注释

平时我们编写代码，在代码量比较少的时候，我们还可以看懂自己写的，但是当项目结构一旦复杂起来，我们就需要用到一个注释了，注释就类似于我们上学时候写的笔记，我们看着笔记就知道自己写的什么东西了！在程序中也是如此。我们来看一下Java中的注释怎么写，看以下代码：

```
/*
 * @Description HelloWorld类
 * @Author Diamond 狂神QQ:24736743
 **/
public class HelloWorld {
    /*
        这是我们Java程序的主入口，
        main方法也是程序的主线程。
    */
    public static void main(String[] args) {
        //输出HelloWorld!
        System.out.println("Hello,World!");
    }
}
```

注释并不会被执行，是给我们写代码的人看的，书写注释是一个非常好的习惯，在很多大公司都是强制要求各位去进行编写注释！比如，我们的BAT三大巨头等等。。。

Java中的注释有三种：

单行注释：只能注释当前行，以//开始，直到行结束

```
//输出HelloWorld!
```

多行注释：注释一段文字，以/*开始，*/结束！

```
/*
    这是我们Java程序的主入口，
    main方法也是程序的主线程。
*/
```

文档注释：用于生产API文档，配合JavaDoc。

【注】文档注释我们现在只作为了解，在学习JavaDoc时候我们会详细给大家说,目前知道有这样的注释就好。

```
/*
 * @Description HelloWorld类
 * @Author Diamond 狂神QQ:24736743
 **/
```

【狂神建议】平时写代码一定要注意注释的规范性，一个好的程序员，一定是有非常良好的编码习惯的，我希望大家能够从小事开始锻炼自己的行为习惯！

标识符

狂神社群笔记资料，禁止外传，本人QQ：24736743

每个人从出生开始就有一个名字，咋们生活中的所有事物也都有名字，这名字是谁规定呢？回答是：造物主，谁生产出来的谁规定名字，在我们的程序中也不例外。

我们作为造物主，需要给所有的东西给上一个名字，比如我们的HelloWorld程序：

HelloWorld是类名，也是我们的文件名。它前面的 public class是关键字，不过是搞Java那群人已经定义好的有特殊作用的，下面的每一个代码都有自己的意思和名字对吧，就是用来作区分！和我们的名字一样，拿来被叫或者称呼的，程序一切都源自于生活，一定要把学程序和生活中的一切联系起来，你会发现这一切都是息息相关的。

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello,World!");
    }
}
```

我们来看看有哪些是Java自己定义好的**关键字**呢?

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	strictfp	short	static	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

这些看起来非常的多，但是随着我们以后的学习我们都会用到，所以完全不用担心自己 看不懂，这些被Java已经规定的关键字，我们自己就不能拿它当做名字了！

Java 所有的组成部分都需要名字。类名、变量名以及方法名都被称为标识符。

我们自己起名字有哪些要求呢?

表示类名的标识符用大写字母开始。

```
1  如：Man, GoodMan
```

表示方法和变量的标识符用小写字母开始，后面的描述性词以大写开始。

```
1  如：eat(),eatFood()
2  //驼峰命名法
```

关于 Java 标识符，有以下几点需要注意：

- 所有的标识符都应该以字母（A-Z 或者 a-z）,美元符（$）、或者下划线（_）开始
- 首字符之后可以是字母（A-Z 或者 a-z）,美元符（$）、下划线（_）或数字的任何字符组合
- 不能使用关键字作为变量名或方法名。
- 标识符是大小写敏感的
- 合法标识符举例：age、$salary、_value、__1_value

- 非法标识符举例：123abc、-salary、#abc

【 JAVA不采用通常语言使用的ASCII字符集，而是采用unicode这样的标准的国际字符集。因此，这里的字母的含义：可以表示英文、汉字等等。】

【可以使用中文命名，但是一般不建议这样去使用，也不建议使用拼音，很Low】

