中国的祖先早就看透千年后的计算机的原理了。世界是一个太极，太极分阴阳，阴阳生四象，四象生八卦，而后生万物。万物的一切存在，生长与灭亡，都离不开这个。

而计算机同样也是，计算机中所有的一切都是数字，信息是数字，图片是数字，视频也是数字，而后计算机中的数字是有0和1构成的，因为计算机在真与假的判断上，是能做到极致的，除此之外，对于计算机硬件的实现要求更简单。非真即假，和中国的非阴即阳是同样的道理。二进制对于计算机来说，很好分别，但是对人类来说就很困难了，因此，为了让人们更容易看，于是有了八进制，16进制等，和中国的八卦差不多。

而后硬件层，硬件包括各种电路，包括CPU和存储器等，都是为了实现信息的存储和处理，也就是实现二进制的存储和处理。

程序设计层，负责实现计算机软件，通过程序操作计算机指令，从而控制计算机的硬件。

操作系统层，是由程序设计出来的专门用于管理计算机硬件的软件。

#### 1.1.2 抽象的概念

抽象，是一种心理模型，即删除了复杂的实现细节，只保留必要的信息。比如，开车去商店买东西，就是一种抽象，指保留了开车的信息，但是车具体运行的细节以及实现原理，并没有提及。

### 计算机原理层

#### 二进制数值与记数系统

0和1是数字，但是数字不是0和1。与货币不是金银，但金银天然是货币是一个道理。

##### 1.2.1 位置计数法

规定位置的价值。比如十进制的943，表示了多少呢？

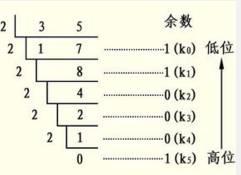
943 = 9 \* 102 + 4 \* 101 + 3\*100

##### 二进制，八进制，十六进制以及十进制之间的相互转换：

二进制，0，1 组成。逢2进1

八进制，0，1，2，3，4，5，6，7 组成。

十六进制，0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。



35 = 100011

#### 数据表示法

### 1.3 计算机硬件层

### 1.4 计算机程序设计层

### 1.5 计算机操作系统层

### 1.6 计算机应用程序层

### 1.7 计算机网络层

### 1.8 计算机与人工智能

## 基本的编程结构

### 2.1 计算机概述

#### 2.1.1 计算机概述

##### 2.1.1.1 计算机硬件

计算机通常由CPU、主板、内存、电源、主机箱、硬盘、显卡、键盘、鼠标，显示器等多个部件组成。

##### 2.1.1.2 计算机软件

系统软件： DOS(Disk Operating System), Windows, Linux, Unix, Mac, Android, iOS

应用软件：office QQ聊天 YY语言 扫雷

#### 2.1.2 键盘常用功能键的任何和快捷键

##### 2.1.2.1 键盘常用的功能键

Tab Shift Ctrl Alt windows 回车（enter）

##### 2.1.2.2 键盘快捷键

Ctrl+A 全选 Ctrl+C 复制 Ctrl+V 粘贴

Ctrl+X 剪切 Ctrl+Z 撤销 Ctrl+S 保存

#### 2.1.3 常见的DOS命令

##### 2.1.3.1 打开DOS窗口

Win + R

输入cmd 回车

##### 2.1.3.2 常用DOS命令

d: 回车 盘符切换

dir(directory):列出当前目录下的文件以及文件夹

md (make directory) : 创建目录

rd (remove directory): 删除目录

cd (change directory)改变指定目录(进入指定目录)

cd.. : 退回到上一级目录

cd \: 退回到根目录

del (delete): 删除文件,删除一堆后缀名一样的文件\*.txt

exit : 退出dos命令行

cls : (clear screen)清屏

### 2.2 JAVA语言概述

#### 2.2.1 JAVA语言概述

##### 2.2.1.1 Java平台版本

Java SE：开发普通桌面和商务应用程序

Java ME：开发电子消费产品以及嵌入式设备

Java EE：为开发企业环境下的应用程序，主要解决Web应用程序开发

##### 2.2.1.2 Java语言特点

简单性 解释性

面向对象 高性能

分布式处理 多线程

健壮性 动态

结构中立 安全性

开源

跨平台

#### 2.2.2 JDK，JRE，JVM的作用以及关系

1. JVM -- java virtual machine

2. JRE -- java runtime environment

3. JDK -- java development kit

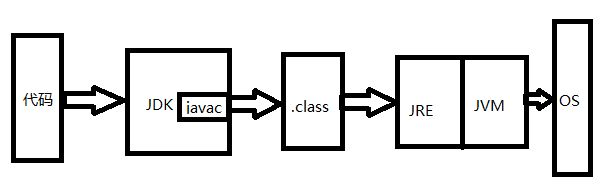
从英文意思可以看出，JVM是java的虚拟机，JRE是java的运行环境，JDK是java的开发工具包。

