笔记总链接：http://bbs.itheima.com/thread-200600-1-1.html

一、Java语言基础组成-Part 2

1.6 运算符

1.6.3 比较运算符

比较运算符			
运算符	运算	范例	结果
==	相等于	4==3	false
!=	不等于	4 != 3	true
<	小于	4 < 3	false
>	大于	4 > 3	true
<=	小于等于	4 <= 3	false
>=	大于等于	4 >= 3	true
instanceof	检查是否是类的对象	"Hello" instanceof String	true

P.S.

1、比较运算符的结果都是boolean型，也就是要么是true，要么是false。

2、比较运算符“==”不能误写成“=” 。

示例1：

```
class OperatorDemo
{
    public static void main(String[] args){
        System.out.println( 3 > 2 );
        System.out.println( 3 == 2 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
true
false
```

1.6.4 逻辑运算符

逻辑运算符			
运算符	运算	范例	结果
&	AND（与）	false&true	false
|	OR（或）	false|true	true
^	XOR（异或）	true^false	true
!	NOT（非）	!true	false
&&	AND（短路）	false&&true	false
||	OR（短路）	false||true	true

逻辑运算符用于连接两个boolean类型的表达式。

"&"符号的运算特点：

true & true = true;

true & false = false;

false & true = false;

false & false = false;

"&"符号运算规律：

运算的两边只要有一个是false，结果肯定是false。

只有两边都为true，结果才是true。

示例1：

```
class OperatorDemo
{
    public static void main(String[] args){
        int x = 6;
        System.out.println( x > 2 & x < 5 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
false
```

"|"符号的运算特点：

true | true = true;

true | false = true;

false | true = true;

false | false = false;

"|"符号运算规律：

运算的两边只要有一个是true，结果肯定是true。

只有两边都为false，结果是false。

示例2：

```
class OperatorDemo
{
    public static void main(String[] args){
        int x = 3;
        System.out.println( x > 2 | x < 5 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
true
```

"^"符号的运算特点：

true ^ true = false;

true ^ false = true;

false ^ true = true;

false ^ false = false;

"^"符号运算规律：

^符号的两边结果如果相同，结果是false。

两边的结果不同，结果是true。

示例3：

```
class OperatorDemo
{
    public static void main(String[] args){
        int x = 3;
        System.out.println( x > 2 ^ x < 5 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
false
```

"！"符号运算规律：

true = false

!false = true

!!true = true

示例4：

```
class OperatorDemo
{
    public static void main(String[] args){
        int x = 3;
        System.out.println( ! ( x > 2 ) );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
false
```

P.S.

&&：和&运算的结果是一样的，但是运算过程有点小区别。

&：无论左边的运算结果是什么，右边都参与运算。

&&：当左边为false时，右边不参加运算，这样可以提升效率。

||：和|运算的结果是一样的，但是运算过程有点小区别。

|：无论左边的运算结果是什么，右边都参与运算。

||：当左边为true时，右边不参加运算，这样可以提升效率。

使用&&和||比使用&和|更高效一些。

逻辑运算符用于连接布尔型表达式，在Java中不可以写成3<x<6，应该写成x>3 & x<6 。

1.6.5 位运算符

位运算符		
运算符	运算	范例
<<	左移	3<<2=12-->3*2*2=12
>>	右移	3>>1=1-->3/2=1
>>>	无符号右移	3>>>1=1-->3/2=1
&	与运算	6&3=2
|	或运算	6 | 3=7
^	异或运算	6^3=5
~	反码	~6=-7

位运算符的细节	
<<	空位补0，被移除的高位丢弃，空缺位补0。
>>	被移位的二进制最高位是0，右移后，空缺位补0； 最高位是1，空缺位补1。
>>>	被移位二进制最高位无论是0或者是1，空缺位都用0补。
&	二进制位进行&运算，只有1&1的结果是1，否则是0。
|	二进制位进行|运算，只有0|0时结果是0，否则是1。
^	相同二进制位进行^运算，结果是0：1^1=0，0^0=0； 不相同二进制位^运算结果是1：1^0=1，0^1=1。

P.S.

位运算是直接对二进制位进行运算。

与运算例子：

6 & 3 = 2

110

& 011

010

示例：

```
class OperatorDemo
{
    public static void main(String[] args){
        System.out.println( 6 & 3 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
2
```

P.S.

用“与运算”可以很方便提取某一二进制数的其中几位数。

例如：取17476二进制数的后四位。

  0100010001000100

& 0000000000001111

  0000000000000100

或运算例子：

6 | 3 = 7

110

| 011

111

示例：

```
class OperatorDemo
{
    public static void main(String[] args){
        System.out.println( 6 | 3 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
7
```

异或运算例子：

6 ^ 3 = 5

110

^ 011

101

示例：

```
class OperatorDemo
{
    public static void main(String[] args){
        System.out.println( 6 ^ 3 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
5
```

P.S.

一个数异或同一个数两次，结果还是这个数。

例子：

6 ^ 3 ^ 3 = 6

6 ^ 3 = 5

110

^ 011

101

^ 011

110

技巧：

利用异或运算可以实现对数据简单地进行加密，例如对一幅图片的所有数据异或3进行加密，那么这幅图片就无法查看了。解密只需要再对图片的数据执行异或3操作即可。

取反运算：

取反操作就是对二进制数值的每一位0变1，1变0。

示例：

```
class OperatorDemo
{
    public static void main(String[] args){
        System.out.println( ~6 );
    }
}
```
复制代码

运算结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
-7
```

左移运算例子：

3 << 2 = 12 相当于 3 * 2（2） = 12

00000000000000000000000000000011

00000000000000000000000000001100

3 << 3 = 24 相当于 3 * 2（3） = 24

00000000000000000000000000000011

00000000000000000000000000011000

示例：

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

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
12
24
```

总结：

左移几位其实就是该数据乘以2的几次方。

<<：可以完成2的次幂运算。

右移运算例子：

6 >> 1 = 3 相当于 6/2（1）=3

00000000000000000000000000000110

00000000000000000000000000000011

6 >> 2 = 1 相当于 6/2（2） = 1

00000000000000000000000000000110

00000000000000000000000000000001

示例：

```
class OperatorDemo
{
    public static void main(String[] args){
        System.out.println( 6 >> 1 );
        System.out.println( 6 >> 2 );
    }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day02>javac OperatorDemo.java

D:\code\day02>java OperatorDemo
3
1
```

总结：

右移几位其实就是该数据除以2的几次幂。

附：

以上例子中括号表示次幂的意思。

P.S.

