# Java知识梳理

## JAVA基础

### **GitHub Idea的简单使用**<https://www.ycbbs.vip/?p=140>

Git工具的使用:<https://www.runoob.com/w3cnote/git-guide.html>

### **[JDK是什么?JRE是什么?JVM是什么？JDK和JRE的区别?以及jdk安装和测试。](https://www.ycbbs.vip/?p=102)**

#### Java Runtime Environment (JRE) 包含：

Java虚拟机(JVM)、库函数、运行 Java 应用程序和 Applet 所必须文件

Java 2 SDK (Development Kit)包含：JRE的超集，包含编译器和调试器等用于程序开发的文件 。

Java运行环境的三项主要功能：  
1. 加载代码：由class loader 完成；  
2. 校验代码：由bytecode verifier 完成；  
3. 执行代码：由 runtime interpreter完成。

**sdk（也就是jdk）**是jre的超集，是在jre的基础上增加了编译器及其他一些开发工具。  
jre就是java运行时环境，包括了jvm和其它一些java核心api,任何一台电脑，只有安装了jre才可以运行java程序。  
**总结**：  
1. 如果只是要运行JAVA程序，之需要JRE就可以。 JRE通常非常小，也包含了JVM.

#### JVM是什么?

JVM是一种规范，可以使用软件来实现，也可以使用硬件来实现，就是一个虚拟的用于执行 bytecodes 字节码的计算机。他也定义了指令集、寄存器集、结构栈、垃圾收集堆、内存区域。

JVM 负责将 java 字节码解释运行，边解释边运行，这样，速度就会受到一定的影响。JAVA提供了另一种解释运行的方法JIT（just in time），可以一次解释完，再运行特定平台上的机器码，高级的 JIT 可以只能分析热点代码，并将这些代码转成本地机器码，并将结果缓存起来，下次直接从内存中调用，这样就大大提高了执行 JAVA 代码的效率。这样就实现了跨平台、可移植的功能。

JVM 是指在一台计算机上由软件或硬件模拟的计算机；它类似一个小巧而高效的 CPU。

byte-code 代码是与平台无关的是虚拟机的机器指令。

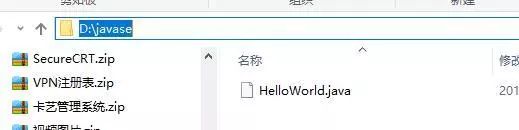
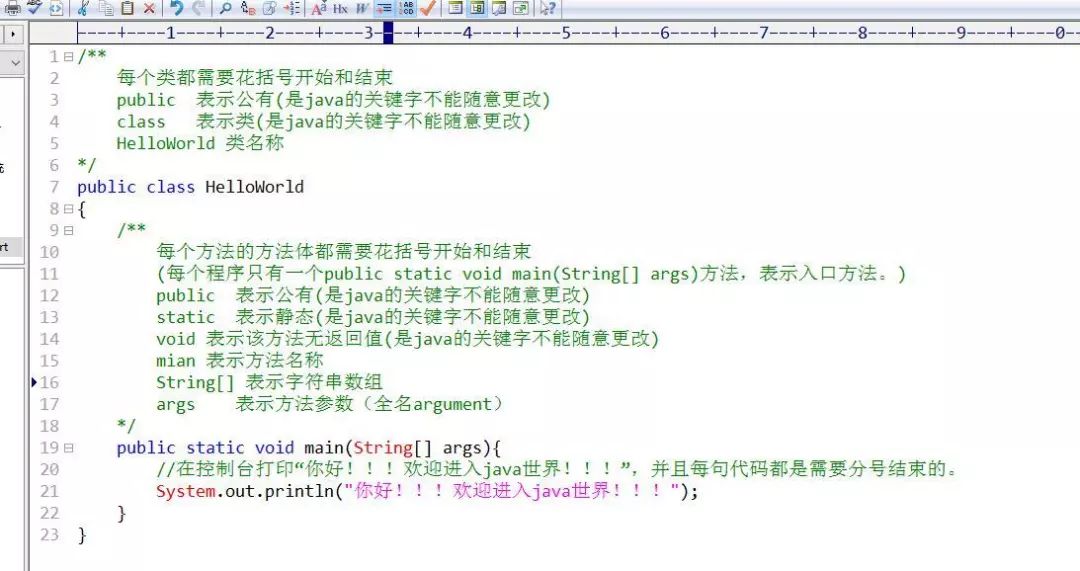
java字节代码运行的两种方式:  
1)方式interpreter(解释)  
2)Just-in-time(即时编译):  
有代码生成器将字节代码转换成本机的机器代码,然后可以以较高速度执行。

**总结**：  
1. JAVA 的跨平台实现的核心是不同平台使用不同的虚拟机。  
2. 不同的操作系统有不同的虚拟机。  
3. Java 虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，随处运行”。

### **[3、Java入门之HelloWorld](https://www.ycbbs.vip/?p=142)**

**书写第一个java程序**

1) 先创建一个HelloWorld.java文件,将文件放在一个电脑固定的一个位置。至于放在什么位置根据个人喜好来定。小编将在D:\javase文件夹中创建。

注意：文件夹名称和.java文件尽量使用英文名称。  
  
2) 使用[editplus下载地址](https://pan.baidu.com/s/1dtT0JymSIBBQR160J-QuNQ" \o "editplus下载地址)  
3) 然后开始编写程序  
  