其中，如果只是运行java程序，只需要安装JRE即可，而JVM是JRE的组成部分，它有完善的硬件架构。而JDK除了包含JRE之外，还包含开发java程序所必须的类库以及工具包。

JDK = JRE + 工具

JRE = JVM + 类库

我们开发的实际情况是：我们利用JDK（调用JAVA API）开发了属于我们自己的JAVA程序后，通过JDK中的编译程序（javac）将我们的文本java文件编译成JAVA字节码，在JRE上运行这些JAVA字节码，JVM解析这些字节码，映射到CPU指令集或OS的系统调用。



#### 2.2.3 JDK的下载以及安装

##### 2.2.3.1 JDK的下载

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

##### 2.2.3.2 JDK环境变量的配置

JAVA\_HOME = JDK的安装路径，

CLASSPATH = java的加载类，%JAVA\_HOME%/lib/dt.jar;%JAVA\_HOME%/lib/tools.jar

与import和package有关。

Path = ;%JAVA\_HOME%/bin; 使得系统可以在任何路径下识别java命令，同理可以配置QQ等

#### 2.2.4 第一个程序：Hello World的案例

public class Demo01{

public static void main(String[] args){

System.out.println("hello world");

}

}

1. 在java程序中最基本的单位是类
2. 要想执行java程序，必须有程序入口，即main函数。

程序执行流程：

1. 编写java源代码 notepad++
2. 通过javac对java程序进行编译，javac Demo01.java （带扩展名）
3. 通过java命令，运行.class文件，java Demo01（不带扩展名）

常见问题：

（1） 扩展名隐藏问题

（2） java编程规范，一个文件只有一个被public修饰的类，且类名和文件名相同。

（3） java语言严格区分大小写。因此注意单词不要写错

（4） 括号都是成对出现

（5） 找不到主方法，请查看主方法是否写正确。

#### 2.2.5 练习

1. 快捷键的练习

2. 环境变量的练习

3. helloworld程序

4. 理解JDK，JRE，JVM的概念

5. 了解计算机基本层次结构

### 2.3 java语言基础

#### 2.3.1 注释

##### 2.3.1.1 注释

注释，就是对程序进行解释说明的文字

分类：

A:单行注释 //

B:多行注释 /\*\*/

C:文档注释(后面讲) /\*\* \*/

作用：

解释说明程序，同时也会帮助我们调试程序(注解)

##### 2.3.1.2 练习

/\*\*

\*这是一个HelloWord示例

\*/

public class Demo01{

//这是程序入口——main函数

public static void main(String[] args){

//这是打印语句，将打印出hello world

System.out.println("hello world");

}

}

#### 2.3.2 标识符与关键字

##### 2.3.2.1 标识符

###### Java标识符命名规则：

1） 标识符由字母、数字、下划线“\_”、美元符号“$”或者人民币符号“￥”组成，并且首字母不能是数字。

2） 不能把关键字和保留字作为标识符。

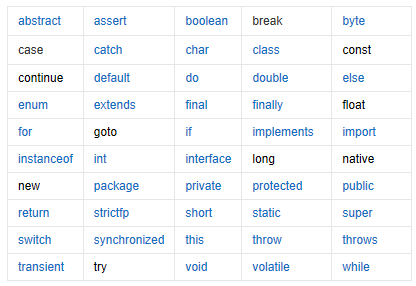
3） 标识符没有长度限制。

4） 标识符对大小写敏感。

###### Java标识符命名规范：

|  |  |  |
| --- | --- | --- |
| Java命名规范 | 示例 | 说明 |
| 包的命名 | scnu.cn.edu.controllor | 全部小写，由域名组成 |
| 类的命名 | GraphicsTest | 单词首字母大写 |
| 方法的命名 | drawImage | 首字母小写，其后单词首字母大写 |
| 常量的命名 | MAX\_VALUE | 全部大写，下划线分词 |
| 参数命名 | fValue | 同方法的命名 |
| 变量命名 | s\_sentence或者strSentence（常用） | 匈牙利命名，类型\_变量名|类型变量名 |

##### 2.3.2.2 关键字



#### 2.3.3 常量与变量

##### 2.3.3.1 常量

字面值常量:

(1)字面值常量

A:字符串常量 "hello"

B:整数常量 12,23

C:小数常量 12.345

D:字符常量 'a','A','0'