>>：对于高位出现的空位，原来高位是什么，就用什么补这个空位。

>>>：无符号右移，数据进行右移时，高位出现的空位，无论原高位是什么，空位都用0补。

练习：

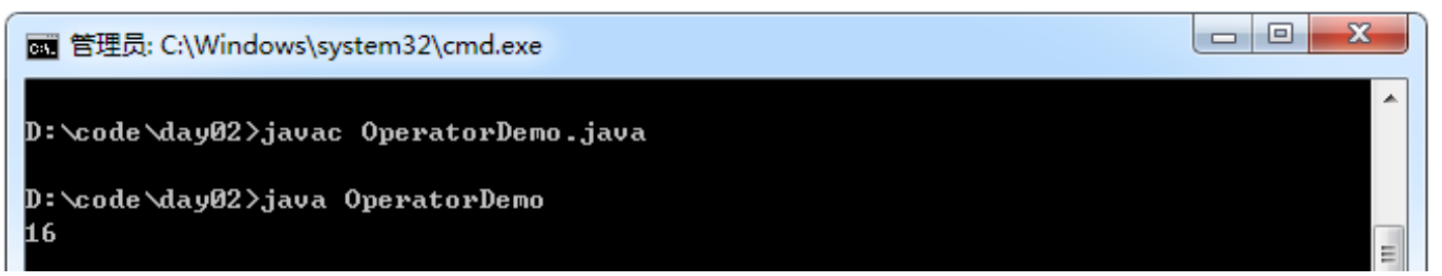
1. 最有效率的方式算出2乘以8等于几？

答案：

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

复制代码

运行结果：



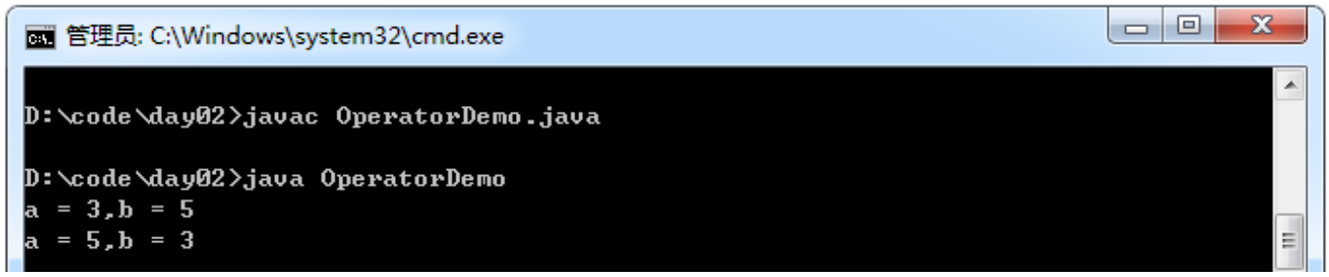
2. 对两个整数变量的值进行互换(可以使用第三方变量)。

答案：

```
class OperatorDemo
{
    public static void main(String[] args){
        int a = 3,b = 5;

        //开发时，使用第三方变量的形式，因为阅读性强。
        int c;

        System.out.println( "a = " + a + ",b = " + b );

        c = a;
        a = b;
        b = c;

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

复制代码

运行结果：



3. 对两个整数变量的值进行互换(不可以使用第三方变量)。

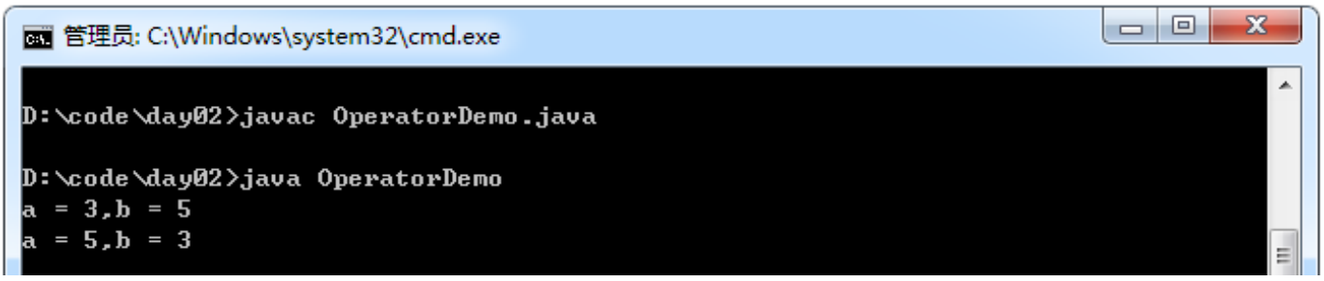
答案1：

```
class OperatorDemo
{
    public static void main(String[] args){
        int a = 3,b = 5;
        int c;

        System.out.println( "a = " + a + ",b = " + b );

        //这种方式不要用，如果两个整数的数值过大，会超出int范围，会强制转换，数据会变化。
        a = a + b;
        b = a - b;
        a = a - b;
```