现在程序就写好了。以上也写出了每行代码的解释，初学者一般喜欢忘记写分号。  
注意：类名一定要与文件名一致大小写都必须一致，并且java语言是严重区分大小写的。

**编译文件和执行文件操作步骤。**

1) 进入到你保存HelloWorld.java的目录下然后回车

命令：cd D:\javase 如下图  


2) 然后使用javac命令来编译HelloWorld.java  
命令：(javac HelloWorld.java)  
如下图：  


3) 现在就会编译出一个可执行文件HelloWorld.class如图  


4) 执行HelloWorld.class文件  
命令：java HelloWorld  
如图：  


注意：每次修改程序时都必须重新编译一次在执行即可。并且一个 java 文件对应多个类文件。每个类文件中只能有一个类的修饰符为 public。  
总结  
计算机的高级编程语言类型: 编译型 ，解释型。  
Java 语言是两种类型的结合；  
1）编译  
利用编译器（javac）将源程序编译成字节码文件，文件名：源文件名.class  
2）运行  
利用虚拟机（解释器，java）解释执行 class 字节码文件。

### 4、[Java标识符](https://www.ycbbs.vip/?p=165)

**标识符使用标识符时，需要遵守几条规则**

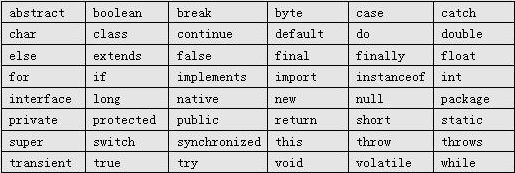
1) 标识符可以由字母、数字、下划线（\_）、美元符（$）组成，但不能包含 @、%、空格等其它特殊字符，不能以数字开头。譬如：111stu 就是不合法。

2) 标识符不能是 Java关键字和保留字（Java预留的关键字，以后的升级版本中有可能作为关键字），但可以包含关键字和保留字。如：不可以使用 void 作为标识符，但是 Stuvoid 可以。

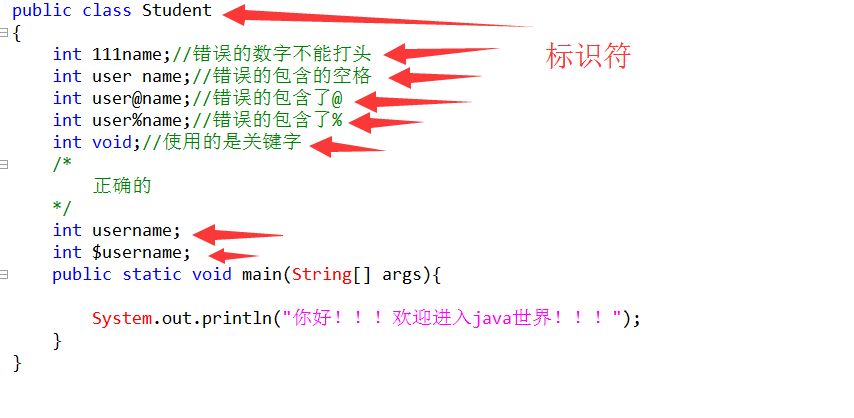
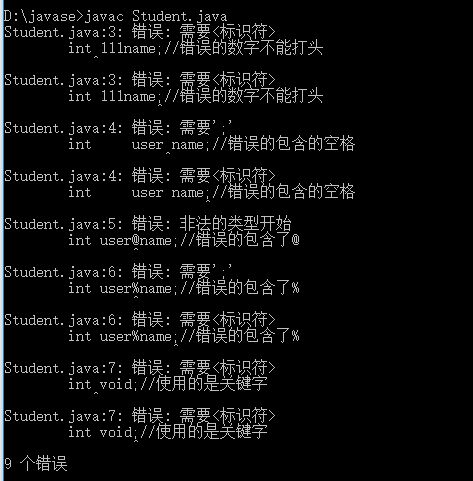
3) 标识符是严格区分大小写的。并且没有长度限制的。 一定要分清楚 Student 和 student 是两个不同的标识符！

4) 标识符的命名最好能反映出其作用，做到见名知意。  
例如：username password

5) Java 语言中有一些具有特殊用途的词被称为关键字。

6) Java 中常用关键字：  


注意：java 中是可以使用中文命名的。因为 java 内部使用了 Unicode 字符集

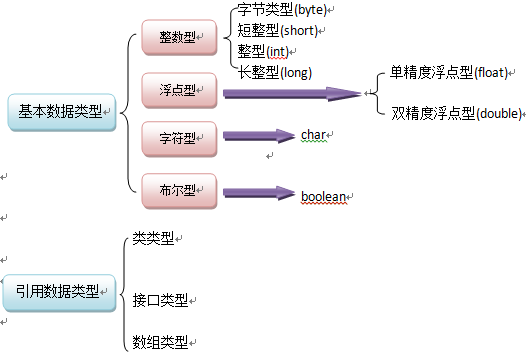
如图显示  
  


大家可以比对根据上面的错误信息！！！

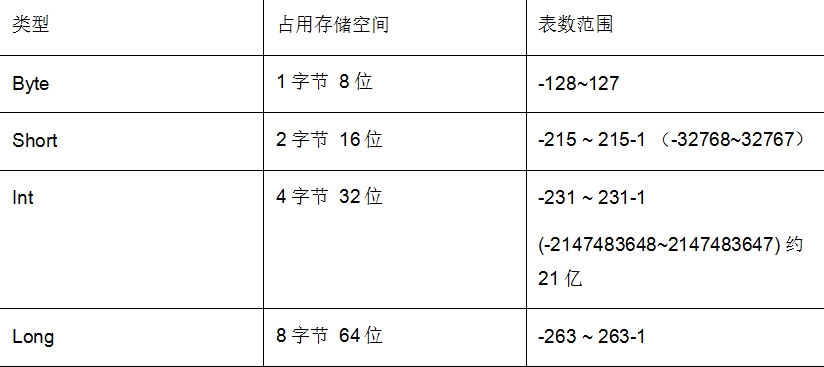
### [Java数据类型（上）](https://www.ycbbs.vip/?p=168)

