第一阶段Java笔记

# 命令行

盘符切换：盘符名:

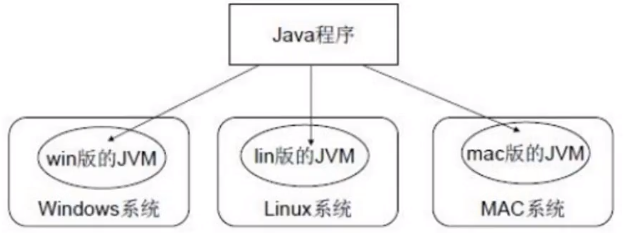
切换当前文件夹：cd 文件夹名+tab

回根路径：cd \

查看当前内容： dir

退出：exit

# Java虚拟机——JVM

JVM(Java Virtual Machine): Java虚拟机,简称JVM，是运行所有Java程序的假想计算机，是Java程序的**运行环境**，是Java最具吸引力的特性之一。我们编写的Java代码，都运行在JVM之上。  
  
跨平台：任何软件的运行，都必须要运行在操作系统之上，而我们用Java编写的软件可以运行在任何的操作系统上,这个特性称为Java语言的跨平台特性。该特性是由JVM实现的，我们编写的程序运行在JVM\_上，而JVM运行在操作系统上。但Java的虚拟机本身不具备跨平台功能的，每个操作系统下都有不同版本的虚拟机。  


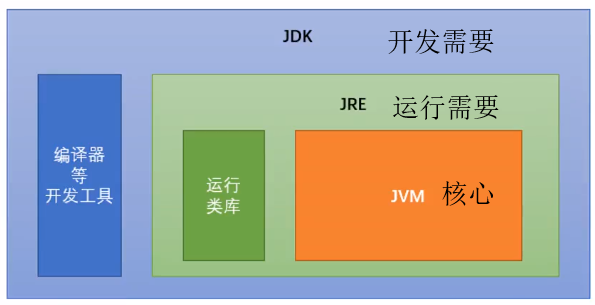
# JRE和JDK

JRE （Java Runtime Environment) ：是Java程序的**运行时环境**，包含JVM和运行时所需要的核心类库。

JDK (Java Development Kit)：是Java程序开发工具包，包含JRE和开发人员使用的工具。

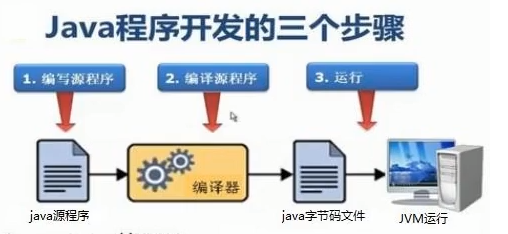
如果想要运行一个已有的Java程序，那么只需安装JRE即可。

如果想要开发一个全新的Java程序，那么必须安装JDK。



# Hello world

Java开发三步骤：编写，编译，运行。



## 程序注释与helloworld说明

public class后面代表定义一个类的名称，类是Java中所有源代码的基本组织单位。类名必须和所在文件名称一样，包括大小写。

main方法代表程序执行的起点。

# 标识符

标识符：在程序中,自己定义内容。比如类的名字、方法的名字和变量的名字等都是标识符。

**命名规则:（硬性要求）**

标识符可以包含英文字母26个(区分大小写)、0-9数字、 $ (美元符号)和\_ (下划线)。

标识符不能以数字开头。

标识符不能是关键字。

**命名规范: （软性建议）**

类名规范：首字母大写,后面每个单词首字母大写(大驼峰式)。

变量名规范：首字母小写，后面每个单词首字母大写(小驼峰式)。

方法名规范：首字母小写，后面每个单词首字母大写(小驼峰式)。

# 变量与常量

## 变量

常量的分类:

1.字符串常量：凡是用**双引号**引起来的部分，叫做字符串常量。例如: "abc"、 "Hello"、 "123"

2.整数常量：直接写上的数字，没有小数点。例如: 100、 200、 0、-250

3.浮点数常量：直接写上的数字，有小数点。例如: 2.5、 -3.14、 0.0

4.字符常量：凡是用**单引**号引起来的**单个字符**，就做字符常量。例如: 'A'、 'b'、'9'、 '中'

5.布尔常量：只有两种取值。true、 false.