```
13.
14.             System.out.println( "a = " + a + ",b = " + b );
15.         }
16.     }
```

复制代码

运行结果：

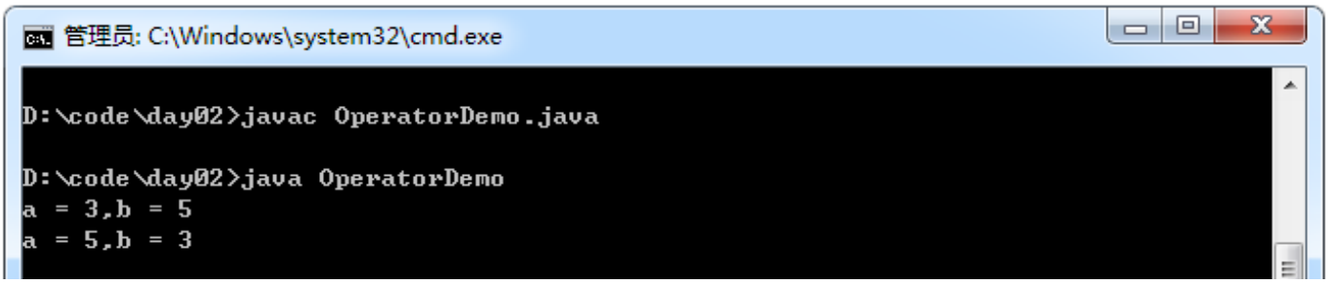


答案2：

```
01.   class OperatorDemo
02.   {
03.       public static void main(String[] args){
04.           int a = 3,b = 5;
05.           int c;
06.
07.           System.out.println( "a = " + a + ",b = " + b );
08.
09.           a = a ^ b;
10.           b = a ^ b;
11.           a = a ^ b;
12.
13.          System.out.println( "a = " + a + ",b = " + b );
14.       }
15.   }
```

复制代码

运行结果：



1.6.5 三元运算符

格式：

(条件表达式)？表达式1：表达式2；

如果条件为true，运算后的结果是表达式1；

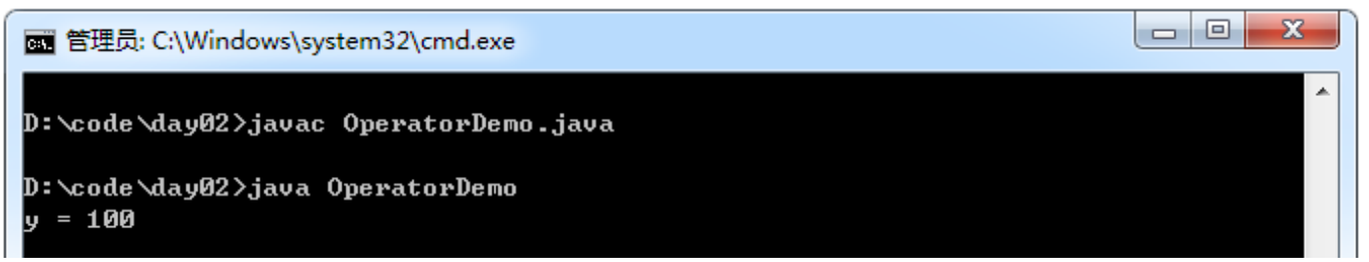
如果条件为false，运算后的结果是表达式2。

示例1：

```
class OperatorDemo
{
    public static void main(String[] args){
        int x = 3,y;
        y = ( x > 1 ) ? 100 : 200;
        System.out.println( "y = " + y );
    }
}
```

复制代码

运行结果：

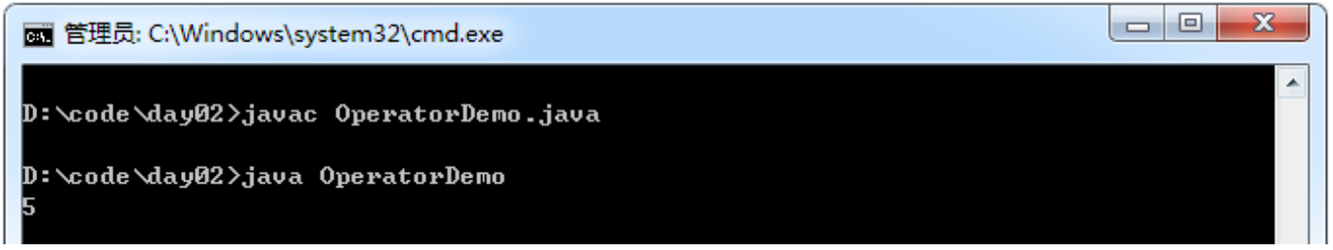


示例2：获取两个整数中的较大的整数。

```
class OperatorDemo
{
    public static void main(String[] args){
        int a = 3,b = 5;
        int max = a > b ? a : b;
        System.out.println( max );
    }
}
```

复制代码

运行结果：

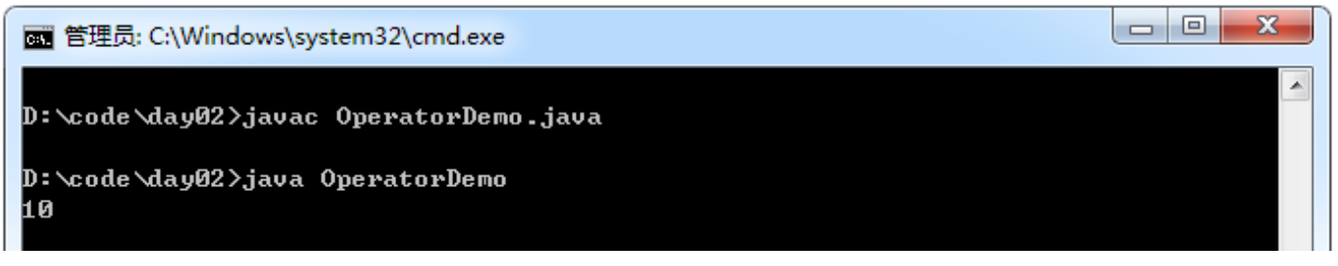


示例3：获取三个整数中的较大的整数。