```
public static void main(String[] args) {
    String 王者荣耀 = "最强王者";
    System.out.println(王者荣耀);
}
```

课外扩展：各种字符集和编码详解

演示：合法标识符以及不合法标识符

数据类型

Java是一种**强类型语言**，每个变量都必须声明其类型。

1、强弱类型语言

说到强类型语言，那什么是强类型语言呢?

强类型语言也称为强类型定义语言。要求变量的使用要严格符合规定，所有变量都必须先定义后才能使用。

Java、.NET、C++等都是强制类型定义的。也就是说，一旦一个变量被指定了某个数据类型，如果不经过转换，那么它就永远是这个数据类型了。

安全性高，运行效率相对较慢，鱼和熊掌不可兼得！强类型定义语言在速度上可能略逊色于弱类型定义语言，但是强类型定义语言带来的严谨性能够有效的避免许多错误。

与其相对应的是弱类型语言。

弱类型语言也称为弱类型定义语言。与强类型定义相反。像vb，php等就属于弱类型语言·

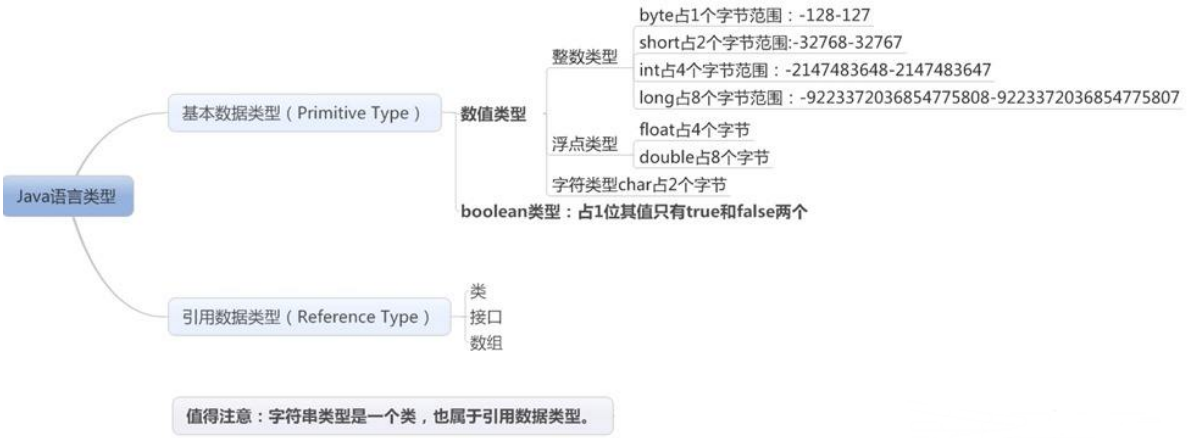
在VBScript中，可以将字符串'12'和整数3进行连接得到字符串'123'，也可以把它看成整数123，而不需要显示转换。是不是十分的随便，我们Java就不是这样的。

但其实它们的类型没有改变，VB只是在判断出一个表达式含有不同类型的变量之后，自动在这些变量前加了一个clong（）或（int）（）这样的转换函数而已。能做到这一点其实是归功于VB的编译器的智能化而已，这并非是VB语言本身的长处或短处。

好了，到这里，我们应该对强弱类型语言有了一定的了解！我们继续回到数据类型这个话题.

2、数据类型

Java的数据类型分为两大类：基本类型（primitive type）和引用类型 （reference type）



【注：引用数据类型的大小统一为4个字节，记录的是其引用对象的地址！】

基本数据类型			占内存	取值范围	默认值
数值型	整型	byte	1字节	-128 ~ 127	(byte)0
		short	2字节	-32768 ~ 32767	(short)0
		int	4字节	-2147483648 ~ 2147483647即-2^{15} ~ $2^{15}-1$	0
		long	8字节	-9223372036854775808L~9223372036854775807L 即-2^{31} ~ $2^{31}-1$	0L
	浮点型	float	4字节	对于负数-3.402823E38 ~ -1.401298E-45 对于正数1.401298E-45 ~ 3.402823E38	0.0F
		double	8字节	对于负数 -1.79769313486232E308 ~ -4.94065645841247E-324 对于正数 4.94065645841247E-324~1.79769313486232E308	0.0
字符型		char	2字节	unicode字符，用单引号括起来	'\u0000'
逻辑型		boolean	1字节	true，false	false

如果你看到这一堆头疼的话，没关系，不用记，JDK中类型对应的包装类都帮忙写好了，我们需要时候可以直接看到！可以把以下代码拷贝进行查看结果：

```
public static void main(String[] args) {
    // byte
    System.out.println("基本类型：byte 二进制位数：" + Byte.SIZE);
    System.out.println("包装类：java.lang.Byte");
    System.out.println("最小值：Byte.MIN_VALUE=" + Byte.MIN_VALUE);
    System.out.println("最大值：Byte.MAX_VALUE=" + Byte.MAX_VALUE);
    System.out.println();

    // short
    System.out.println("基本类型：short 二进制位数：" + Short.SIZE);
    System.out.println("包装类：java.lang.Short");
    System.out.println("最小值：Short.MIN_VALUE=" + Short.MIN_VALUE);
    System.out.println("最大值：Short.MAX_VALUE=" + Short.MAX_VALUE);
    System.out.println();

    // int
    System.out.println("基本类型：int 二进制位数：" + Integer.SIZE);
    System.out.println("包装类：java.lang.Integer");
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

```
        System.out.println("最小值：Integer.MIN_VALUE=" + Integer.MIN_VALUE);
        System.out.println("最大值：Integer.MAX_VALUE=" + Integer.MAX_VALUE);
        System.out.println();

        // long
        System.out.println("基本类型：long 二进制位数：" + Long.SIZE);
        System.out.println("包装类：java.lang.Long");
        System.out.println("最小值：Long.MIN_VALUE=" + Long.MIN_VALUE);
        System.out.println("最大值：Long.MAX_VALUE=" + Long.MAX_VALUE);
        System.out.println();

        // float
        System.out.println("基本类型：float 二进制位数：" + Float.SIZE);
        System.out.println("包装类：java.lang.Float");
        System.out.println("最小值：Float.MIN_VALUE=" + Float.MIN_VALUE);
        System.out.println("最大值：Float.MAX_VALUE=" + Float.MAX_VALUE);
        System.out.println();

        // double
        System.out.println("基本类型：double 二进制位数：" + Double.SIZE);
        System.out.println("包装类：java.lang.Double");
        System.out.println("最小值：Double.MIN_VALUE=" + Double.MIN_VALUE);
        System.out.println("最大值：Double.MAX_VALUE=" + Double.MAX_VALUE);
        System.out.println();

        // char
        System.out.println("基本类型：char 二进制位数：" + Character.SIZE);
        System.out.println("包装类：java.lang.Character");
        // 以数值形式而不是字符形式将Character.MIN_VALUE输出到控制台
        System.out.println("最小值：Character.MIN_VALUE="
                           + (int) Character.MIN_VALUE);
        // 以数值形式而不是字符形式将Character.MAX_VALUE输出到控制台
        System.out.println("最大值：Character.MAX_VALUE="
                           + (int) Character.MAX_VALUE);
    }
```