E:布尔常量 true,false

F:空常量 null(后面讲)

(2)在Java中针对整数常量提供了四种表现形式

A:二进制 由0，1组成。以0b开头。

B:八进制 由0，1，...7组成。以0开头。

C:十进制 由0，1，...9组成。整数默认是十进制。

D:十六进制 由0，1，...9,a,b,c,d,e,f(大小写均可)组成。以0x开头。

自定义常量（后面）

2.3.3.2 变量

在程序执行过程中，其值在某个范围内可以改变的量。

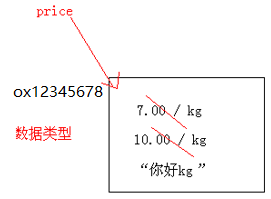
变量的组成规则：

1. 首先要对其限定。——数据类型

2. 其次需要对这个空间起个名字，不能拿着内存地址去运算，太复杂。——变量名

3. 即使有了数据类型，有变量名了，但是内存里是空的，也没法运算。——需要初

始化。



变量的定义格式：

A:数据类型 变量名 = 初始化值;

B:数据类型 变量名;

变量名 = 初始化值;

示例：

int a = 10;

int b;

b=20;

int c=30,d=40;

System.out.println(a);

System.out.println(b);

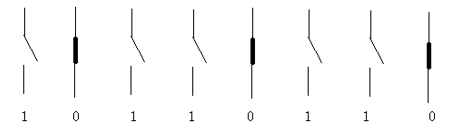
System.out.println(c);

System.out.println(d);

#### 2.3.4 进制转换

进制，是一种进位的方式。X进制，表示逢x进1。

计算机的电子原件的状态，开，关。那么我们在表达数据的时候，也是按照开，关的状态来表示的。



如果我们表达数据仅仅用着两种状态，那么能够表达的数据是比较少的。而我们常见的数据，英文字母，数字，标点符号，这就很多了。因此两种状态是不够了。

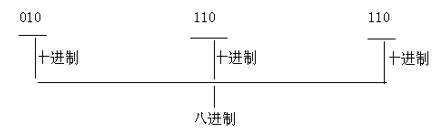
为了表示更多的数据，国际化标准组织就规定：用8个这样的信号来表示一个数据。而这个数据的单位，我们叫做字节（B）。

由1，0组成的数据就是二进制数据。

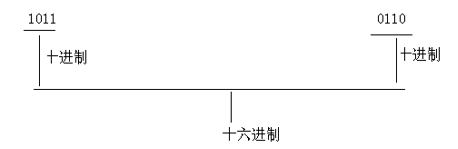
1Byte = 8bit，1KB = 1024Byte，1MB = 1024KB，1GB = 1024 MB，1TB = 1024GB

这样表示之后，对于计算机来说，是可以的，但是对人来说表示形式有点长，而且容易有密集恐惧症。

因此，把二进制的数据，从右开始，每三位一组，最左边不够，用0补齐。人后计算其十进制的数值，然后将其组合起来。这就是八进制了。



这种形式，还不够简洁，那么四位一组，就是十六进制了。



从上面可以看出：进制越大，表现形式越短。

##### 2.3.4.1 其他进制到十进制 （位置记数法）

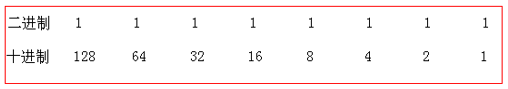
系数：就是每一个位上的数值

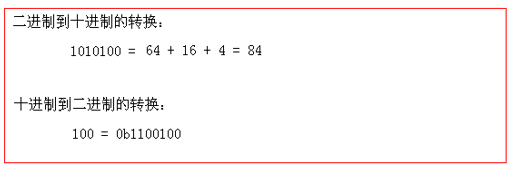
基数：x进制的基数就是x

权：对每一个位上的数据，从右，并且从0开始编号，对应的编号就是该数据的权。

结果：系数\*基数^权次幂之和。 943 = 9\*102+4\*101+3\*100

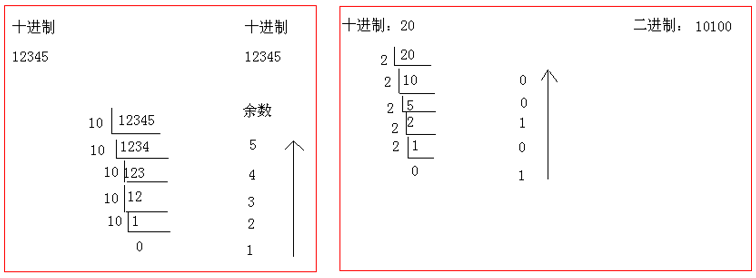
8421码，是BCD码的一种。其表达的意思就是每一个二进制位上的数据对应一个固定的值。只需要把二进制数据中为1 的位置对应的数值相加即可。