```
01.   class OperatorDemo
02.   {
03.       public static void main(String[] args){
04.           int o = 4,p = 5,q = 10;
05.           int temp = o > p ? o : p;
06.           int max = temp > q ? temp : q;
07.           System.out.println( max );
08.       }
09.   }
```

复制代码

运行结果：



P.S.

表达式：就是具有一定语法规则的语句。

1.7 程序流程控制

1.7.1 判断结构

if语句

格式一：

if（条件表达式）

{

　　执行语句；

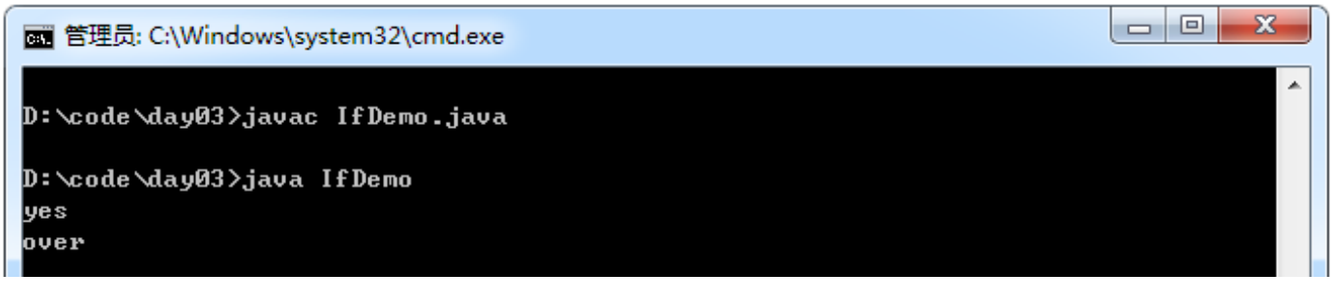
}

示例：

```
class IfDemo
{
    public static void main(String[] args){
        int x = 3;
        if( x > 1 ){
            System.out.println("yes");
        }
        System.out.println("over");
    }
}
```

复制代码

运行结果：



P.S.

1、如果if语句中只有一条语句，那么可以不写大括号。不过初学者一定要写括号，以免出错。

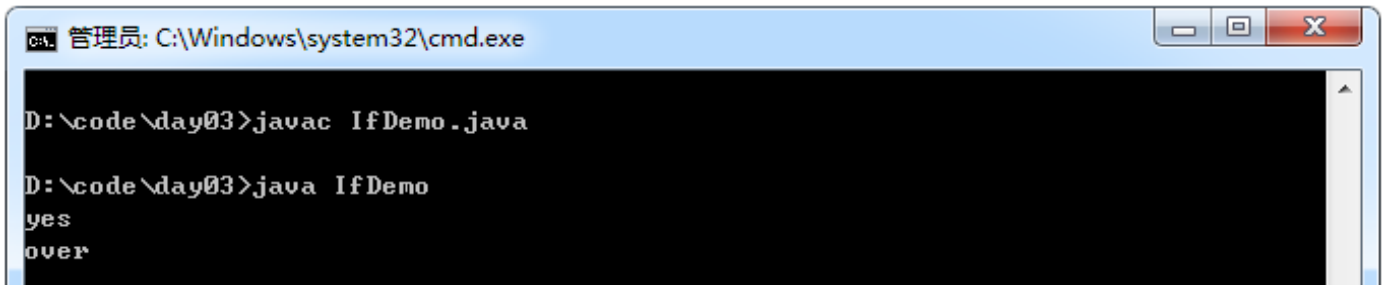
2、如果if语句没写大括号，if就只能控制离它最近的单条语句。

示例：

```
class IfDemo
{
    public static void main(String[] args){
        int x = 3;
        if( x > 1 )
            System.out.println("yes");
            System.out.println("over");
    }
}
```

复制代码

运行结果：



格式二：

if（条件表达式）

{

　　执行语句；

}

else

{

　　执行语句；

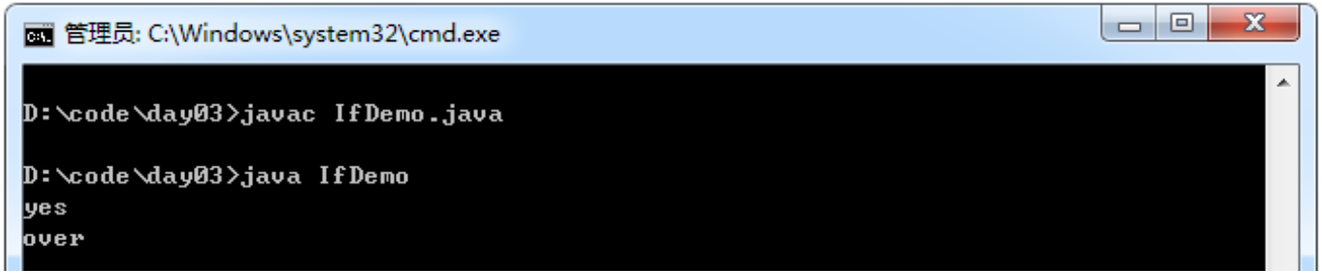
}

示例1：

```
class IfDemo
{
    public static void main(String[] args){
        int x = 3;
        if( x > 1 ){
            System.out.println("yes");
        }else{
             System.out.println("over");
        }
        System.out.println("over");
    }
}
```

复制代码

运行结果：



示例2：

```
class IfDemo
{
    public static void main(String[] args){
        int a = 3,b;
        if( a > 1 )
            b = 100;
        else
            b = 200;
        System.out.println("b = " + b);
    }
}
```

复制代码

运行结果：

P.S.

三元运算符就是if else语句的简写格式。

例如：b = a > 1 ? 100 : 200;就可以实现和上面同样的功能。

简写格式什么时候用？

当if else运算后，有一个具体的结果时，可以简化写成三元运算符。

格式三：

if(条件表达式)

{

    执行语句；

}

else if (条件表达式)

{

执行语句；

}

……

else

{

执行语句；

}

示例：