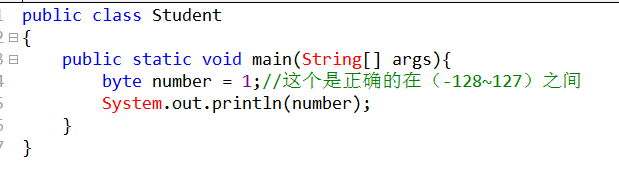
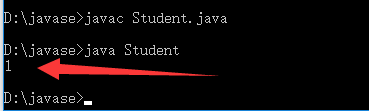
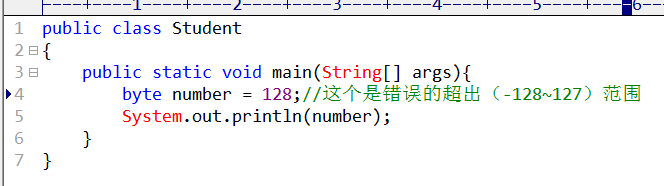
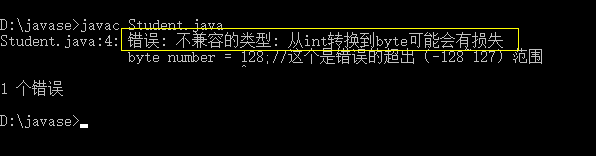
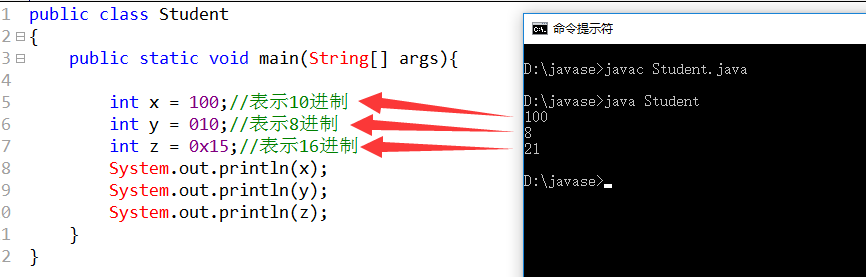
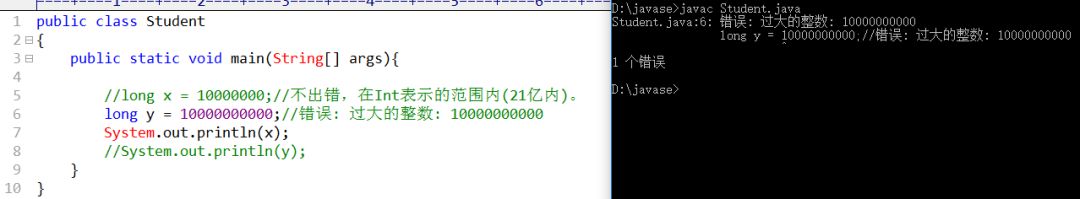
#### Java是一种强类型语言，每个变量都必须声明其类型。

1) Java的数据类型分为两大类：基本类型和引用类型

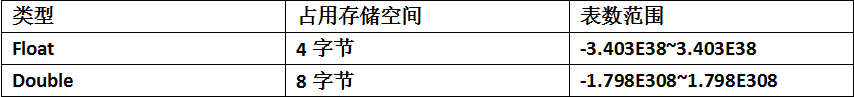
2) 在客户碰到困难的时候施以援手，雪中送炭比锦上添花更让人感动；  


#### 整数型（byte,short,int,long）

大家来看一个图：  


1) 举例一个byte其它以此类推：  
byte 是一个字节保存的,有8个位,即8个0、1。  
8 位的第一个位是符号位， 也就是说 0000 0001 代表的是数字1  
1000 0000 代表的就是-1，所以正数最大位 0111 1111，也就是数字127  
负数最大为 1111 1111，也就是数字-128，所以 Java 中的一个 byte 是 1 个字节，其范围是-128~127  
  
  
下面看看错误的，大家比对一下。  
  
  
2) Java 语言整型常数的三种表示形式：  
十进制整数，如：100, -500, 0。  
八进制整数，要求以 0 开头，如：011。  
十六进制数，要求 0x 或 0X 开头，如：0x15 。  
如图：  
  
Java语言的整型常数默认为int型，声明long型常量可以后加‘ l ’或‘L ’ ，如：  
long a = 10000000; //不出错，在Int表示的范围内(21亿内)。  
long b = 10000000000;//必须要加10000000000L，要不然就会报错错误: 过大的整数: 10000000000  


#### 浮点型

  
float类型又被称作单精度类型，尾数可以精确到 7 位有效数字，在很多情况下，float 类型的精度很难满足需求。

double 表示这种类型的数值精度是 float 类型的两倍，又被称作双精度，绝大部分应用程序都采用 double 类型。

Java 浮点类型常量有两种表示形式

十进制数形式，例如:

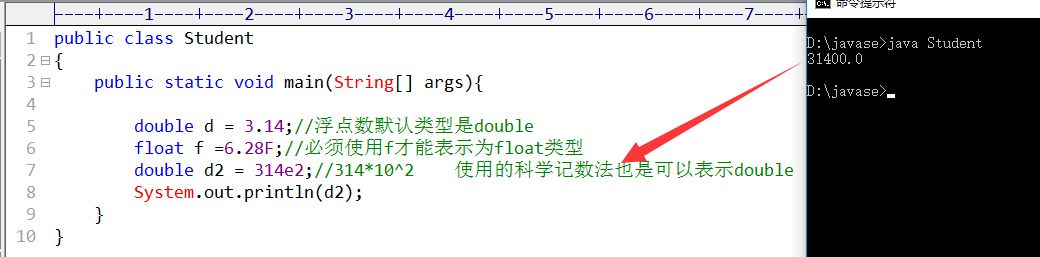
3.14 314.0 0.314

科学记数法形式，如

314e2 314E2 314E-2

double f = 314e2; //314\*10^2-->31400.0

double f2 = 314e-2; //314\*10^(-2)-->3.14

Float类型的数值有一个后缀F/f，没有后缀F/f的浮点数值默认为double类型。也可以在浮点数值后添加后缀D/d, 以明确其为double类型：  
  
**浮点类型float, double的数据不适合在计算或者比较中使用，因为会丢失精度。  
如果需要精确数字计算，需要使用 BigDecimal 类。**

注意：主要理由：由于字长有限，浮点数能够精确表示的数是有限的，因而也是离散的。浮点数一般都存在舍入误差，很多数字无法精确表示(例如0.1)，其结果只能是接近， 但不等于。

PS：官方文档中好像说float指数的取值范围为-126~127，double指取的取值数范围为-1022~1023

#### java.math.BigDecimal

BigDecimal构造方法

　　1.public BigDecimal(double val)    将double表示形式转换为BigDecimal \*不建议使用

　　2.public BigDecimal(int val)　　将int表示形式转换成BigDecimal

　　3.public BigDecimal(String val)　　将String表示形式转换成BigDecimal

为什么不建议采用第一种构造方法呢？来看例子

[IMG_256](https://www.cnblogs.com/LeoBoy/p/javascript:void(0);)

public static void main(String[] args)

{

BigDecimal bigDecimal = new BigDecimal(2);

BigDecimal bDouble = new BigDecimal(2.3);

BigDecimal bString = new BigDecimal("2.3");

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

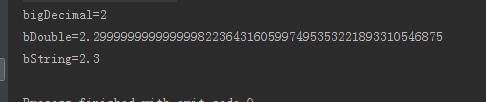
System.out.println("bDouble=" + bDouble);

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

}

[IMG_257](https://www.cnblogs.com/LeoBoy/p/javascript:void(0);)

运行结果如下



为什么会出现这种情况呢？

 JDK的描述：1、参数类型为double的构造方法的结果有一定的不可预知性。有人可能认为在Java中写入newBigDecimal(0.1)所创建的BigDecimal正好等于 0.1（非标度值 1，其标度为 1），但是它实际上等于0.1000000000000000055511151231257827021181583404541015625。这是因为0.1无法准确地表示为 double（或者说对于该情况，不能表示为任何有限长度的二进制小数）。这样，传入到构造方法的值不会正好等于 0.1（虽然表面上等于该值）。

        2、另一方面，String 构造方法是完全可预知的：写入 newBigDecimal("0.1") 将创建一个 BigDecimal，它正好等于预期的 0.1。因此，比较而言，**通常建议优先使用String构造方法**。

**当**double**必须用作**BigDecimal**的源时，**请使用Double.toString(double)转成String，然后使用String构造方法，或使用BigDecimal的静态方法valueOf，如下

[IMG_259](https://www.cnblogs.com/LeoBoy/p/javascript:void(0);)

public static void main(String[] args)

{

BigDecimal bDouble1 = BigDecimal.valueOf(2.3);

BigDecimal bDouble2 = new BigDecimal(Double.toString(2.3));

System.out.println("bDouble1=" + bDouble1);

System.out.println("bDouble2=" + bDouble2);

}

[IMG_260](https://www.cnblogs.com/LeoBoy/p/javascript:void(0);)

结果如下

IMG_261

总结

　　　(1)商业计算使用BigDecimal。

        (2)尽量使用参数类型为String的构造函数。

        (3) BigDecimal都是不可变的（immutable）的，在进行每一步运算时，都会产生一个新的对象，所以在做加减乘除运算时千万要保存操作后的值。

        (4)我们往往容易忽略JDK底层的一些实现细节，导致出现错误，需要多加注意。

### 6、[Java数据类型(下)](https://www.ycbbs.vip/?p=170)

#### Char类型

Char 是字符类型（单个字符）占2个字节16位。

单引号用来表示字符常量。例如‘A’是一个字符，它与“A”是不同的，“A”表示含有一个字符的字符串。

char 类型用来表示在 Unicode 编码表中的字符。

Unicode 编码被设计用来处理各种语言的所有文字，它占2个字节，可允许有65536个字符；ASCII码占1个字节，可允许有128个字符，是Unicode编码表中前128个字符。

char eChar = 'a';

char cChar ='中';

Unicode具有从0到65535之间的编码,他们通常用从’\u0000’到’\uFFFF’之间的十六进制值来表示（前缀为u表示Unicode）

char c = ‘\u0061’;

Java 语言中还允许使用转义字符 ‘\’ 来将其后的字符转变为其它的含义，

char c2 = '\n'; //代表换行符



#### char运算

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。

在 JAVA 中，对 char 类型字符运行时，直接当做 ASCII 表对应的整数来对待

char a = 'a';

int b = 2;

int c = a+b;

System.out.println(c);//答案是99

#### Boolean类型

boolean类型一位，不是一个字节boolean类型有两个值，true和false,不可以 0 或非 0 的整数替代 true 和 false ，这点和C语言不同。

#### 总结：

用单引号''标识，只能放单个字符。

char+char，char+int——类型均提升为int，附值char变量后，输出字符编码表中对应的字符。

boolean类型一位，不是一个字节boolean类型有两个值，true和false,不可以 0 或非 0 的整数替代 true 和 false ，这点和C语言不同。

boolean bool = true;//或者boolean bool = false;

boolean 类型用来判断逻辑条件，一般用于if、while、do while。boolean sex = false;