##### 2.3.4.2 十进制到其他进制

除基取余，直到商为0，余数反转。

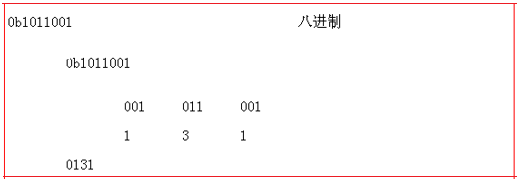


##### 2.3.4.3 进制转换的快速转换法

A:十进制和二进制间的转换

8421码。

B:二进制到八进制，十六进制的转换



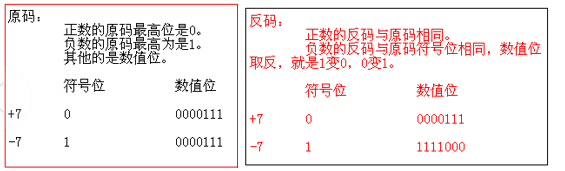
##### 2.3.4.4 数据表示法

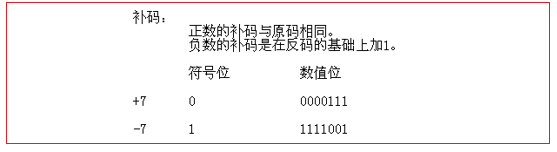
有符号的数据表示法：原码，反码，补码。

在计算机执行操作的时候，事实上都是采用数据对应的补码来进行操作的。

因为只有补码，有唯一的 0 。

下面以7为例，其二进制为 111.

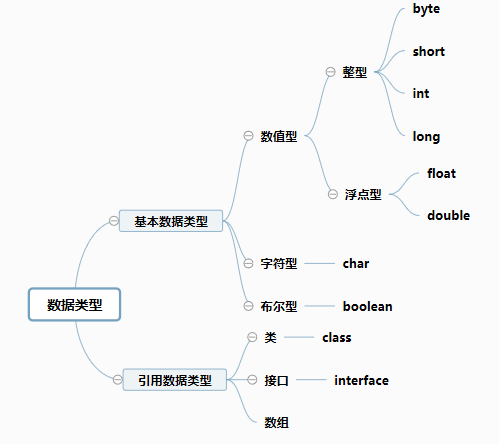




从上面可以看出，0 的原码有两种 + 0，- 0；反码同样也有两种。而只有补码0是唯一的。

#### 2.3.5 数据类型以及数据类型的转换

##### 2.3.5.1 数据类型



注意：

整数默认是int类型，浮点数默认是double。

长整数要加L或者l。

单精度的浮点数要加F或者f。

char，字符型，是可以存储汉字的，因为java采用unicode编码，char实际占用两个字节。

##### 2.3.5.2 数据类型的空间占用

|  |  |
| --- | --- |
| 类型 | 占内存大小 |
| byte | 1B |
| short | 2B |
| int | 4B |
| long | 8B |
| float | 4B |
| double | 8B |

##### 2.3.5.3 数据类型的转换

###### 2.3.5.3.1 默认转换

boolean类型不参与转换

A:从小到大

B:byte,short,char -- int -- long -- float -- double

C:byte,short,char之间不相互转换，直接转成int类型参与运算。

示例：

double e = a;

byte f = 20;

a = f;

System.out.println(e); //10.0

System.out.println(f); //20

System.out.println(a); //20

int g = 'a';

System.out.println(g); //97

// char h = f;

// System.out.println(h); //不兼容类型

###### 2.3.5.3.2 强制转换

A:从大到小

B:可能会有精度的损失，一般不建议这样使用。

C:格式：

目标数据类型 变量名 = (目标数据类型) (被转换的数据);

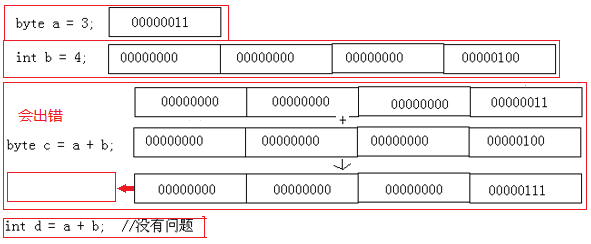
示例：

double a1 = 12.3;

a = (int)a1;

System.out.println(a); //12

总结:



###### 2.3.5.3 思考题和面试题：

A:下面两种方式有区别吗?

