

day03-选择语句和循环语句

今日学习内容：

- 顺序结构
- if语句
- if-else语句
- if-else if-else语句
- switch语句
- while循环语句
- do-while循环语句
- for语句
- break、continue、return语句
- 嵌套循环

今日学习目标：

- 了解什么是顺序结构
- 必须掌握if、if-else、if-else if-else的语法和使用
- 必须掌握switch语句的语法和使用
- 必须掌握while、do-while、for的语法和使用
- 掌握什么是嵌套循环和如何使用
- 掌握控制循环语句break、continue、return三者的区别和用法

1、运算符

1.1、逻辑运算符（掌握）

逻辑运算符用于连接两个boolean表达式，结果也是boolean类型的。

语法格式为：

```
boolean result = boolean表达式A 逻辑运算符 boolean表达式B;
```

运算规则如下：

运算符	运算规则	范例	结果
&	与	false&true	false
	或	false true	true
^	异或	true^false	true
!	非	!true	false
&&	短路与	false&&true	false
	短路或	false true	true

逻辑运算符运算规律 (记住):

& 与运算, 可以理解为 "并, 并且"

true & true => true

true & false => false

false & true => false

false & false => false

总结 : & 运算, 只要两边的表达式有一个为false, 结果就为false

| 或运算, 可以理解为 "或, 或者"

true | true => true

true | false => true

false | true => true

false | false => false

总结 : | 运算, 只要两边的表达式有一个为true, 结果就为true

! 非运算, 可以理解为 取反

!true = false

!false = true

1.2、基本使用 (掌握)

```
public class Test01Logic {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int c = 30;
        //与操作
        System.out.println((a > b) & (a > c)); // false & false
        System.out.println((a > b) & (a < c)); // false & true
        System.out.println((a < b) & (a > c)); // true & false
        System.out.println((a < b) & (a < c)); // true & true

        //或操作
        System.out.println((a > b) | (a > c)); // false | false
        System.out.println((a > b) | (a < c)); // false | true
        System.out.println((a < b) | (a > c)); // true | false
    }
}
```

```

        System.out.println((a < b) | (a < c)); // true | true

        //相反操作
        System.out.println((a > b)); // false
        System.out.println(!(a > b)); // !false
        System.out.println(!(a > b)); // !!false
    }
}

```

1.3、&和&&的区别 (掌握)

&：&左边表达式无论真假，&右边表达式都进行运算；

&&：如果&&左边表达式为真，&&右边表达式参与运算，否则&&右边表达式不参与运算，故称短路与。

| 和 || 的区别同理，||，左边为真，右边不参与运算。

&& 短路与 可以理解为 "并，并且"

```

true && true => true
true && false => false
false && true => false
false && false => false

```

&& 运算，只要两边的表达式有一个为false，结果就为false，如果第一个表达式为false，后续表达式不再运算；

```

public class Test02Logic {
    public static void main(String[] args) {
        // 逻辑运算符
        // 短路运算 && ||
        int m = 10;
        int n = 20;
        // boolean r1 = false && false;
        boolean r1 = (m > n) && (++m > 1);
        System.out.println("r1 = " + r1);
        System.out.println("m = " + m);

        boolean r2 = (m > n) | (m < 1);
        System.out.println("r2 = " + r2);
    }
}

```

上述代码，一行一行的测试，测试完，注释该行代码。

2、运算优先级 (记结论)

表达式的运算都是有优先级的，基本上和数学中的优先级类似，

```
public class Test01Priority {
    public static void main(String[] args) {
        int m = 10;
        int n = 20;
        // 给定一个表达式,如果让你计算结果,你首先想到的是?
        m++ > n * 2 && (m + n) * 2 > 100;
    }
}
```

优先级	运算符	类	结合性
1	()	括号运算符	由左至右
1	[]	方括号运算符	由左至右
2	!, + (正号)、- (负号)	一元运算符	由右至左
2	~	位逻辑运算符	由右至左
2	++, --	递增与递减运算符	由右至左
3	*, /, %	算术运算符	由左至右
4	+, -	算术运算符	由左至右
5	<<, >>	位左移、右移运算符	由左至右
6	>, >=, <, <=	关系运算符	由左至右
7	==, !=	关系运算符	由左至右
8	& (位运算符 AND)	位逻辑运算符	由左至右
9	^ (位运算符 XOR)	位逻辑运算符	由左至右
10	(位运算符 OR)	位逻辑运算符	由左至右
11	&&	逻辑运算符	由左至右
12		逻辑运算符	由左至右
13	?:	条件运算符	由右至左
14	=	赋值运算符	由右至左

注意：赋值符号最后运算的，并且是从右向左运算的。

一句话：() 的优先级最高，赋值运算符优先级最低；赋值运算符的运算方向从右向左。

3、选择结构

3.1 顺序结构 (重点)

如果代码里没有流程控制，程序是按照书写的格式从上而下一行一行执行的，一条语句执行完之后继续执行下一条语句，中间没有判断和跳转，直到程序的结束。



```
public class SequenceDemo{
    public static void main(String[] args) {
        System.out.println("A");
        System.out.println("B");
        System.out.println("C");
        System.out.println("D");
    }
}
```