案例一

if(sex==true){System.out.println("你是男的");}

else{System.out.println("你是女的");}

案例二

while(sex){

...}

案例三

do{

}while(sex)

### 7、[Java字符串(String)](https://www.ycbbs.vip/?p=173)

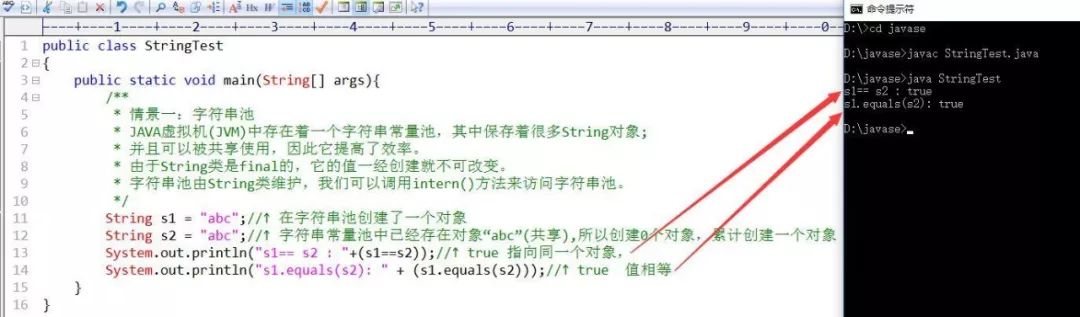
#### String是一个特殊的包装类数据。

String 即可以用 String str= new String("abc");的形式来创建，也可以用String str ="abc"；的形式来创建。  
**String str = "abc"创建对象的过程：**

1 首先在常量池中查找是否存在内容为”abc“的字符串对象  
2 如果不存在则在常量池中创建一个”abc“的字符串对象，并让str引用该对象  
3 如果存在则直接让 str 引用该对象

注意：常量池属于类信息的一部分，而类信息反映到 JVM 内存模型中是对应存在于 JVM 内存模型的方法区，也就是说这个类信息中的常量池概念是存在于在方法区中。

一般这种情况下，”abc“在编译时就被写入字节码中，所以 class 被加载时，JVM 就为”abc“在常量池中分配内存，所以和静态区差不多。

如图：  


**String str = new String("abc")创建实例的过程：**

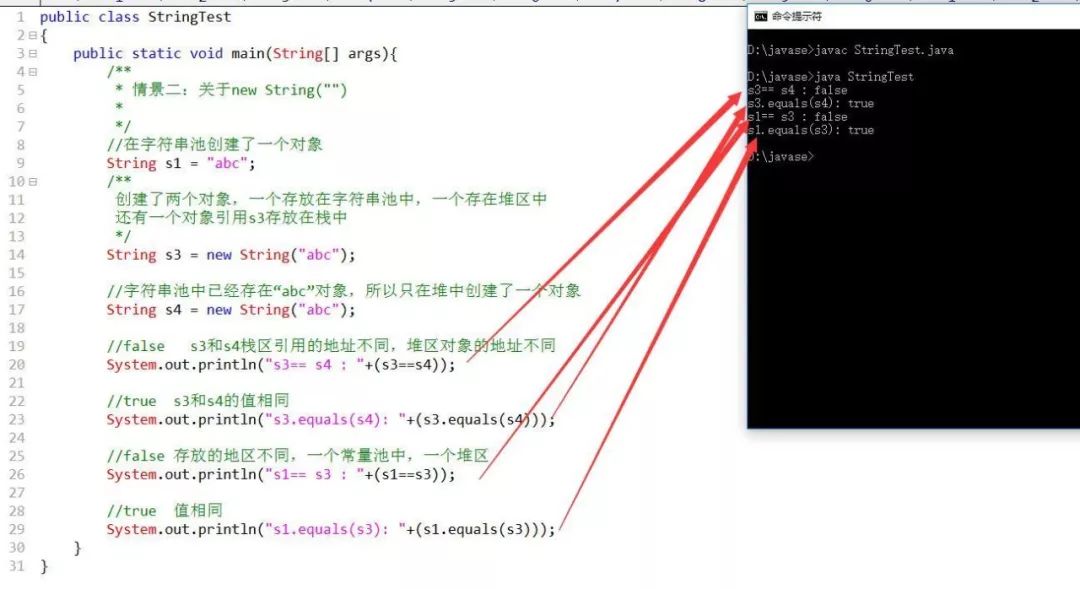
1首先定义一个str的String类型的引用并存放在栈中

2在字符串常量池中查看是否存在内容为"abc"字符串对象

3若存在则跳过这个步骤，若不存在，则在字符串常量池中创建一个内容为"abc"的字符串对象。

4执行new操作，在堆中创建一个指定的对象"abc"，这里堆的对象是字符串常量池“abc”对象的一个拷贝对象。

5让str指向堆中“abc”这个对象（也就是存储这个对象的在堆中的地址）



#### **字符串中的常用方法** (1) 字符串的连接 public String concat(String str) 该方法的参数为一个 String 类对象,作用是将参数中的字符串 str 连接到原来字符串的后面.

(2)求字符串的长度  
public int length()  
返回字串的长度,这里的长度指的是字符串中 Unicode 字符的数目.