如果你是热爱学习的人，你现在应该非常想知道这字节到底是什么东西，所以我给大家科普一下相关知识：

```
/*
    位（bit）：是计算机 内部数据 储存的最小单位，11001100是一个八位二进制数。
    字节（byte）：是计算机中 数据处理 的基本单位，习惯上用大写 B 来表示，
        1B（byte,字节）= 8bit（位）
    字符：是指计算机中使用的字母、数字、字和符号

    ASCIIS码：
        1个英文字母（不分大小写）= 1个字节的空间
        1个中文汉字 = 2个字节的空间
        1个ASCII码 = 一个字节
    UTF-8编码：
        1个英文字符 = 1个字节
        英文标点 = 1个字节
        1个中文（含繁体） = 3个字节
        中文标点 = 3个字节
    Unicode编码：
        1个英文字符 = 2个字节
        英文标点 = 2个字节
        1个中文（含繁体） = 2个字节
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

```
        中文标点 = 2个字节

    1bit表示1位，
    1Byte表示一个字节 1B=8b。
    1024B=1KB
    1024KB=1M
    1024M=1G.
*/
```

那有人会问：电脑的32位和64位的区别是什么呢?

```
/*
    32位操作系统只可以使用32位的cpu，而64位的CPU既可以安装32位操作系统也可以安装64位操作系统。

    寻址能力简单点说就是支持的内存大小能力，64位系统最多可以支达128 GB的内存，而32位系统最多只可以支持4G内存。

    32位操作系统只可以安装使用32位架构设计的软件，而64位的CPU既可以安装使用32位软件也可以安装使用64位软件。

    现在的电脑都是64位了!
*/
```

【好了，回到正题，我们了解了这些知识后，我们自己定义一些**变量**来看！】

```
public static void main(String[] args) {

    //整型
    int i1=100;
    //长整型
    long i2=998877665544332211L;
    //短整型
    short i3=235;
    //浮点型
    double d1=3.5;  //双精度
    double d2=3;
    float f1=(float)3.5;   //单精度
    float f2=3.5f;   //单精度

    //布尔类型 boolean true真/false假
    boolean isPass=true;
    boolean isOk=false;
    boolean isBig=5>8;
    if(isPass){
        System.out.println("通过了");
    }else{
        System.out.println("未通过");
    }

    //单字符
    char f='女';
    char m='男';

}
```

【Java语言的整型常数默认为int型，浮点数默认是Double】

狂神社群笔记资料，禁止外传，本人QQ：24736743

3、整型拓展

在我们计算机中存在很多进制问题，十进制，八进制，十六进制等等的问题，他们怎么表示呢?

```
十进制整数，如：99，-500，0。

八进制整数，要求以 0 开头，如：015。

十六进制数，要求 0x 或 0X 开头，如：0x15 。
```

演示：

```
//整型
int i=10;
int i2=010;
int i3=0x10;
System.out.println(i); //10
System.out.println(i2); //8
System.out.println(i3); //16
```

4、浮点型拓展

【金融面试问：银行金融业务用什么类型表示？】

浮点类型float, double的数据不适合在不容许舍入误差的金融计算领域。
如果需要进行不产生舍入误差的精确数字计算，需要使用**BigDecimal**类。

```
public static void main(String[] args) {
    float f = 0.1f;
    double d = 1.0/10;
    System.out.println(f==d);   //false

    float d1 = 2131231231f;
    float d2 = d1+1;
    if(d1==d2){
        System.out.println("d1==d2");
    }else{
        System.out.println("d1!=d2");
    }
}
```

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，其结果只能是接近，但不等于；二进制浮点数不能精确的表示0.1,0.01,0.001这样10的负次幂。并不是所有的小数都能可以精确的用二进制浮点数表示。

最好完全避免使用浮点数比较！

大数值：Java.math下面的两个有用的类：BigInteger和BigDecimal，这两个类可以处理任意长度的数值。BigInteger实现了任意精度的整数运算。BigDecimal实现了任意精度的浮点运算。

浮点数使用总结：

1. 默认是double
2. 浮点数存在舍入误差，很多数字不能精确表示。如果需要进行不产生舍入误差的精确数字计算，需要使用BigDecimal类。
3. 避免比较中使用浮点数

狂神社群笔记资料，禁止外传，本人QQ：24736743

5、字符型拓展

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"表示一个字符串。

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；

【科普：2字节=16位 2的16次方=65536，我们用的Excel原来就只有这么多行，并不是无限的】

【代码演示：字符转int看结果】

```
public static void main(String[] args) {
    char c1 = 'a';
    char c2 = '中';
    System.out.println(c1);
    System.out.println((int) c1);   //97
    System.out.println(c2);
    System.out.println((int) c2);   //20013
}
```

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 c3 = '\u0061';
System.out.println(c3);   //a
```

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，有如下常用转义字符：

\b	退格（backspace）	\u0008
\n	换行	\u000a
\r	回车	\u000d
\t	制表符（tab）	\u0009
\ "	双引号	\u0022
\ '	单引号	\u0027
\\	反斜杠	\u005c