float f1 = 12.345f; //这里12.345本身就是float类型

float f2 = (float)12.345; //这里的12.345是int类型，强转成float类型。可能会损失精度，但是不会报错。

因此，上面两种定义，结果都是12.345，但是推荐使用第一种，因为，第二种是转换而来，可能会损失精度。

float f3 = 12.345; //这里会报错，因为i12.345是double类型。从大到小的类型转换需要强制转换。

B:下面的程序有问题吗，如果有，在哪里呢?

byte b1 = 3;

byte b2 = 4;

byte b3 = b1 + b2; //这个是有问题的，因为byte在运算的时候是先将byte类型转换成int类型，再进行运算，这个过程叫做类型提升，因此b1+b2的结果是int类型。而b3是byte类型。

byte b4 = 3 + 4; //这里注意，3，4是常量，常量的计算是，先把结果计算出来，然后再看左边的数据类型，如果结果在数据类型范围内，那么就会自动转成左边的数据类型。因此，不会报错。其实，java在编译的时候已经将3+4计算出来了，因此，在运行的时候实际运行的是byte b4 = 7.

C:下面的操作结果是什么呢?

byte b = (byte)130; // - 127

分析过程：

A:首先取130的二进制：10000010 → 00000000 00000000 00000000 10000010

这个值，同时是原码，补码和反码。

B:强制转换，做截取操作，得到的是byte类型 → 10000010。

注意：这个值是补码，因为计算机是按补码运算的。

C:由补码求原码

符号位 数值位

补码 1 0000010

反码 1 0000001

原码 1 1111110 → - 127

D:字符参与运算

是查找ASCII里面的值

'a' 97

'A' 65

'0' 48

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

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

E:字符串参与运算

这里 + 其实是字符串的连接，字符串与其他类型的数据连接，最后还是字符串。

System.out.println("hello"+'a'+1); //helloa1

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

System.out.println("5+5="+5+5); //5+5=55

System.out.println(5+5+"=5+5"); //10=5+5

#### 2.3.6 练习

1：什么是注释?注释分为几种?注释有什么作用?

2：什么是关键字?有什么特点?

3：什么是标识符?由哪些部分组成?常见的命名规则有哪些?

4：什么是常量?常量的分类?字面值常量表现形式有哪些?

5：计算出十进制46对应的二进制，八进制，十六进制的值。

6：什么是变量?变量的定义格式?要使用变量需要注意什么?

7：Java中的数据类型分几类?基本数据类型有哪些?

8：数据类型转换：隐式转换；强制转换。

9. 类型转换

第一题：

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？

第二题：

byte by = 130;有没有问题?有问题如何解决?结果是多少呢?

### 2.4 流程控制

#### 2.4.1 运算符

##### 2.4.1.1 算术运算符，关系运算符，逻辑运算符

|  |  |  |  |
| --- | --- | --- | --- |
| 算术运算符 | 关系运算符 | 逻辑运算符 | 位运算符 |
| + 加法 | == 等于 | && 与（短路） | 按位与 & |
| * 减法 | != 不等于 | || 或（短路） | 按位或 | |
| * 乘法 | > 大于 | ! 非 | 按位异或 ^ |
| / 除法 | < 小于 | & 与（非短路） | 左移 << |
| % 取模 | >= 大于等于 | | 或（非短路） | 右移 >> |
| -- 自减 | <= 小于等于 |  | 无符号右移 >>> |
| ++ 自加 | expression ? true:false(三元) |  |  |

##### 2.4.1.2 算术运算符

//定义变量

int x = 3; //把3赋值给int类型的变量x

int y = 4;

System.out.println(x+y);

System.out.println(x-y);

System.out.println(x\*y);

System.out.println(x/y); //整数相除只能得到整数

//我就想得到小数，该肿么办呢?

//只需要把操作的数据中任意的一个数据变为浮点数

System.out.println(x\*1.0/y);

//%的应用

System.out.println(x%y); //得到的是余数

//自增与自减运算

//定义两个变量

int x = 3;

int y = 4;

//字符串的拼接

//System.out.println("x:"+x);

//System.out.println("y:"+y);

System.out.println("x:"+x+",y:"+y);

//单独使用

//x++;

//y--;

++x;

--y;

//System.out.println(x);

System.out.println("x:"+x+",y:"+y);

//意外的类型,常量是不可以这样做的

//System.out.println(10++);

System.out.println("-------------------");

//参与运算使用

int a = 3;

int b = 4;

//int c = a++;

//int d = b--;

int c = ++a;

int d = --b;