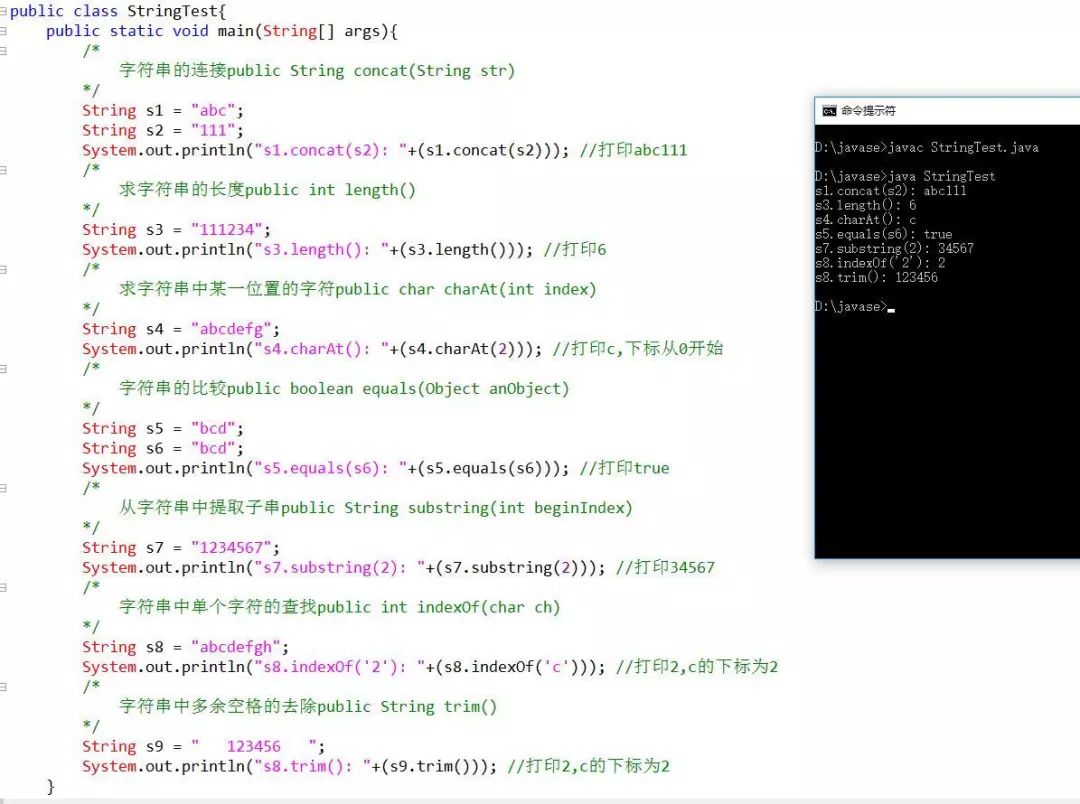
(3)求字符串中某一位置的字符  
public char charAt(int index)  
该方法在一个特定的位置索引一个字符串,以得到字符串中指定位置的字符.值得注意的是,在字符串中第一个字符的索引是0,第二个字符的索引是1,依次类推,最后一个字符的索引是length()-1.

(4)字符串的比较  
比较字符串可以利用String类提供的下列方法:  
public boolean equals(Object anObject)  
该方法比较两个字符串,和Character类提供的equals方法相似,因为它们都是重载Object类的方法.该方法比较当前字符串和参数字符串,在两个字符串相等的时候返回true,否则返回false.

(5)从字符串中提取子串  
public String substring(int beginIndex)  
该方法从beginIndex位置起,从当前字符串中取出剩余的字符作为一个新的字符串返回.

(6) 字符串中单个字符的查找  
字符串中单个字符的查找可以利用String类提供的下列方法:  
public int indexOf(int ch)

该方法用于查找当前字符串中某一个特定字符ch出现的位置.该方法从头向后查找,如果在字符串中找到字符ch,则返回字符ch在字符串中第一次出现的位置;如果在整个字符串中没有找到字符ch,则返回-1.

(7) 字符串中多余空格的去除  
public String trim()  
该方法只是去掉开头和结尾的空格,并返回得到的新字符串.值得注意的是,在原来字符串中间的空格并不去掉.  


### 8、[Java之变量](https://www.ycbbs.vip/?p=175)

#### 变量(variable)

我们通过变量来操纵存储空间中的数据，变量就是指这个存储空间！空间位置是确定的，但是里面放置什么值不确定！

Java是一种强类型语言，每个变量都必须声明其类型。

Java变量是程序中最基本的存储单元，其要素包括变量名，变量类型和作用域。

变量在使用前必须对其声明, 只有在变量声明以后，才能为其分配相应长度的存储单元，声明格式为：

type varName [=value] [{,varName[=value]}] ；

注意事项：

每个变量都有类型，类型可以是基本类型，也可以是引用类型。  
变量名必须是合法的标识符(标识符前面已经讲解)。变量声明是一条完整的语句，因此每一个声明都必须以分号结束

变量声明举例：

double salary ；

boolean done；

long earthPopulation ；

int age ；

可以在一行中声明多个变量：

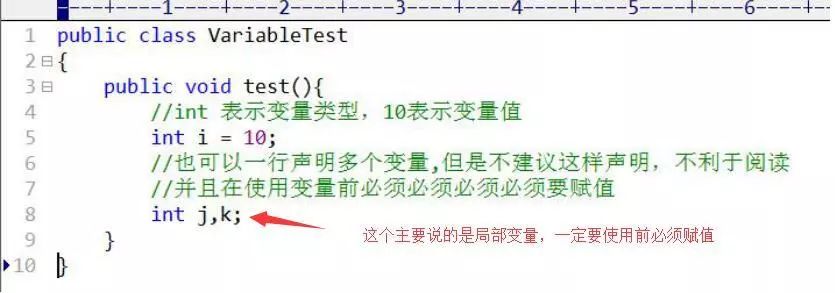
int i ,j;