【以后我们学的String类，其实是字符序列(char sequence)。在这里给大家一个思考题】

```java
//代码1
String sa=new String("Hello world");
String sb=new String("Hello world");
System.out.println(sa==sb);  // false
//代码2
String sc="Hello world";
String sd="Hello world";
System.out.println(sc==sd);  // true
```

大家可以先思考下为什么，之后我们学到对象的时候，会给大家进行内存级别的分析，那时候你会恍然大悟！

6、布尔型拓展

boolean类型（一位，不是一个字节），就是0|1

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 类型用来判断逻辑条件，一般用于程序流程控制。

```java
boolean flag = false;
if(flag){
  // true分支
}else{
  // false分支
}
```

【编码规范：很多新手程序员喜欢这样写】

```java
if (is == true && un == false ) {...}
```

只有新手才那么写。对于一个熟练的人来说，应该用如下方式来表示：

```java
if ( is && !un ) {....}
```

这点都不难理解吧。所以要习惯去掉所有的==fasle 和 ==true。Less is More！！ 代码要精简易读！

类型转换

由于Java是强类型语言，所以要进行有些运算的时候的，需要用到类型转换。

整型、实型（常量）、字符型数据可以混合运算。

运算中，不同类型的数据先转化为同一类型，然后进行运算。

转换从低级到高级（根据容量来看）。

```
低  ------------------------------------->  高

byte,short,char-> int -> long-> float -> double
```

数据类型转换必须满足如下规则：

- 不能对boolean类型进行类型转换。
- 不能把对象类型转换成不相关类的对象。
- 在把容量大的类型转换为容量小的类型时必须使用强制类型转换。
- 转换过程中可能导致溢出或损失精度，例如：

狂神社群笔记资料，禁止外传，本人QQ：24736743

```
int i =128;
byte b = (byte)i;
```

因为 byte 类型是 8 位，最大值为127，所以当 int 强制转换为 byte 类型时，值 128 时候就会导致溢出。

- 浮点数到整数的转换是通过舍弃小数得到，而不是四舍五入，例如：

```
(int)23.7 == 23;
(int)-45.89f == -45
```

1、自动类型转换

自动类型转换：容量小的数据类型可以自动转换为容量大的数据类型。

例如: short数据类型的位数为16位，就可以自动转换位数为32的int类型，同样float数据类型的位数为32，可以自动转换为64位的double类型。

【演示】

```
public class ZiDongLeiZhuan{
    public static void main(String[] args){
        char c1='a';//定义一个char类型
        int i1 = c1;//char自动类型转换为int
        System.out.println("char自动类型转换为int后的值等于"+i1);
        char c2 = 'A';//定义一个char类型
        int i2 = c2+1;//char 类型和 int 类型计算
        System.out.println("char类型和int计算后的值等于"+i2);
    }
}
```

【解析：c1 的值为字符 **a** ,查 ASCII 码表可知对应的 int 类型值为 97， A 对应值为 65，所以 **i2=65+1=66。】**

2、强制类型转换

强制类型转换，又被称为造型，用于显式的转换一个数值的类型.

在有可能丢失信息的情况下进行的转换是通过造型来完成的，但可能造成精度降低或溢出。

强制类型转换的语法格式：`(type)var`，运算符“()”中的type表示将值var想要转换成的目标数据类型。 条件是转换的数据类型必须是兼容的。

【演示】

```
public static void main(String[] args) {
    double x  = 3.14;
    int nx = (int)x;   //值为3

    char c = 'a';
    int d = c+1;
    System.out.println(d); //98
    System.out.println((char)d);  //b
}
```

当将一种类型强制转换成另一种类型，而又超出了目标类型的表示范围，就会被截断成为一个完全不同的值，溢出。

```java
public static void main(String[] args) {
    int x = 300;
    byte bx = (byte)x;    //值为44
    System.out.println(bx);
}
```

3、常见错误和问题

1. 操作比较大的数时，要留意是否溢出，尤其是整数操作时；

```java
public static void main(String[] args) {
    int money = 1000000000;  //10亿
    int years = 20;
    int total = money*years;   //返回的是负数
    long total1 = money*years;  //返回的仍然是负数。默认是int，因此结果会转成int值，再转成long。但是已经发生了数据丢失
    long total2 = money*((long)years);   //先将一个因子变成long，整个表达式发生提升。全部用long来计算。
    System.out.println(total);
    System.out.println(total1);
    System.out.println(total2);
}
```

2. L和l 的问题：
   - 不要命名名字为l的变量
   - long类型使用大写L不要用小写。

```java
public static void main(String[] args) {
    int l = 2;
    long a = 23451l;
    System.out.println(l+1); //3
    System.out.println(a);   //23451
}
```

4、JDK7扩展

JDK7新特性: 二进制整数

由于我们在开发中也经常使用二进制整数，因此JDK7为我们直接提供了二进制整数的类型。

我们只要以：0b开头即可。

```java
int a = 0b0101;
```

JDK7新特性：下划线分隔符

在实际开发和学习中，如果遇到特别长的数字，读懂它令人头疼！JDK7为我们提供了下划线分隔符，可以按照自己的习惯进行分割。

```java
int b = 1_2234_5678;
```

我们很容易就知道这是1亿2234万5678啦！ 非常符合国人的习惯！

狂神社群笔记资料，禁止外传，本人QQ：24736743

```java
public static void main(String[] args) {
    int a = 0b0101;
    int b = 1_2345_7893;
    System.out.println(a);  //5
    System.out.println(b);  //123457893
}
```

变量，常量

1、变量（variable)

变量是什么：就是可以变化的量!