System.out.println("a:"+a); //4, 4

System.out.println("b:"+b); //3, 3

System.out.println("c:"+c); //3, 4

System.out.println("d:"+d); //4, 3

+ 的运用：

//加法

System.out.println(3+4);

//正号

System.out.println(+4);

System.out.println('a');

System.out.println('a'+1); //这里是加法

//字符串连接符

System.out.println("hello"+'a'+1);

System.out.println('a'+1+"hello");

练习：

++,--的练习题

第一题：

int a = 10;

int b = 10;

int c = 10;

a = b++;

c = --a;

b = ++a;

a = c--;

请分别计算出a,b,c的值

第二题：

int x = 4;

int y = (x++)+(++x)+(x\*10);

请分别计算出x,y的值

##### 2.4.1.3 比较运算符

int x = 3;

int y = 4;

int z = 3;

System.out.println(x == y);

System.out.println(x == z);

System.out.println((x+y) == (x+z));

System.out.println("------------");

System.out.println(x != y);

System.out.println(x > y);

System.out.println(x >= y);

System.out.println(x < y);

System.out.println(x <= y);

System.out.println("------------");

int a = 10;

int b = 20;

//boolean flag = (a == b);

//boolean flag = (a = b); //这个是有问题的，不兼容的类型

//System.out.println(flag);

int c = (a = b); //把b赋值给a，然后把a留下来

System.out.println(c);

##### 2.4.1.4 逻辑运算符

int a = 3;

int b = 4;

int c = 5;

//&逻辑与

System.out.println((a > b) & (a > c)); //false & false = false

System.out.println((a > b) & (a < c)); //false & true = false

System.out.println((a < b) & (a > c)); //true & false = false

System.out.println((a < b) & (a < c)); //true & true = true

System.out.println("---------------");

//|逻辑或

System.out.println((a > b) | (a > c)); //false | false = false

System.out.println((a > b) | (a < c)); //false | true = true

System.out.println((a < b) | (a > c)); //true | false = true

System.out.println((a < b) | (a < c)); //true | true = true

System.out.println("---------------");

//^逻辑异或

System.out.println((a > b) ^ (a > c)); //false ^ false = false

System.out.println((a > b) ^ (a < c)); //false ^ true = true

System.out.println((a < b) ^ (a > c)); //true ^ false = true

System.out.println((a < b) ^ (a < c)); //true ^ true = false

System.out.println("---------------");

//!逻辑非

System.out.println(!(a > b)); //!false = true

System.out.println(!(a < b)); //!true = false

System.out.println(!!(a > b)); //!!false = false

System.out.println(!!!(a > b)); //!!false = true

**&&和&的区别? 同理||和|的区别?**

**A:最终结果一样。**

**B:&&具有短路效果。左边是false，右边不执行。**

**开发中常用的逻辑运算符：**

**&&,||,!**

int a = 3;

int b = 4;

int c = 5;

//&&双与

System.out.println((a > b) && (a > c)); //false && false = false

System.out.println((a > b) && (a < c)); //false && true = false

System.out.println((a < b) && (a > c)); //true && false = false

System.out.println((a < b) && (a < c)); //true && true = true

System.out.println("----------------");

int x = 3;

int y = 4;

//boolean b1 = ((x++ == 3) & (y++ == 4));

//boolean b1 = ((x++ == 3) && (y++ == 4));

//boolean b1 = ((++x == 3) & (y++ == 4));

boolean b1 = ((++x == 3) && (y++ == 4));

System.out.println("x:"+x);

System.out.println("y:"+y);

System.out.println(b1);

##### 2.4.1.5 位运算符

int a = 3;

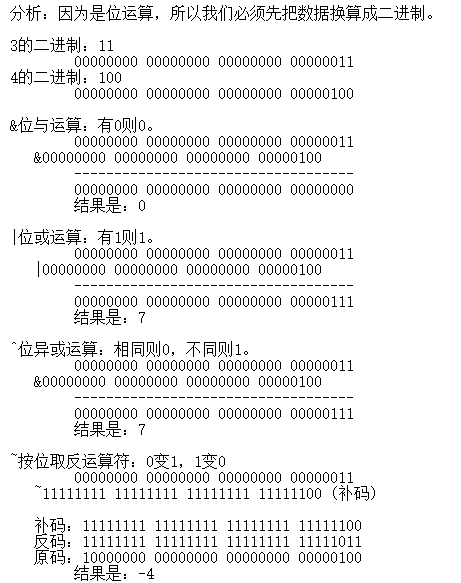
int b = 4;

System.out.println(3 & 4);

System.out.println(3 | 4);

System.out.println(3 ^ 4);

System.out.println(~3);



//<< 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;