不提倡这种风格，逐一声明每一个变量可以提高程序可读性。

可以将变量的声明和初始化放在同一行中，例如：

int age = 18;

float e = 2.718281828f;

如图：  


#### 变量可划分为

局部变量(lacal variable)：方法或语句块内部定义的变量。生命周期是**从声明位置开始到”}”为止**。  
在使用前必须先声明和初始化(赋初值)。

实例变量（成员变量 member variable）：方法外部、类的内部定义的变量。**从属于对象，生命周期伴随对象始终**。隶属于类。

如果不自行初始化，他会自动初始化成该类型的默认初始值（数值型变量初始化成0或0.0，字符型变量的初始化值是16位的0，布尔型默认是false）

静态变量（类变量 static variable）：使用static定义。 从属于类，生命周期伴随类始终，**从类加载到卸载**。如果不自行初始化，它会自动初始化成该类型的默认初始值（数值型变量初始化成0或0.0，字符型变量的初始化值是16位的0，布尔型默认是false）

#### 常量

常量：初始化(initialize)后不能再改变值！也就是说永远只有一个值  
如图（常量上面的局部变量、成员变量、 静态变量）：  


#### 变量命名规则(规范)

注意：规范不代表是规定

1. 所有变量、方法、类名：见名知意

2. 类成员变量：首字母小写和驼峰原则 : `monthSalary`

3. 局部变量：首字母小写和驼峰原则

4. 常量：大写字母和下划线：`MAX\_VALUE`

5. 类名：首字母大写和驼峰原则: `Man`, `GoodMan`

6. 方法名：首字母小写和驼峰原则:` run()`, `runRun()`

### 9、[Java基本数据类型转换](https://www.ycbbs.vip/?p=179)

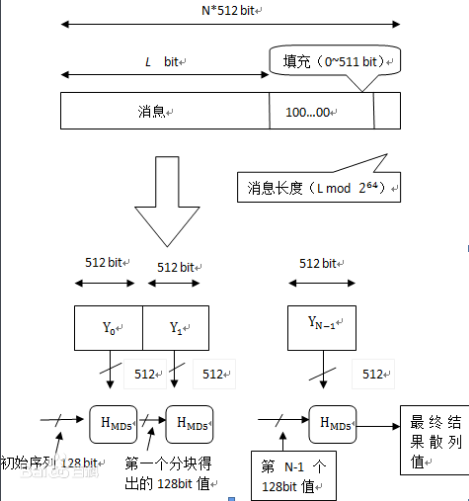
## 加解密算法

### md5

**MD5信息摘要算法**（英语：MD5 Message-Digest Algorithm），一种被广泛使用的[密码散列函数](https://baike.baidu.com/item/%E5%AF%86%E7%A0%81%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B0/14937715" \t "https://baike.baidu.com/item/MD5/_blank)，可以产生出一个128位（16[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82/1096318" \t "https://baike.baidu.com/item/MD5/_blank)）的散列值（hash value），用于确保信息传输完整一致。2004年，证实MD5算法无法防止碰撞（collision），因此不适用于安全性认证，如[SSL](https://baike.baidu.com/item/SSL/320778" \t "https://baike.baidu.com/item/MD5/_blank)公开密钥认证或是[数字签名](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E7%AD%BE%E5%90%8D/212550" \t "https://baike.baidu.com/item/MD5/_blank)等用途。

原理：MD5算法的原理可简要的叙述为：MD5码以512位分组来处理输入的信息，且每一分组又被划分为16个32位子分组，经过了一系列的处理后，算法的输出由四个32位分组组成，将这四个32位分组级联后将生成一个128位[散列值](https://baike.baidu.com/item/%E6%95%A3%E5%88%97%E5%80%BC/10398613" \t "https://baike.baidu.com/item/MD5/_blank)。 [6]

总体流程如下图所示，每次的运算都由前一轮的128位结果值和当前的512bit值进行运算 [7]  。



**算法步骤**

按位补充数据

在MD5算法中，首先需要对信息进行填充，这个数据按位(bit)补充，要求最终的位数对512求模的结果为448。也就是说数据补位后，其位数长度只差64位(bit)就是512的整数倍。即便是这个数据的位数对512求模的结果正好是448也必须进行补位。补位的实现过程：首先在数据后补一个1 bit； 接着在后面补上一堆0 bit, 直到整个数据的位数对512求模的结果正好为448。总之，至少补1位，而最多可能补512位 [8]  。

扩展长度

在完成补位工作后，又将一个表示数据原始长度的64 bit数(这是对原始数据没有补位前长度的描述，用二进制来表示)补在最后。当完成补位及补充数据的描述后，得到的结果数据长度正好是512的整数倍。也就是说长度正好是16个(32bit) 字的整数倍 [8]  。

初始化MD缓存器

MD5运算要用到一个128位的MD5缓存器，用来保存中间变量和最终结果。该缓存器又可看成是4个32位的寄存器A、B、C、D，初始化为 [8]  :

A ： 01 23 45 67

B： 89 ab cd ef

C： fe dc ba 98

D： 76 54 32 10

处理数据段

首先定义4个非线性函数F、G、H、I，对输入的报文运算以512位数据段为单位进行处理。对每个数据段都要进行4轮的逻辑处理，在4轮中分别使用4个不同的函数F、G、H、I。每一轮以ABCD和当前的512位的块为输入，处理后送入ABCD(128位) [8]  。