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

这就好像我们家里有一个大衣柜，里面有十分多的小格子，我们给格子上贴上标签，放衣服，放鞋子，放手表等等，此时我们知道了哪里该放什么，但是，我们并不知道里面到底放的是什么牌子的鞋子，是衣服还是裤子。那个标签就相当于我们的变量，我们给他起了个名字，但是里面要放什么需要我们自己去放。

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]}] ;
//数据类型  变量名 = 值；可以使用逗号隔开来声明多个同类型变量。
```

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。
- 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

【演示】

```java
int a, b, c;         // 声明三个int型整数：a、 b、c
int d = 3, e = 4, f = 5; // 声明三个整数并赋予初值
byte z = 22;         // 声明并初始化 z
String s = "runoob";  // 声明并初始化字符串 s
double pi = 3.14159; // 声明了双精度浮点型变量 pi
char x = 'x';        // 声明变量 x 的值是字符 'x'。
```

【编码规范】

虽然可以在一行声明多个变量，但是不提倡这个风格，逐一声明每一个变量可以提高程序可读性。

2、变量作用域

变量根据作用域可划分为三种：

- 类变量（静态变量： static variable）：独立于方法之外的变量，用 static 修饰。
- 实例变量（成员变量：member variable）：独立于方法之外的变量，不过没有 static 修饰。
- 局部变量（lacal variable）：类的方法中的变量。

狂神社群笔记资料，禁止外传，本人QQ：24736743

```java
public class Variable{
    static int allClicks=0;    // 类变量
    String str="hello world";  // 实例变量

    public void method(){
        int i =0;  // 局部变量
    }
}
```

局部变量

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)。

局部变量没有默认值，所以局部变量被声明后，必须经过初始化，才可以使用。

```java
public static void main(String[] args) {
    int i;
    int j = i+5 ; // 编译出错，变量i还未被初始化
    System.out.println(j);
}
```

修改为：

```java
public static void main(String[] args) {
    int i=10;
    int j = i+5 ;
    System.out.println(j);
}
```

实例变量

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。

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）

```java
public class Test {
    // 这个实例变量对子类可见
    public String name;
    // 私有变量，仅在该类可见
    private double salary;
    ...
}
```

静态变量

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。

(注：讲完内存分析后我们再深入！先放一放这个概念！)

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）

狂神社群笔记资料，禁止外传，本人QQ：24736743

```java
public class Employee {
    //salary是静态的私有变量
    private static double salary;
    // DEPARTMENT是一个常量
    public static final String DEPARTMENT = "开发人员";
    public static void main(String[] args){
        salary = 10000;
        System.out.println(DEPARTMENT+"平均工资:"+salary);
    }
}
```

3、常量

常量(Constant)：初始化(initialize)后不能再改变值！不会变动的值。

所谓常量可以理解成一种特殊的变量，它的值被设定后，在程序运行过程中不允许被改变。

```
final 常量名=值;
final double PI=3.14; final String LOVE="hello";
```

常量名一般使用大写字符。
程序中使用常量可以提高代码的可维护性。例如，在项目开发时，我们需要指定用户的性别，此时可以定义一个常量 SEX，赋值为 "男"，在需要指定用户性别的地方直接调用此常量即可，避免了由于用户的不规范赋值导致程序出错的情况。

4、变量的命名规范

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()

运算符

运算符operator

Java 语言支持如下运算符：

- 算术运算符: +，-，*，/，%，++，--
- 赋值运算符 =
- 关系运算符: >，<，>=，<=，==，!= instanceof
- 逻辑运算符: &&，||，!
- 位运算符: &，|，^，~ ，>>，<<，>>> (了解！！！)
- 条件运算符 ？ ：
- 扩展赋值运算符:+=，-=，*=，/=

1、二元运算符

两个操作数，来看看我们小时候的数学运算；

```java
public static void main(String[] args) {
    int a = 10;
    int b = 20;
    int c = 25;
    int d = 25;
    System.out.println("a + b = " + (a + b) );
    System.out.println("a - b = " + (a - b) );
    System.out.println("a * b = " + (a * b) );
    System.out.println("b / a = " + (b / a) );
}
```

整数运算

如果两个操作数有一个为Long, 则结果也为long

没有long时，结果为int。即使操作数全为shot,byte，结果也是int.

```java
public static void main(String[] args) {
    long a = 1231321311231231L;
    int b = 1213;
    short c = 10;
    byte d = 8;

    System.out.println((a+b+c+d));  //Long类型
    System.out.println((b + c + d));//Int类型
    System.out.println((c + d));//Int类型
}
```

浮点运算

如果两个操作数有一个为double, 则结果为double.

只有两个操作数都是float, 则结果才为float.