```
01.   class IfDemo
02.   {
03.       public static void main(String[] args){
04.           int x = 3;
05.           if( x > 4 )
06.               System.out.println("a");
07.           else if( x > 3 )
08.               System.out.println("b");
09.           else if ( x > 2 )
10.               System.out.println("c");
11.           else if(  x > 1 )
12.               System.out.println("d");
13.
14.           System.out.println("over");
15.       }
16.   }
```

复制代码

运行结果：

P.S.

if(表达式)后面切忌加上分号，否则表示无论表达式为true或者false，都不执行任何语句。

示例：

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

复制代码

运行结果：

说明：由于if(false);语句后面加了分号，因此不执行任何操作。{ System.out.println("Hello World"); }为局部代码块。

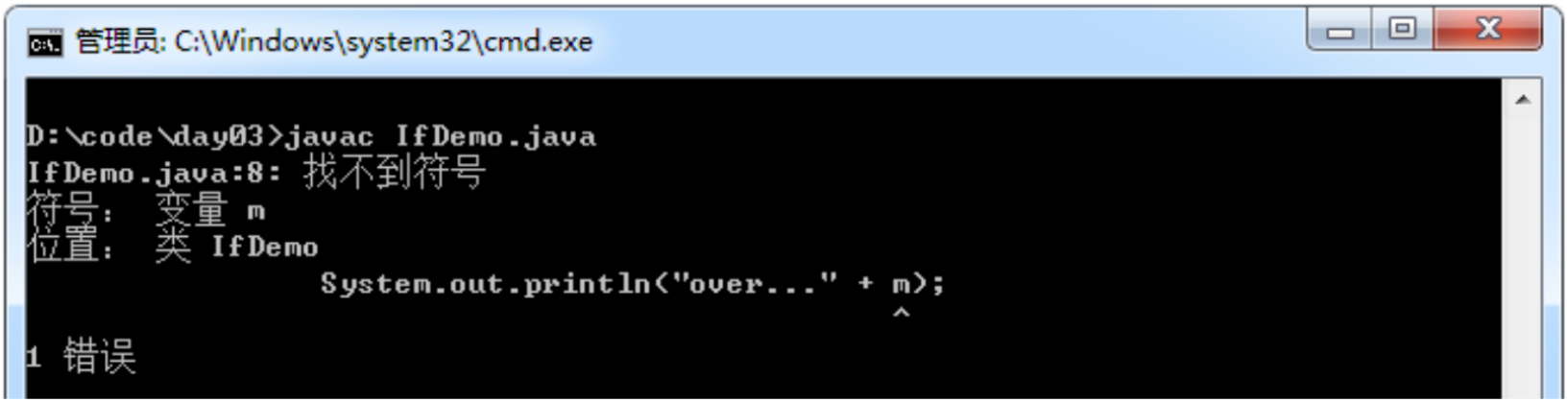
局部代码块：局部代码块可以定义局部变量的生命周期。

示例：

```
class IfDemo
{
    public static void main(String[] args){
    {
        int m = 89;
        System.out.println("Hello World..." + m);
    }
    System.out.println("over..." + m);
    }
}
```

复制代码

运行结果：



说明：变量m是局部变量，生命周期仅在局部代码块括号中，当代码执行到局部代码块右括号外，变量m便消失了。局部代码块外的代码当然也就访问不到变量m了。

if语句特点：

1、每一种格式都是单条语句。

2、第二种格式与三元运算符的区别：三元运算符运算完一定要有值出现。好处是：可以写在其他表达式中。

3、条件表达式无论写成什么样子，只看最终的结构是否是true或者false。

练习1：根据用户定义的数值不同，打印对应的星期。

```
class IfDemo
{
```

```
03.         public static void main(String[] args)
04.         {
05.                 /*
06.                 需求：根据用户指定的具体数据，判断该数据对应的星期。
07.
08.                 思路：
09.                 虽然用户输入无法获取，但是那只是具体数据的一种获取手段而已。
10.                 而我们要做的功能仅仅是对用户指定的数据进行对应星期的打印而已。
11.                 所以具体的数据不确定，可以使用变量来表示。
12.                 我们只对变量进行操作即可。至于变量的值，可以由用户决定。
13.                 因为数据的不确定性，所以要对数据进行判断。
14.                 使用if语句。
15.                 */
16.
17.                 int week = 4;
18.
19.                 if(week==1)
20.                         System.out.println(week+"对应中文是星期一");
21.                 else if(week==2)
22.                         System.out.println(week+"对应中文是星期二");
23.                 else if(week==3)
24.                         System.out.println(week+"对应中文是星期三");
25.                 else if(week==4)
26.                         System.out.println(week+"对应中文是星期四");
27.                 else if(week==5)
28.                         System.out.println(week+"对应中文是星期五");
29.                 else if(week==6)
30.                         System.out.println(week+"对应中文是星期六");
31.                 else if(week==7)
32.                         System.out.println(week+"对应中文是星期日");
33.                 else
34.                         System.out.println(week+"没有对应的星期");
35.         }
36. }
```

复制代码

运行结果：

练习2：根据用于指定月份，打印该月份所属的季节。

```
class IfDemo
{
        public static void main(String[] args)
        {
                /*
                一年有四季。
                春季：3 4 5
                夏季：6 7 8
                秋季：9 10 11
                冬季：12 1 2
                根据用户输入的月份，给出对应的季节。
                */

                int month = 9;

                if(month<1 || month>12)
                        System.out.println(month+"月没有对应的季节");
                else if(month>=3 && month<=5)
                        System.out.println(month+"月是春季");
                else if(month>=6 && month<=8)
                        System.out.println(month+"月是夏季");
                else if(month>=9 && month<=11)
                        System.out.println(month+"月是秋季");
                else
                        System.out.println(month+"月是冬季");
        }
}
```

复制代码

运行结果：

1.7.2 选择结构

switch语句

格式：

switch(表达式)

{

case 取值1:

执行语句；

break；

case 取值2:

执行语句；

break；

……

default:

执行语句；

break；

}

switch语句特点：

1、switch语句选择的类型只有四种：byte，short，int，char。

2、case与default没有顺序。先执行第一个case，没有匹配的case执行default。

3、结束switch语句的两种情况：①遇到break，②执行到switch语句结束。

4、如果匹配的case或者default没有对应的break，那么程序会继续向下执行，运行可以执行的语句，直到遇到break或者switch结尾结束。

5、进入switch语句后，执行顺序是先执行case，然后从上到下，最后再执行default。即使default放在case上面，执行顺序也不变。

示例1：

```
class SwitchDemo
{
    public static void main(String[] args){
        int x= 3;
        switch(x)
        {
            default:
                System.out.println("d" );
                break;
            case 4:
                System.out.println("a" );
                break;
```