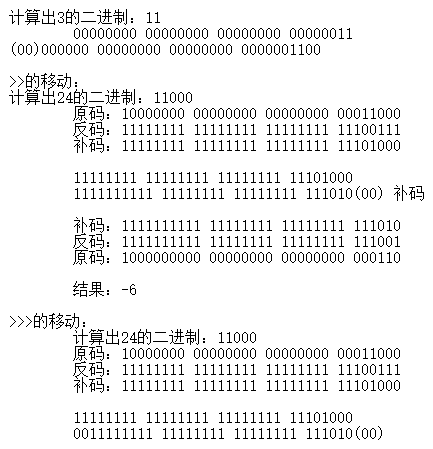
//>> 把>>左边的数据除以2的移动次幂

System.out.println(24 >> 2); //24 / 2^2 = 24 / 4 = 6

System.out.println(24 >>> 2);

System.out.println(-24 >> 2);

System.out.println(-24 >>> 2);



#### 2.4.2 键盘输入输出

2.4.2.1 键盘输入

A:导包

import java.util.Scanner;

位置：在class的上边

B:创建对象

Scanner sc = new Scanner(System.in);

C:获取数据

int x = sc.nextInt();

或者

String line = s.nextLine();

示例：

Scanner input = new Scanner(System.in);

// System.out.println("Enter a number:");

// int intValue1 = input.nextInt();

System.out.println("Enter a String:");

String strValue1 = input.nextLine();

System.out.println(strValue1);

// System.out.println(intValue1);

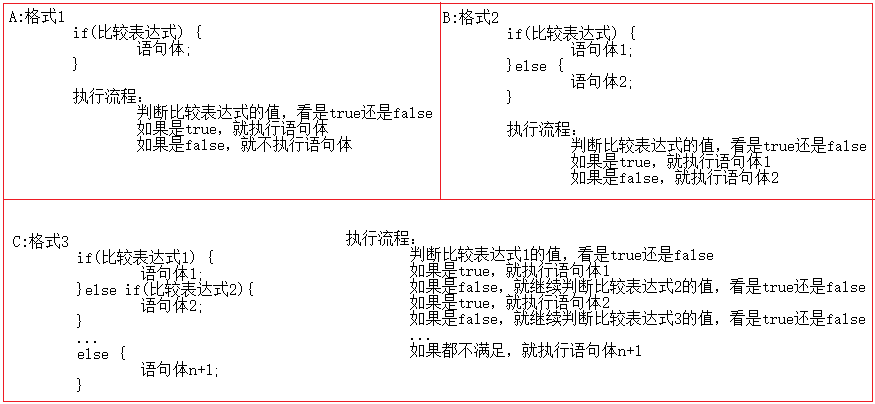
#### 2.4.3 流程控制语句

(1)顺序结构 从上往下，依次执行

(2)选择结构 按照不同的选择，执行不同的代码

(3)循环结构 做一些重复的代码

##### 2.4.3.1 if语句



###### 2.4.3.1.1 练习

看程序写结果：请自己独立分析，先不要编译运行。

第一题

int x = 1,y = 1;

if(x++==2 & ++y==2)

{

x =7;

}

System.out.println("x="+x+",y="+y);

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

第二题

int x = 1,y = 1;

if(x++==2 && ++y==2)

{

x =7;

}

System.out.println("x="+x+",y="+y);

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

第三题

int x = 1,y = 1;

if(x++==1 | ++y==1)

{

x =7;

}

System.out.println("x="+x+",y="+y);

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

第四题

int x = 1,y = 1;

if(x++==1 || ++y==1)

{

x =7;

}

System.out.println("x="+x+",y="+y);

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

第五题

boolean b = true;

if(b==false)

System.out.println("a");

else if(b)

System.out.println("b");

else if(!b)

System.out.println("c");

else

System.out.println("d");

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

13:编写代码实现如下内容：if语句实现

考试成绩分等级。

90~100 A等。

80-89 B等。

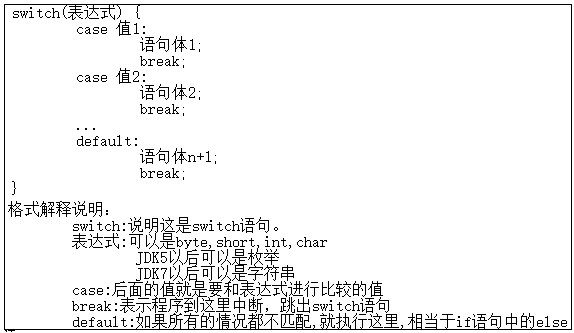
70-79 C等。

60-69 D等。

60以下 E等。

请根据给定成绩，输出对应的等级。

##### 2.4.3.2 switch语句



执行流程:

A:首先计算表达式的值

B:和每一个case进行匹配，如果有就执行对应的语句体，看到break就结束。

C:如果没有匹配，就执行default的语句体n+1。

注意事项:

A:case后面只能是常量，不能是变量，而且，多个case后面的值不能出现相同的

B:default可以省略吗?