```java
public static void main(String[] args) {
    float a = 3.14565F;
    double b = 3.194546464;
    float c = 1.3123123F;

    System.out.println(a+b); //double类型
    System.out.println(b+c); //double类型
    System.out.println(a+c); //float类型
}
```

关系运算符

返回布尔值!

运算符	描述	例子
==	检查如果两个操作数的值是否相等，如果相等则条件为真。	(A == B) 为假。
!=	检查如果两个操作数的值是否相等，如果值不相等则条件为真。	(A != B) 为真。
>	检查左操作数的值是否大于右操作数的值，如果是那么条件为真。	(A> B) 为假。
<	检查左操作数的值是否小于右操作数的值，如果是那么条件为真。	(A <B) 为真。
>=	检查左操作数的值是否大于或等于右操作数的值，如果是那么条件为真。	(A>= B) 为假。
<=	检查左操作数的值是否小于或等于右操作数的值，如果是那么条件为真。	(A <= B) 为真。

狂神社群笔记资料，禁止外传，本人QQ：24736743

2、取模运算

就是我们小学的取余；`5%3 余 2`

其操作数可以为浮点数,一般使用整数。如：5.9%3.9=2.000000004

要点:

负数%负数＝负数;
负数%正数＝负数;
正数%负数＝正数;

```
public static void main(String[] args) {
    System.out.println(9 % 4); //1
    System.out.println(-9 % -4); //-1
    System.out.println(-10 % 4); //-2
    System.out.println(9 % -4); //1
}
```

【注：一般都是正整数运算，进行结果的判断！】

3、一元运算符

自增（++）自减（--）运算符是一种特殊的算术运算符，在算术运算符中需要两个操作数来进行运算，而自增自减运算符是一个操作数，分为前缀和后缀两种。

```
public static void main(String[] args) {
    int a = 3;
    int b = a++;    //执行完后,b=3。先给b赋值，再自增。
    int c = ++a;    //执行完后,c=5。先自增,再给b赋值
}
```

注意：java中的乘幂处理

```
public static void main(String[] args) {
    int a = 3^2;    //java中不能这么处理，  ^是异或符号。
    double b = Math.pow(3, 2);
}
```

Math类提供了很多科学和工程计算需要的方法和常数。特殊的运算都需要运用到方法!

4、逻辑运算符

逻辑与：&&和&，逻辑或：||和|，逻辑非：！。

操作符	描述	例子
&&	称为逻辑与运算符。当且仅当两个操作数都为真，条件才为真。	（A && B）为假。
||	称为逻辑或操作符。如果任何两个操作数任何一个为真，条件为真。	（A || B）为真。
!	称为逻辑非运算符。用来反转操作数的逻辑状态。如果条件为true，则逻辑非运算符将得到false。	！（A && B）为真。

【演示】

```
public static void main(String[] args) {
    boolean a = true;
    boolean b = false;
    System.out.println("a && b = " + (a&&b));
    System.out.println("a || b = " + (a||b) );
    System.out.println("!(a && b) = " + !(a && b));
}
```

逻辑与和逻辑或采用**短路的方式**。从左到右计算，如果确定值则不会再计算下去。在两个操作数都为true时，结果才为true，但是当得到第一个操作为false时，其结果就必定是false，这时候就不会再判断第二个操作了。

逻辑与只要有一个为false, 则直接返回false.

逻辑或只要有一个为true, 则直接返回true;

【演示】

```
public static void main(String[] args){
    int a = 5;//定义一个变量;
    boolean b = (a<4)&&(a++<10);
    System.out.println("使用短路逻辑运算符的结果为"+b);
    System.out.println("a的结果为"+a);
}
```

解析： *该程序使用到了短路逻辑运算符(&&)，首先判断 a<4 的结果为 false，则 b 的结果必定是 false，所以不再执行第二个操作 a++<10 的判断，所以 a 的值为 5。*

5、位运算符

Java定义了位运算符，应用于整数类型(int)，长整型(long)，短整型(short)，字符型(char)，和字节型(byte)等类型。位运算符作用在所有的位上，并且按位运算。

```
A = 0011 1100
B = 0000 1101
-----------------
A&b = 0000 1100
A | B = 0011 1101
A ^ B = 0011 0001
~A= 1100 0011
```

操作符	描述	例子
&	如果相对应位都是1，则结果为1，否则为0	（A&B），得到12，即0000 1100
|	如果相对应位都是0，则结果为0，否则为1	（A | B）得到61，即 0011 1101
^	如果相对应位值相同，则结果为0，否则为1	（A ^ B）得到49，即 0011 0001
~	按位取反运算符翻转操作数的每一位，即0变成1，1变成0。	（~A）得到-61，即1100 0011
<<	按位左移运算符。左操作数按位左移右操作数指定的位数。	A << 2得到240，即 1111 0000
>>	按位右移运算符。左操作数按位右移右操作数指定的位数。	A >> 2得到15即 1111
>>>	按位右移补零操作符。左操作数的值按右操作数指定的位数右移，移动得到的空位以零填充。	A>>>2得到15即0000 1111

右移一位相当于除2取商。

左移一位相当于乘2。

【常见面试题：int a=2*8怎样运算效率最快？】

解析：

```
public static void main(String[] args) {
    System.out.println(2 << 3);
}
```

用移位运算 int a=2<<3;
a就是2乘以8 最后结果是16 这是最省内存 最有效率的方法

这个方法确实高效率的。我来解释一下：
2的二进制是10 在32位存储器里面是0000 0000 0000 0010
左移三位后变成 0000 0000 0001 0000 也就是16

解释一下，在系统中运算是以二进制的形式进行的。相比来说俩个二进制数相乘运算比移位运算慢一些。

位操作是程序设计中对位模式按位或二进制数的一元和二元操作。 在许多古老的微处理器上， 位运算比加减运算略快， 通常位运算比乘除法运算要快很多。 在现代架构中， 情况并非如此:位运算的运算速度通常与加法运算相同(仍然快于乘法运算). **详细的需要了解计算机的组成原理！**

6、扩展运算符

运算符	用法举例	等效的表达式
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