```
            case 2:
                System.out.println("b" );
                break;
            case 3:
                System.out.println("c" );
                break;
        }
    }
}
```

复制代码

运行结果：

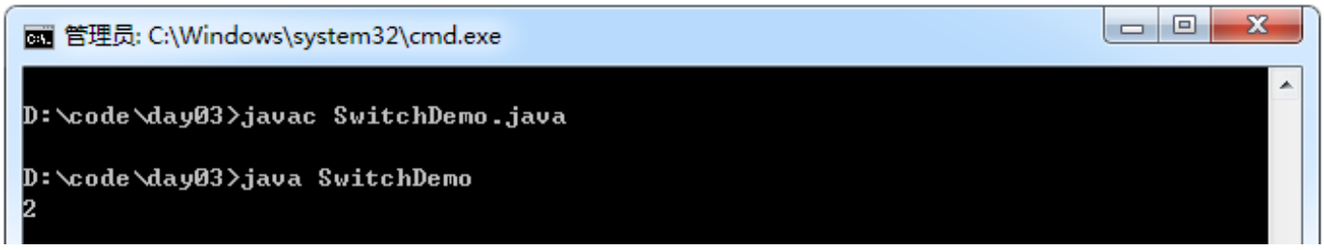
示例2：

```
class SwitchDemo
{
    public static void main(String[] args){
        int a = 4,b = 2;
        char opr = '-' ;

        switch(opr)
        {
            case '+' :
                System.out.println(a+b);
                break;
            case '-' :
                System.out.println(a-b);
                break;
            case '*' :
                System.out.println(a*b);
                break;
            case '/' :
                System.out.println(a/b);
                break;
            default:
                System.out.println("无法运算，符号不支持" );
```

```
                    break;
            }
        }
    }
```

复制代码

运行结果：

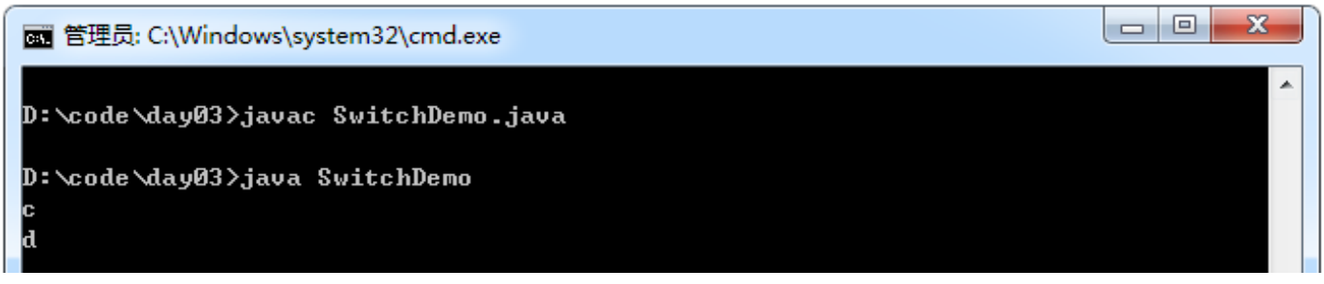


示例3：部分case和default中没有break语句的情况。

```
class SwitchDemo
{
        public static void main(String[] args){
              int x= 3;
              switch (x)
             {
                    case 4:
                          System.out.println("a" );
                          break ;
                    case 2:
                         System.out.println("b" );
                    case 3:
                         System.out.println("c" );
                    default :
                         System.out.println("d" );
             }
        }
}
```

复制代码

运行结果：

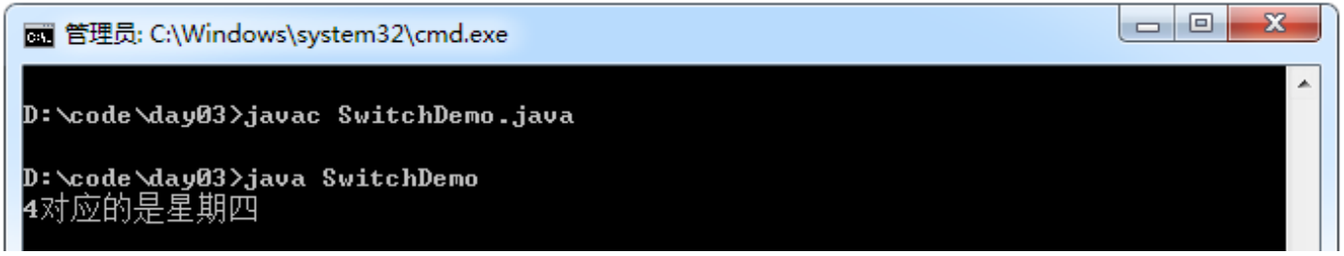


示例4：

```
class SwitchDemo
{
        public static void main(String[] args){
             /*
             用户输入的数据对应的星期。
             */
             int week = 4;
             switch(week)
             {
                   case 1:
                         System.out.println(week + "对应的是星期一" );
                         break;
                   case 2:
                         System.out.println(week + "对应的是星期二" );
                         break;
                   case 3:
                         System.out.println(week + "对应的是星期三" );
                         break;
                   case 4:
                         System.out.println(week + "对应的是星期四" );
                         break;
                   case 5:
                         System.out.println(week + "对应的是星期五" );
                         break;
                   case 6:
                         System.out.println(week + "对应的是星期六" );
                         break;
                   case 7:
                         System.out.println(week + "对应的是星期日" );
                         break;
                   default:
                         System.out.println(week + "没有对应的星期" );
                         break;
             }
        }
}
```

复制代码

运行结果：

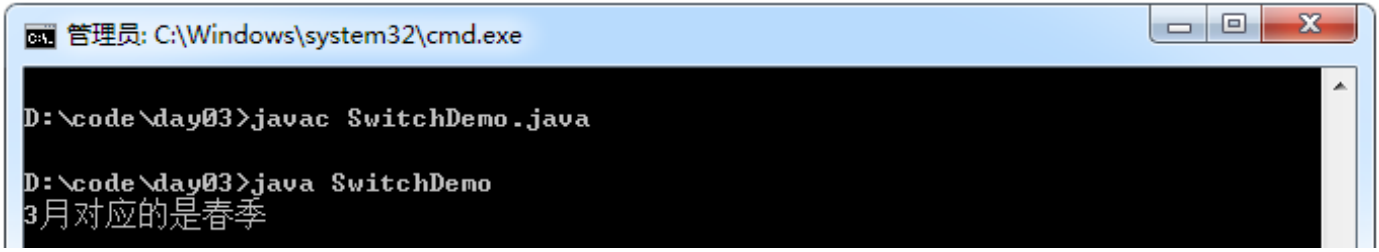


示例5：多个case同一种处理方式的情况。