可以省略，但是不建议，因为它的作用是对不正确的情况给出提示。

特殊情况：case就可以把值固定。

C:break可以省略吗?

可以省略，但是结果可能不是我们想要的。

会出现一个现象：case穿透。

最终我们建议不要省略

D:default一定要在最后吗?

不是，可以在任意位置。但是建议在最后。

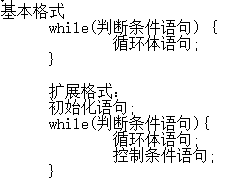
E:switch语句的结束条件

a:遇到break就结束了，如果没有break将会一直执行到结束。

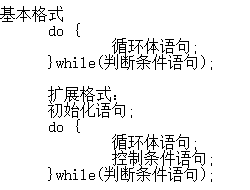
b:执行到末尾就结束了

##### 2.4.3.3 while语句

###### 2.4.3.3.1 while 语句

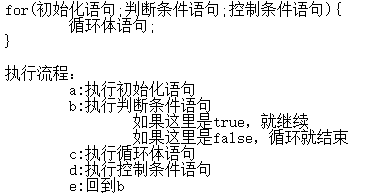


###### 2.4.3.3.2 do - while语句



##### 2.4.3.4 for语句

格式：



案例：

a:输出10次HelloWorld

b:输出1-10的数据

c:输出10-1的数据

d:求1-10的和

e:求1-100的和,求1-100的偶数和,求1-100的奇数和

f:求5的阶乘

g:在控制台打印水仙花数

水仙花数是指一个 n 位数（n≥3 ），它的每个位上的数字的 n 次幂之和等于它本身

h:统计水仙花个数

i:改进版的回文数

一个五位数

个位 = 万位

十位 = 千位

个位 + 十位 + 千位 + 万位 = 百位

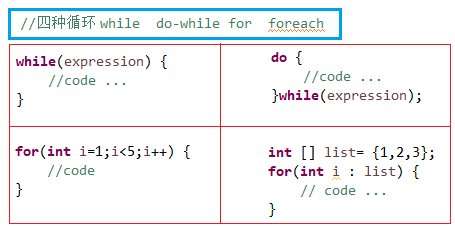
j:统计1-1000之间同时满足如下条件的数据有多少个

x%3==2

x%5==3

x%7==2

##### 2.4.3.5 四种循环语句



##### 2.4.3.6 控制跳转语句

(1)break:中断的意思

A:用在循环和switch语句中，离开此应用场景无意义。

B:作用

a:跳出单层循环

b:跳出多层循环，需要标签语句的配合

(2)continue:继续

A:用在循环中，离开此应用场景无意义。

B:作用

a:跳出单层循环的一次，可以继续下一次

C:填空题

for(int x=1; x<=10; x++) {

if(x%3 == 0) {

//补齐代码

}

System.out.println("Java基础班");

}

如何让控制台输出2次：Java基础班

如何让控制台输出7次：Java基础班

如何让控制台输出13次：Java基础班

(3)return:返回

A:用于结束方法的，后面还会在继续讲解和使用。

B:一旦遇到return,程序就不会在继续往后执行。

#### 2.4.4 练习

A:比较两个数是否相等

B:获取两个数中的最大值

C:获取三个数中的最大值(if语句的嵌套)

D:根据成绩输出对应的等级

E:根据月份，输出对应的季节

F:根据x计算对应y的值并输出

G:看程序，分析下面程序的结果：

int x = 2,y=3;

switch(x)

{

default:

y++;

case 3:

y++;

break;

case 4:

y++;

}

System.out.println("y="+y);

3:根据输入的值，判断是星期几。(分别用if语句和switch语句实现)

输入：1

输出：星期1

4:for循环的格式?要能看懂执行流程。

用for循环完成如下案例

求和

求偶数和

求奇数和

打印水仙花数

统计水仙花数

九九乘法表

5:while循环的格式?要能看懂执行流程

用while循环完成如下案例

求和

纸张折叠成珠穆朗玛峰高度的次数

6:break,continue和return分别有什么用?

## 面向过程的设计方法

### 3.1 ja**va 方**法

### 3.2 jav**a数组**

## 第四节 面向对象的设计方法