```
public static void main(String[] args) {
    int a=10;
    int b=20;

    a+=b; // a = a + b

    System.out.println(a+":"+b);
}
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

7、字符串连接符

“+” 运算符两侧的操作数中**只要有一个是字符串**(String)类型，系统会自动将另一个操作数转换为字符串然后再进行连接。

```
//字符串
String s1="Hello 中文!";
String s2=1+"";  //转换成String
//int
int c = 12;
System.out.println("c=" + c);
```

8、三目条件运算符

三目条件运算符，语法格式：

```
x ? y : z
```

其中x为boolean类型表达式，先计算x的值，若为true，则整个三目运算的结果为表达式y的值，否则整个运算结果为表达式z的值。

【演示】

```
public static void main(String[] args) {
    int score = 80;
    String type = score < 60 ? "不及格" : "及格";
    System.out.println("type= " + type);
}
```

三元运算符在真实开发中十分的常见，大家可以多练习使用，之后我们会讲解分支语句，可以利用三元运算符做到更加精简代码！便于理解！

9、运算符优先级

我们小学都学过：先加减，后乘除，所以优先级我们并不陌生。

当多个运算符出现在一个表达式中，谁先谁后呢？这就涉及到运算符的优先级别的问题。在一个多运算符的表达式中，运算符优先级不同会导致最后得出的结果差别甚大。

下表中具有最高优先级的运算符在的表的最上面，最低优先级的在表的底部。

狂神社群笔记资料，禁止外传，本人QQ：24736743

类别	操作符	关联性
后缀	() [] . (点操作符)	左到右
一元	+ + - ！~	从右到左
乘性	* /%	左到右
加性	+ -	左到右
移位	>> >>> <<	左到右
关系	>> = << =	左到右
相等	== !=	左到右
按位与	&	左到右
按位异或	^	左到右
按位或	|	左到右
逻辑与	&&	左到右
逻辑或	||	左到右
条件	？：	从右到左
赋值	= + = - = * = / =%= >> = << =&= ^ = | =	从右到左

大家不需要去刻意的记住，表达式里面优先使用小括号来组织！！方便理解和使用，不建议写非常冗余的代码运算！

```
public static void main(String[] args) {
    boolean flag = 1<4*5&&122>3||'q'+3<5;
    System.out.println(flag);
}
```

包机制

1、问题发现

存在这样一个问题：当定义了多个类的时候，可能会发生类名的重复问题。

解决方式：在java中采用包机制处理开发者定义的类名冲突问题。

就好比我们平时的用电脑，一个文件夹下不能存在同名的文件，我们要是有这样的需求，但是又不想换名字，我们就可以考虑使用新建一个文件夹来存放！在我们的java中也是这样的。

【演示：重名文件】

还有一个问题：我们平时在IDE中可以跑的文件，用命令行就会报错；

【演示：HelloWorld！IDE和命令行】

这是为什么呢?
罪魁祸首正是代码第一行：package com.kuang.demo01;

这就是Java中的包机制，使用package com.kuang.demo01;

就要求此份.java文件必须保存在这样一个目录下，这样Java解释器才能找到它。 在IDEA中能正确运行，你可以去Windows下的工程中查看，HelloWorld这个文件必是在这样的目录结构下的。

2、包的作用

狂神社群笔记资料，禁止外传，本人QQ：24736743

为了更好地组织类，Java 提供了包机制，用于区别类名的命名空间。

包的作用：

1、把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。

2、如同文件夹一样，包也采用了树形目录的存储方式。同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。

3、包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

Java 使用包（package）这种机制是为了防止命名冲突，访问控制，提供搜索和定位类（class）、接口、枚举（enumerations）和注释（annotation）等。

包语句的语法格式为：

```
package pkg1[. pkg2[. pkg3…]];
```

例如,一个Something.java 文件它的内容:

```
package net.java.util;
public class Something{
    ...
}
```

那么它的路径应该是 **net/java/util/Something.java** 这样保存的。 package(包) 的作用是把不同的 java 程序分类保存，更方便的被其他 java 程序调用。

一个包（package）可以定义为一组相互联系的类型（类、接口、枚举和注释），为这些类型提供访问保护和命名空间管理的功能。

以下是一些 Java 中的包：

- **java.lang**-打包基础的类
- **java.io**-包含输入输出功能的函数

开发者可以自己把一组类和接口等打包，并定义自己的包。而且在实际开发中这样做是值得提倡的，当你自己完成类的实现之后，将相关的类分组，可以让其他的编程者更容易地确定哪些类、接口、枚举和注释等是相关的。

由于包创建了新的命名空间（namespace），所以不会跟其他包中的任何名字产生命名冲突。使用包这种机制，更容易实现访问控制，并且让定位相关类更加简单。

3、创建包

创建包的时候，你需要为这个包取一个合适的名字。之后，如果其他的一个源文件包含了这个包提供的类、接口、枚举或者注释类型的时候，都必须将这个包的声明放在这个源文件的开头。

