编程语言，按照编译和执行的过程分为，编译型语言和解释型语言

编译型语言： 源文件--编译-->目标代码文件（机器码指令和数据） 编译之后，生成的目标代码文件，可以直接运行，效率高。不能跨平台

解释型语言：源文件--解释-->直接执行 不生成目标代码文件，而是每次运行时，都进行解释执行。效率低，可以跨平台

java是一半编译一半解释：

java基本语法规则：

1，一个java源文件中，可以有多个类，但是最多只能有一个被public修饰的类，并且，被public修饰的类必须与文件名同名。但，通常建议一个源文件中只有一个类。

2，java语言中的语句以；作为语句的结束符。

3，java语言严格区分大小写。

4，java源文件编译使用 javac命令，运行使用java命令

5，main（）方法是整个程序的入口，执行流进入main方法，逐条运行代码。

6，main（）方法的格式是固定的。

关键字：被java语言赋予了特殊含义的字符串。

标识符：凡是由程序员自己定义的名字，都叫做标识符。

int i=5;//定义一个变量

int i; //变量的声明

i=5; //变量的赋值

// 在同一作用域内，标识符必须具有唯一性（避免二义性）

标识符命名规则：1，数字，字母，下划线，$ 组成，不能以数字开头，长度无限制。

2，某一个区域中是唯一的，在不同的区域中可以使用同一名字

3，严格区分大小写。

4，不能使用关键字、保留字

5，不能使用空格来分隔

标识符命名约定：1，类，接口，首字母大写，如果由多个单词组成，每个单词的首字母大写。myname MyName

2，方法名，变量名，首字母小写，如果由多个单词组成，从第二个单词开始首字母大写。myName（） myName

3，常量名：全部由大写字母组成，如果由多个单词组成，单词间用下划线分隔。

基本数据类型，引用类型。

基本数据类型：8种，表示一个基本的值。

引用类型：类或者由类创建的对象，都称之为引用类型。

数据类型的转换：转换的前提是，数据类型兼容。

自动类型转换：从表示范围小的类型，向表示范围大的类型转换。 如：int———long

强制类型转换：从表示范围大的类型，向表示范围小的类型转换。

在java语言中，所有小数类型的字面量。默认的数据类型是double，后面加f、F改变默认数据类型为float

所有小于128的整数的字面量，默认数据类型是int，后面加L、l改变默认类型为long

**char** 也是保存的数字

char类型是可以运算的因为char在ASCII等字符编码表中有对应的数值

对char进行运算时

char输出结果转换为数字对应的字符

int输出结果转换为相应字符汉字对应的数字

例：char m=29579 输出m 则为“王”

int m='王' 输出m 为29579 （即完成了王的ASCII之间的转换）

https://blog.csdn.net/ibelieve618/article/details/54948729（博客解释）

编程示例：（只能有一个m，所以跑程序只能注释其他只跑其中一个）

package example;

public class example\_char {

public static void main(String[] args) {

char m=29579;

System.out.println(m);//输出 王

int m = '王';

System.out.println(m); //输出 29579

char m = 'a';

System.out.println(m);//输出 a

char m = 'a'+'b';

System.out.println(m);//输出 Ã

int m = 'a'+'b';

System.out.println(m);//输出 195 没有超出int范围

char m = 195;

System.out.println(m);//输出 Ã

int m = '中'+'国';

System.out.println(m);//输出 42282

char m ='中'+1;

System.out.println(m);//输出 丮

}

}

**Java中逻辑运算符是按位运算**

例：a=1;b=2

即a:00000001 b:00000010

a|b 按位运算 值为：00000011 即 3

Java中 移位（<<符号位不动 <<<符号位也移动） 向左（<<）几位就是乘2的几次 向右(>>)就是除以2的几次

**1.**计算机中存储整数都是用的补码，取反运算也是对补码进行取反

**2**.所有的数字大小都是根据原码的大小计算的

**3**.正整数的反码，补码和原码相同，负整数的反码是除了符号位以外取反，负数的补码为反码+1

**4.**对补码再求补码即是原码

对一个数进行取反运算：对该数的二进制码进行求补码（正数原反补码都相同，负数反码为除符号位取反、补码对反码+1）

求出补码之后，对补码进行取反（连同符号位），取反后观察符号位，判断是正数还是负数，然后对该数进行求补码

**eg**:

计算int类型 ~(+9)：

原码：0...000...1001 (总共32位，四字节)

反码：0...000...1001

补码：0...000...1001

取反运算：1...111...0110 ，最终结果转换为原码显示，需要对该值再求补码，符号位为1，是负数，求补码需取反再加一，

得1...000...1010，即十进制的 -10

eg:

计算int类型 ~(-9)：

原码：1...000...1001 (总共32位，四字节)

反码：1...111...0110

补码：1...111...0111

取反运算：0...000...1000 ，最终结果转换为原码显示，符号位为0，是正数，正数补码与原码相同，所以也为0...000...1000,即十进制的8

**string和StringBuffer和StringBuilder**

String是不可变的；StringBuffer是可变的

String类是不可变的，对象一旦被创建，就不能被修改；可以使用=直接赋值，此时使用常量池；

也可以使用new创建，不使用常量池

StringBuffer是可变的，对象创建后，可以修改；必须使用new关键字创建。

StringBuilder是不同步的，在单线程情况下使用比StringBuffer高效；必须使用new关键字

**自动装箱和拆箱**

装箱：基本数据类型转换为包装器类型，称为装箱（boxing）；例如，int型转换为Integer类型；

拆箱：包装器类型转换为基本数据类型，称为拆箱（unboxing）；例如Integer类型转换为int类型；

JDK1.5以后，装箱拆箱可以自动进行