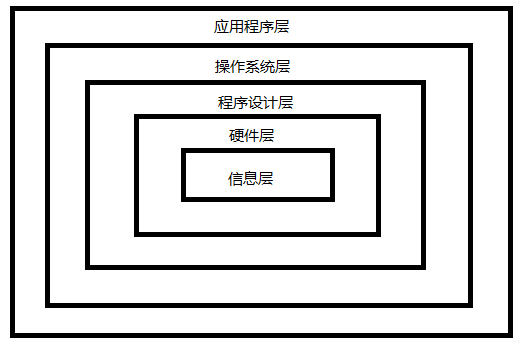
# AP 计算机科学—java

## 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？

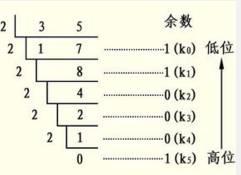
943 = 9 \* 102 + 4 \* 101 + 3\*100

##### 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

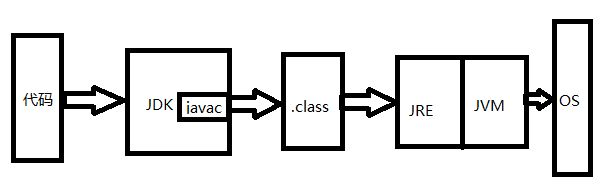
从英文意思可以看出，JVM是java的虚拟机，JRE是java的运行环境，JDK是java的开发工具包。

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");

}

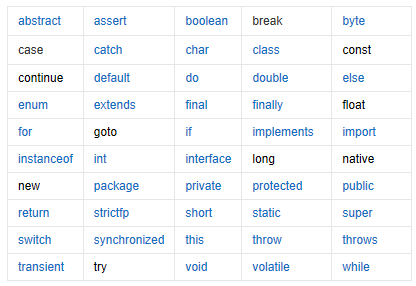
}

#### 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

##### 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

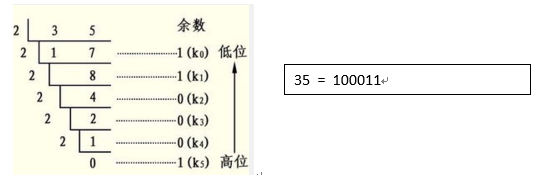
基数：x进制的基数就是x

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。

##### 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

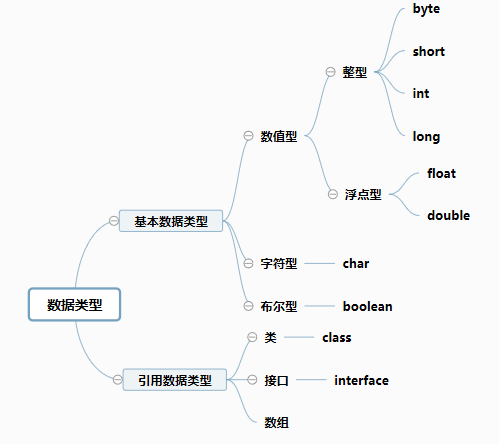
A:十进制和二进制间的转换

8421码。

B:二进制到八进制，十六进制的转换

#### 2.3.5 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。

##### 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

###### 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;

float f2 = (float)12.345;

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;

byte b4 = 3 + 4;

C:下面的操作结果是什么呢?

byte b = (byte)130;

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

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

### 2.4 流程控制

## 第三节 面向对象技术