```
01.  class SwitchDemo
02.  {
03.          public static void main(String[] args){
04.               int month = 3;
05.
06.               switch(month){
07.                    case 3:
08.                    case 4:
09.                    case 5:
10.                         System.out.println(month + "月对应的是春季" );
11.                         break;
12.                    case 6:
13.                    case 7:
14.                    case 8:
15.                         System.out.println(month + "月对应的是夏季" );
16.                         break;
17.                    case 9:
18.                    case 10:
19.                    case 11:
20.                         System.out.println(month + "月对应的是秋季" );
21.                         break;
22.                    case 12:
23.                    case 1:
24.                    case 2:
25.                         System.out.println(month + "月对应的是冬季" );
26.                         break;
27.                    default:
28.                         System.out.println(month + "没有对应的季节" );
29.                         break;
30.               }
31.          }
32.  }
```

复制代码

运行结果：



if和switch语句的应用：

if：

1. 对具体的值进行判断。

2. 对区间判断。

3. 对运算结果是boolean类型的表达式进行判断。

switch：

1. 对具体的值进行判断。

2. 值的个数通常是固定的。

对于几个固定的值判断，建议使用switch语句，因为switch语句会将具体的答案都加载进内存，效率相对高。

1.7.3 循环结构

关键词：while,do while,for。

while语句格式：

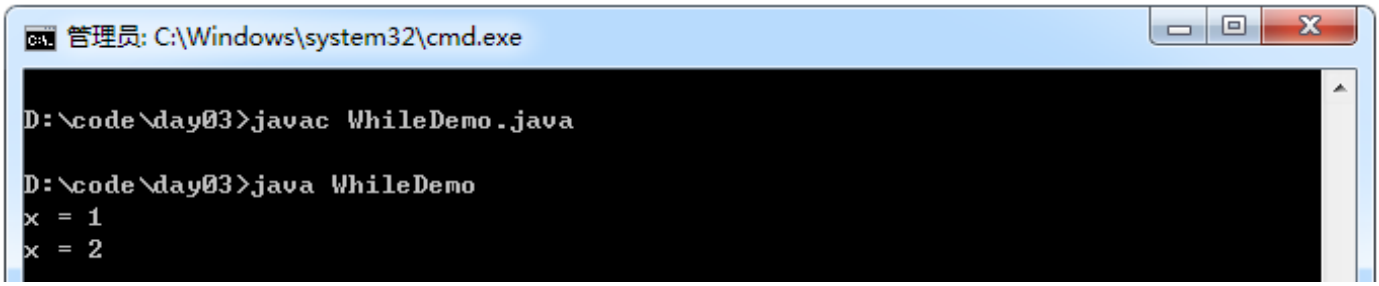
```
while( 条件表达式 )
{
    执行语句；
}
```

示例：

```
class  WhileDemo
{
        public static void main(String[] args)
        {
                int x = 1;
                while(x<3)
                {
                        System.out.println("x="+x);
                        x++;
                }
        }
}
```

复制代码

运行结果：



P.S.

一定要注意不要写while(x < 3);这样的语句，后面的分号就是循环体，代表不执行任何语句，这个循环就成了一个死循环。

do while语句格式：

do

{

    执行语句；

}while(条件表达式);

示例：

```
class DoWhileDemo
{
        public static void main(String[] args){
          int x = 1;
          do{
             System.out.println("x = " + x);
             x++;
          }
          while(x < 3);
       }
}
```

复制代码

运行结果：

while和do while的区别：

do while语句的特点：无论条件是否满足，循环体至少执行一次。

while如果条件不满足，循环体一次都不会执行。

练习1：获取1到10,10个数字的和。

```
class WhileDemo
{
    public static void main(String[] args){
      int x = 1;//记录参与加法的数据。
      int sum = 0;//记录住每一次的和

      while(x <= 10){
         sum += x;
         x++;
      }
      System.out.println("sum = " + sum);
   }
}
```

复制代码

运行结果：

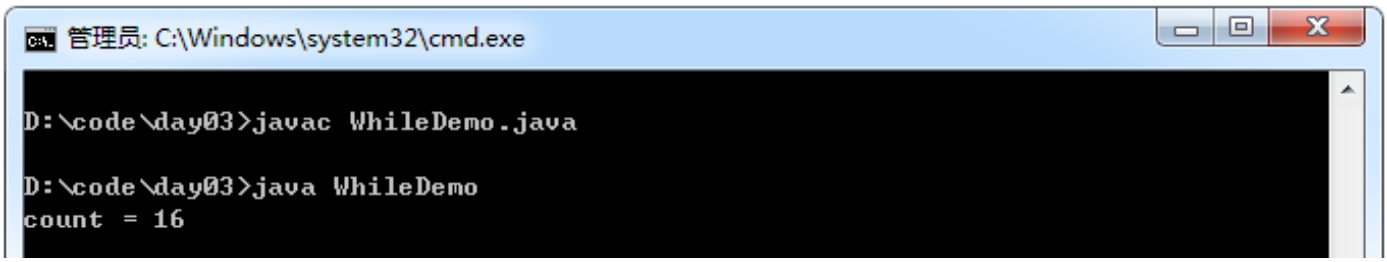
练习2：1~100之间，6的倍数出现的次数。

```
class WhileDemo
{
        public static void main(String[] args){
          int x = 1;
          int count = 0;

          while(x <= 100)
          {
              if(x % 6 == 0){
                   count++;
              }
              x++;
          }
          System.out.println("count = " + count);
        }
```