包声明应该在源文件的第一行，每个源文件只能有一个包声明，这个文件中的每个类型都应用于它。

如果一个源文件中没有使用包声明，那么其中的类，函数，枚举，注释等将被放在一个无名的包（unnamed package）中。

一般利用公司域名倒置作为报名；

例子：

www.baidu.com 包名：com.baidu.www

bbs.baidu.com 包名：com.baidu.bbs

狂神社群笔记资料，禁止外传，本人QQ：24736743

blog.baidu.com 包名：com.baidu.blog

我们平时也可以按照自己的公司域名去写，比如：com.kuangstudy.utils

4、import 关键字

为了能够使用某一个包的成员，我们需要在 Java 程序中明确导入该包。使用 "import" 语句可完成此功能。

在 java 源文件中 import 语句应位于 package 语句之后，所有类的定义之前，可以没有，也可以有多条，其语法格式为：

```
import package1[.package2…].(classname|*);
```

如果在一个包中，一个类想要使用本包中的另一个类，那么该包名可以省略。

要是要用到其他包下的类，就必须要先导包！

如果两个类重名，需要导入对应的包，否则就需要写出完整地址：

```
com.kuang.dao.Hello hello = new com.kuang.dao.Hello()
```

用 **import** 关键字引入，使用通配符 "*"，导入io包下的所有类！

```
import java.io.*;
```

【不建议这样使用，因为会全局扫描，影响速度！】

使用 **import** 关键字引入指定类:

```
import com.kuang.Hello;
```

【注意】类文件中可以包含任意数量的 import 声明。import 声明必须在包声明之后，类声明之前。

【编码规范：推荐参考阿里巴巴开发手册编程规范】

下载地址：阿里巴巴开发手册

JavaDoc

1、简介

JavaDoc是一种将注释生成HTML文档的技术，生成的HTML文档类似于Java的API，易读且清晰明了。在简略介绍JavaDoc写法之后，再看一下在Intellij Idea 中如何将代码中的注释生成HTML文档。

javadoc是Sun公司提供的一个技术，它从程序源代码中抽取类、方法、成员等注释形成一个和源代码配套的API帮助文档。也就是说，只要在编写程序时以一套特定的标签作注释，在程序编写完成后，通过Javadoc就可以同时形成程序的开发文档了。javadoc命令是用来生成自己API文档的，使用方式：使用命令行在目标文件所在目录输入javadoc +文件名.java。

先看一段样例代码：

```
/** 这是一个Javadoc测试程序
 *  @author Kuangshen
 *  @version 1.0
 *  @since 1.5
 * */
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

```java
public class HelloWorld {

    public String name;
    /**
     * @param name 姓名
     * @return 返回name姓名
     * @throws  Exception 无异常抛出
     * */
    public String function(String name) throws Exception{
        return name;
    }
}
```

稍微解释一下：

以 */* 开始，以 */* 结束。

@author 作者名

@version 版本号

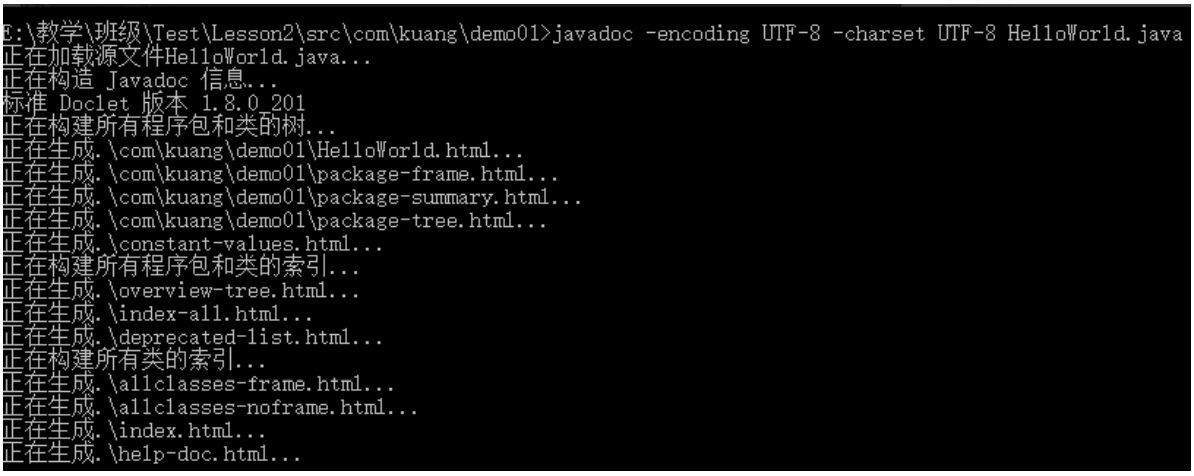
@since 指明需要最早使用的jdk版本

@param 参数名

@return 返回值情况

@throws 异常抛出情况

2、命令行生成Doc




```
-encoding UTF-8 -charset UTF-8
//解决GBK乱码问题，在中间添加编码设置
```

【演示：生成并查看文档】

回顾及总结

这一章，我们学习了Java的基础

- 安装使用了IDEA
- 使用注释
- 了解了Java的关键字
- 数据类型
- 怎么定义一个变量

狂神社群笔记资料，禁止外传，本人QQ：24736743

- 怎么操作这些变量
- 如何生成一篇完整的文档
- 。。。

以后这些东西在我们的生活中会天天遇到！

不积跬步，无以至千里；不积小流，无以成江海！

一生二，二生三，三生万物！

一切一切都说明了基础的重要性！

所以，简单的东西，不要忽略，高手之间的区分，就在于这些细节方面的东西！