6.空常量：null。代表没有任何数据。

## 常量

System.out.println(''); //两个单引号中间必须有且仅有一个字符，没有不行。

System.out.println(null); //空常量不能直接用来打印输出。

## 基本数据类型

Java的数据类型分为两大类：

基本数据类型：整数、浮点数、字符、布尔。

引用数据类型：类、数组、接口。

**四类八种基本数据类型：**

整数型byte short int long

浮点型float double

字符型char

布尔型boolean



Java中默认类型：整数类型是int，浮点类型是double。

**引用数据类型**

字符串、数组、类、接口、Lambda

所有的引用类型变量，都可以赋值一个null。

**注意事项:**

1.字符串不是基本类型，而是引用类型。

2.浮点型可能只是一个近似值，并非精确的值。

3.数据范围与字节数不一一定相关，例如float数据范围比long更加广泛，但是float是4字节，long是8字节。

4.浮点数当中默认类型是double。如果一定要使用float类型，需要加上一个后缀F。如果是整数，默认为int类型，如果一定要使用long类型，需要加上一个后缀L。推荐使用大写字母后缀。

## 使用变量时的注意事项

1.如果创建多个变量，那么变量之间的名称不可以重复。

2.对于float和long类型来说，字母后缀F和L不要丢掉。

3.如果使用byte或者short类型的变量，那么右侧的数据值不能超过左侧类型的范围。

4.没有进行赋值的变量，不能直接使用; 一定要赋值之后，才能使用。

5.变量使用不能超过作用域的范围。

[作用域] :从定义变量的一行开始，一直到直接所属的大括号结束为止。

6.可以通过一个语句来创建多个变量，但是一般情况不推荐这么写。

# 数据类型转换

## 自动类型转换（隐式）

1.特点：代码不需要进行特殊处理，自动完成。

2.规则：数据范围从小到大。

## 强制类型转换

1.特点:代码需要进行特殊的格式处理，不能自动完成。

2.格式:范围小的类型 范围小的变量名 = (范围小的类型)原本范围大的数据;

**注意事项**:

1.强制类型转换一般不推荐使用，因为有可能发生精度损失、数据溢出。

2.byte/short/char这三种类型都可以发生数学运算，例如加法“+”。

3.byte/ short/char这三种类型在**运算**的时候，都会被首先提升成为**int类型**，然后再计算。

# ASCII编码表

数字和字符的对照关系表(编码表) :  
ASCII码表：American Standard Code for Information Interchange，美国信息交换标准代码。  
Unicode码表：万国码。也是数字和符号的对照关系，开头0-127部分和ASCII完全一样，但是从128开始包含有更多字符。

# 算术运算

## 加法

四则运算当中的加号“+\*有常见的三种用法：

1.对于数值来说，那就是加法。

2.对于字符char类型来说，在计算之前，char会被提升成为int，然后再计算。char类型字符，和int类型数字，之间的对照关系表: ASCII、 Unicode。

3.对于字符串String (首字母大写，并不是关键字)来说，加号代表字符串连接操作。任何数据类型和字符串进行连接的时候，结果都会变成字符串。

## 自增自减运算符

自增运算符：++

自减运算符：--

基本含义：让一个变量涨一个数字1，或者让一个变量降一个数字1

使用格式：写在变量名称之前，或者写在变量名称之后。例如: ++num，num++。

使用方式:

1.单独使用:不和其他任何操作混合，自己独立成为一个步骤。

2.混合使用:和其他操作混合，例如与赋值混合，或者与打印操作混合等。

使用区别:

1.在单独使用的时候，前++和后++没有任何区别。也就是: ++num;和num++;是完全一样的。

2.在混合的时候，有[重大区别]

A.如果是[前++]，那么变量立刻马上+1，然后拿着结果进行使用。 [先加后用]

B.如果是[后++]，首先使用变量本来的数值，然后再让变量+1。[先用后加]

**注意事项:**

只有变量才能使用自增、自减运算符。常量不可发生改变，所以不能用。

## 赋值运算符



注意事项：

1. 只有变量才能使用赋值运算符，常量不能进行赋值。
2. 复合运算符其中隐含了一个强制类型转换。

## 逻辑运算符

与“&&”，或“||”，具有短路效果，如果根据左边已经可以判断得到最终结果，那么右边的代码将不再执行，从而节省一定的性能。

int a=10;

//false && …

System.out.println(3 > 4 && ++a < 100); //false

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

int b=20;

// true ll …

System.out.println(3 < 411++b < 100); //true

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

# 循环

## switch

语句格式：

switch(表达式) {

case 常量值1：

语句体1;

break;

case 常量値2:

语句体2;

break;

...

default:

语句体n+1;

break;

}

**注意事项：**

1.多个case后面的数值不可以重夏。

2.switch后面小括号当中只能是下列数据类型:

基本数据类型: byte/ short/char/int

引用数据类型: String字符串、enum枚举

3.switch语句格式可以很灵活:前后顺序可以颠倒，而且break语句可以省略：匹配哪一个case就从哪一个位置向下执行，直到遇到了break或者整体结束为止。

## 三种循环的区别

1. 如果条件判断从来没有满足过，那么for循环和while循环将会执行0次，但是do-while循环会执行一次。
2. for循环的变量在小括号中定义，只有循环内部可以使用，while循环和do-while殉国初始化语句本来就在外面，所以出循环后可以继续使用。

## break关键字

break关键字常用用法：

1. 可以在switch语句中，一旦执行，整个switch语句立即结束。
2. 还可以用在循环语句中，一旦执行，整个循环语句立即结束，打断循环。

## continue关键字

一旦执行，立刻跳过当次循环的剩余内容，马上开始下一轮循环。