```
}
```

复制代码

运行结果：



for语句格式：

for(初始化表达式；循环条件表达式；循环后的操作表达式)

{

执行语句；(循环体)

}

for里面的三个表达式运行的顺序，初始化表达式只读一次，判断循环条件，为真就执行循环体，然后再执行循环后的操作表达式，接着继续判断循环条件，重复找个过程，直到条件不满足为止。

示例：

```
class ForDemo
{
        public static void main(String[] args){
          for(int x = 0; x < 3; x++){
               System.out.println("x = " + x);
          }
    }
```

复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe

D:\code\day03>javac ForDemo.java

D:\code\day03>java ForDemo
x = 0
x = 1
x = 2
```

P.S.

for循环的初始化表达式、循环后的操作表达式可以是多个表达式，通过逗号分隔。

例如：

for(int a = 1,b =2; a < 2 & b < 3; a++,b++){

}

练习1：

```
class ForDemo
{
    public static void main(String[]args)
    {
        int x=1;
        for(System.out.println("a");x<3;System.out.println("c"))
        {
            System.out.println("d");
            x++;
        }
    }
}
```

复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe

D:\code\day03>javac ForDemo.java

D:\code\day03>java ForDemo
a
d
c
d
c
```

1、while与for可以互换，区别在于for为了循环而定义的变量在for循环结束就在内存中释放，而while循环使

用的变量在循环结束后还可以继续使用。

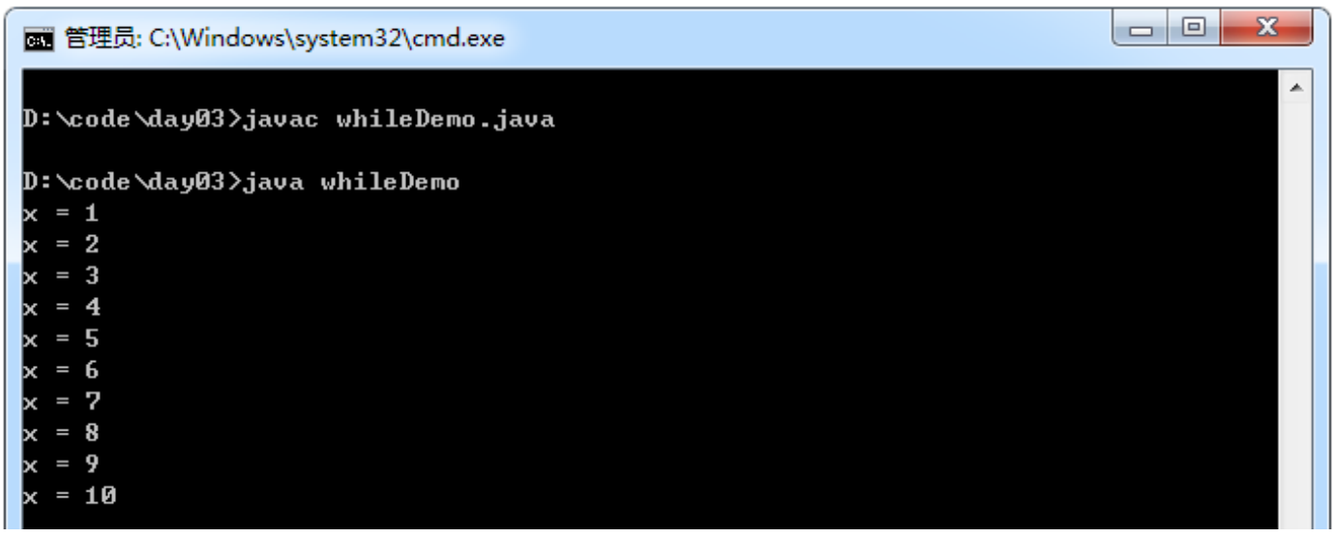
下面通过两个例子对比：

示例1：

```
class whileDemo
{
    public static void main(String[] args){
        //打印1~10十个数字
        int x = 1;
        while(x <= 10)
        {
            System.out.println("x = " + x++);
        }
    }
}
```

复制代码

运行结果：

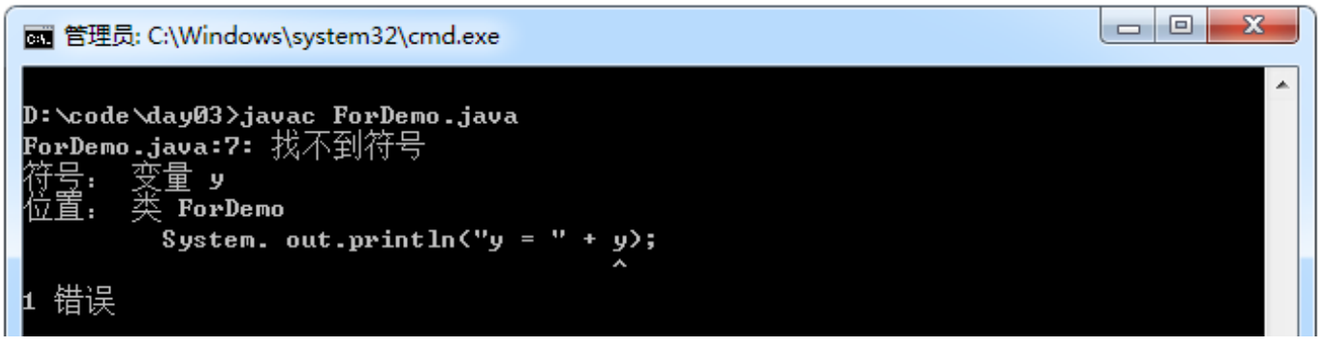


示例2：

```
class ForDemo
{
    public static void main(String[] args){
        for(int y = 1; y < 10; y++){
            System.out.println("y = " + y);
        }
        System.out.println("y = " + y);
    }
}
```

复制代码

运行结果：



2、最简单无限循环格式：while(true)，for(;;),无限循环存在的原因是并不知道循环多少次，而是根据某些条件，来控制循环。

3、在使用循环时候，一定要明确哪些语句需要参与循环，哪些不需要。循环通常情况下，需要定义条件，需要控制次数。

~END~

~爱上海，爱黑马~