输出

信息摘要最终处理成以A, B, C, D 的形式输出。也就是开始于A的低位在前的顺序字节，结束于D的高位在前的顺序字节 [8]  。

**应用**

用于密码管理

当我们需要保存某些密码信息以用于身份确认时，如果直接将密码信息以明码方式保存在数据库中，不使用任何保密措施，系统管理员就很容易能得到原来的密码信息，这些信息一旦泄露， 密码也很容易被破译。为了增加安全性，有必要对[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "https://baike.baidu.com/item/MD5/_blank)中需要保密的信息进行加密，这样，即使有人得到了整个数据库，如果没有解密算法，也不能得到原来的密码信息。MD5算法可以很好地解决这个问题，因为它可以将任意长度的输入串经过计算得到固定长度的输出，而且只有在明文相同的情况下，才能等到相同的密文，并且这个算法是不可逆的，即便得到了加密以后的密文，也不可能通过解密算法反算出明文。这样就可以把用户的密码以MD5值（或类似的其它算法）的方式保存起来，用户注册的时候，系统是把用户输入的密码计算成 MD5 值，然后再去和系统中保存的 MD5 值进行比较，如果密文相同，就可以认定密码是正确的，否则密码错误。通过这样的步骤，系统在并不知道用户密码明码的情况下就可以确定用户登录系统的合法性。这样不但可以避免用户的密码被具有系统管理员权限的用户知道，而且还在一定程度上增加了密码被破解的难度 [8]  。

电子签名

MD5 算法还可以作为一种[电子签名](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E7%AD%BE%E5%90%8D/305853" \t "https://baike.baidu.com/item/MD5/_blank)的方法来使用，使用 MD5算法就可以为任何文件（不管其大小、格式、数量）产生一个独一无二的“数字指纹”，借助这个“数字指纹”，通过检查文件前后 MD5 值是否发生了改变，就可以知道源文件是否被改动。我们在下载软件的时候经常会发现，软件的下载页面上除了会提供软件的下载地址以外，还会给出一串长长的字符串。这串字符串其实就是该软件的MD5 值，它的作用就在于下载该软件后，对下载得到的文件用专门的软件（如 Windows MD5 check 等）做一次 MD5 校验，以确保我们获得的文件与该站点提供的文件为同一文件。利用 MD5 算法来进行文件校验的方案被大量应用到软件下载站、论坛数据库、系统文件安全等方面 [8]  。

垃圾邮件筛选

在[电子邮件](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6/111106" \t "https://baike.baidu.com/item/MD5/_blank)使用越来越普遍的情况下，可以利用 MD5 算法在邮件接收服务器上进行垃圾邮件的筛选，以减少此类邮件的干扰，具体思路如下：

1、建立一个邮件 MD5 值资料库，分别储存邮件的 MD5 值、允许出现的次数（假定为 3）和出现次数（初值为零）。

2、对每一封收到的邮件，将它的正文部分进行MD5 计算，得到 MD5 值，将这个值在资料库中进行搜索。

3、如未发现相同的 MD5 值，说明此邮件是第一次收到，将此 MD5 值存入资料库，并将出现次数置为1，转到第五步。

4、如发现相同的 MD5 值，说明收到过同样内容的邮件，将出现次数加 1，并与允许出现次数相比较，如小于允许出现次数，就转到第五步。否则中止接收该邮件。结束。

5、接收该邮件 [8]  。

**安全性分析**

MD5 算法自诞生之日起，就有很多人试图证明和发现它的不安全之处，即存在碰撞（在对两个不同的内容使用 MD5算法运算的时候，有可能得到一对相同的结果值） [8]  。2009年，[中国科学院](https://baike.baidu.com/item/%E4%B8%AD%E5%9B%BD%E7%A7%91%E5%AD%A6%E9%99%A2/271375" \t "https://baike.baidu.com/item/MD5/_blank)的谢涛和冯登国仅用了IMG_256 的碰撞算法复杂度，破解了MD5的碰撞抵抗，该攻击在普通计算机上运行只需要数秒钟 [9]  。

## 分布式

分布式计算是计算机科学中一个研究方向，它研究如何把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，然后把这些部分分配给多个计算机进行处理，最后把这些计算结果综合起来得到最终的结果。分布式[网络存储技术](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%AD%98%E5%82%A8%E6%8A%80%E6%9C%AF" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/_blank)是将数据分散地存储于多台独立的机器设备上。[分布式网络](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E7%BD%91%E7%BB%9C/8951687" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/_blank)[存储系统](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F/944115" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/_blank)采用可扩展的[系统结构](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E7%BB%93%E6%9E%84/10394712" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/_blank)，利用多台[存储服务器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E6%9C%8D%E5%8A%A1%E5%99%A8/1625801" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/_blank)分担存储负荷，利用位置服务器定位存储信息，不但解决了传统集中式存储系统中单存储服务器的瓶颈问题，还提高了[系统的可靠性](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E7%9A%84%E5%8F%AF%E9%9D%A0%E6%80%A7/5327458" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/_blank)、可用性和扩展性。

### 分布式限流

**限流**

在高并发系统中，需要使用多种方式来保护系统，例如：

缓存，目的是提升系统访问速度，增大系统处理能力。

降级，当服务器压力剧增的时候，根据实际业务情况，对某些服务或者页面有策略的不处理或者简单处理，从而释放服务器资源，以保证核心业务正常运行，通过牺牲局部利益来保证全局利益。

限流，目的是通过对请求进行限速来保护系统，如果达到限速值就可以采取一定的手段，例如拒绝服务、排队、等待。

所以，限流是保证系统高可用的重要手段。

**分布式限流**

单机版中我们了解到 AtomicInteger、RateLimiter、Semaphore 这几种解决方案，但它们也仅仅是单机的解决手段，在集群环境下就透心凉了，后面又讲述了 Nginx 的限流手段，可它又属于网关层面的策略之一，并不能解决所有问题。例如供短信接口，你无法保证消费方是否会做好限流控制，所以自己在应用层实现限流还是很有必要的。