无论代码怎么运行，程序的输出顺序总是ABCD。

但是，我们的程序应该像一个人的人生轨迹一样，会面临很多分岔路的选择，一旦下定决心走某一条路，就不能再回头。



3.2 选择结构（重点）

选择结构也被称为**分支结构**。代码根据**逻辑判断**，存在多个不同的结果，此时就会产生不同的选择，不同的选择执行不同的代码。Java中存在两种选择结构语句：

- **if-elseif-else 语句**
- **switch 语句**

3.2.1 if（重点）

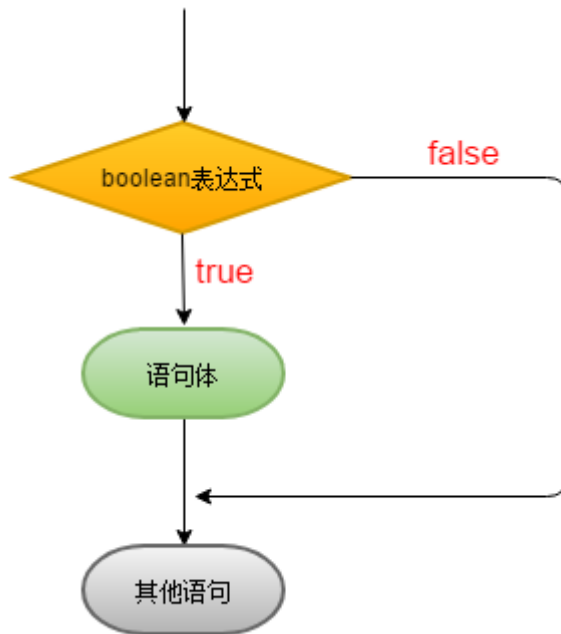
语义：如果。。。那么。。。

```
if(boolean表达式){
    语句体;
}
```

if后面的{}表示一个整体—代码块，称之为语句体，当boolean表达式为true，才执行这里的代码块。

注意：if(boolean表达式)后面是没有分号的。

```
if(boolean表达式); //错误写法，不能有分号
{
    语句体;
}
```



```
public class IfDemo {
    public static void main(String[] args) {
        System.out.println("begin...");
        // 定义一个变量
        int a = 10;
        // 如果a大于5，执行语句体的打印
        if (a > 5) {
            System.out.println("a大于5");
        }
        System.out.println("and...");

        // 如果a大于20，执行语句体的打印
        if (a > 20) {
            System.out.println("a大于20");
        }
        System.out.println("ending...");
    }
}
```

输出结果：

```
begin...
a等于5
and...
ending...
```

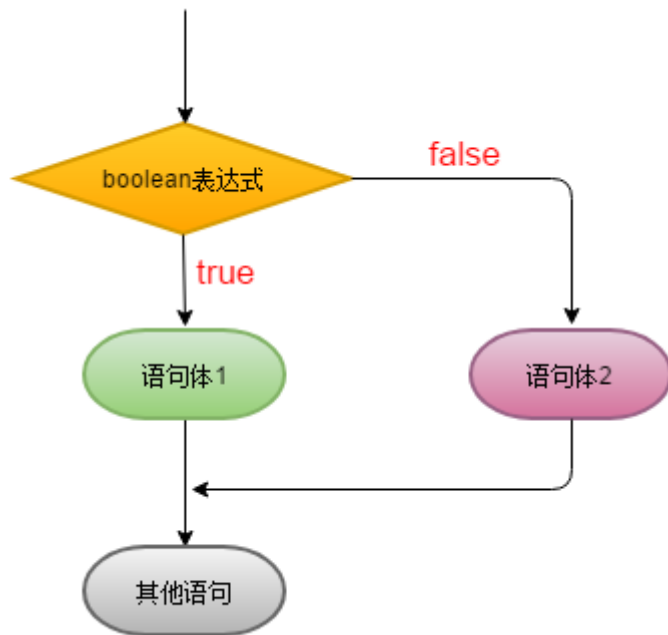
3.2.2 if-else (重点)

语义：如果。。。那么。。。否则。。。

二选一

```
if (boolean表达式) {  
    语句体1;  
} else {  
    语句体2;  
}
```

如果boolean表达式结果为true，就执行语句体1，否则执行语句体2。



```
public class IfElseDemo {  
    public static void main(String[] args) {  
        System.out.println("begin...");  
        // 定义一个变量  
        int a = 10;  
        // 如果变量a的值能被2整除，那么执行语句体的打印  
        if (a % 2 == 0) {  
            System.out.println("a是偶数");  
        } else { // 否则执行这里的语句体  
            System.out.println("a是奇数");  
        }  
  
        System.out.println("and...");  
  
        int b = 11;  
        if (b % 2 == 0) {  
            System.out.println("b是偶数");  
        } else {  
            System.out.println("b是奇数");  
        }  
        System.out.println("ending...");  
    }  
}
```

输出结果：

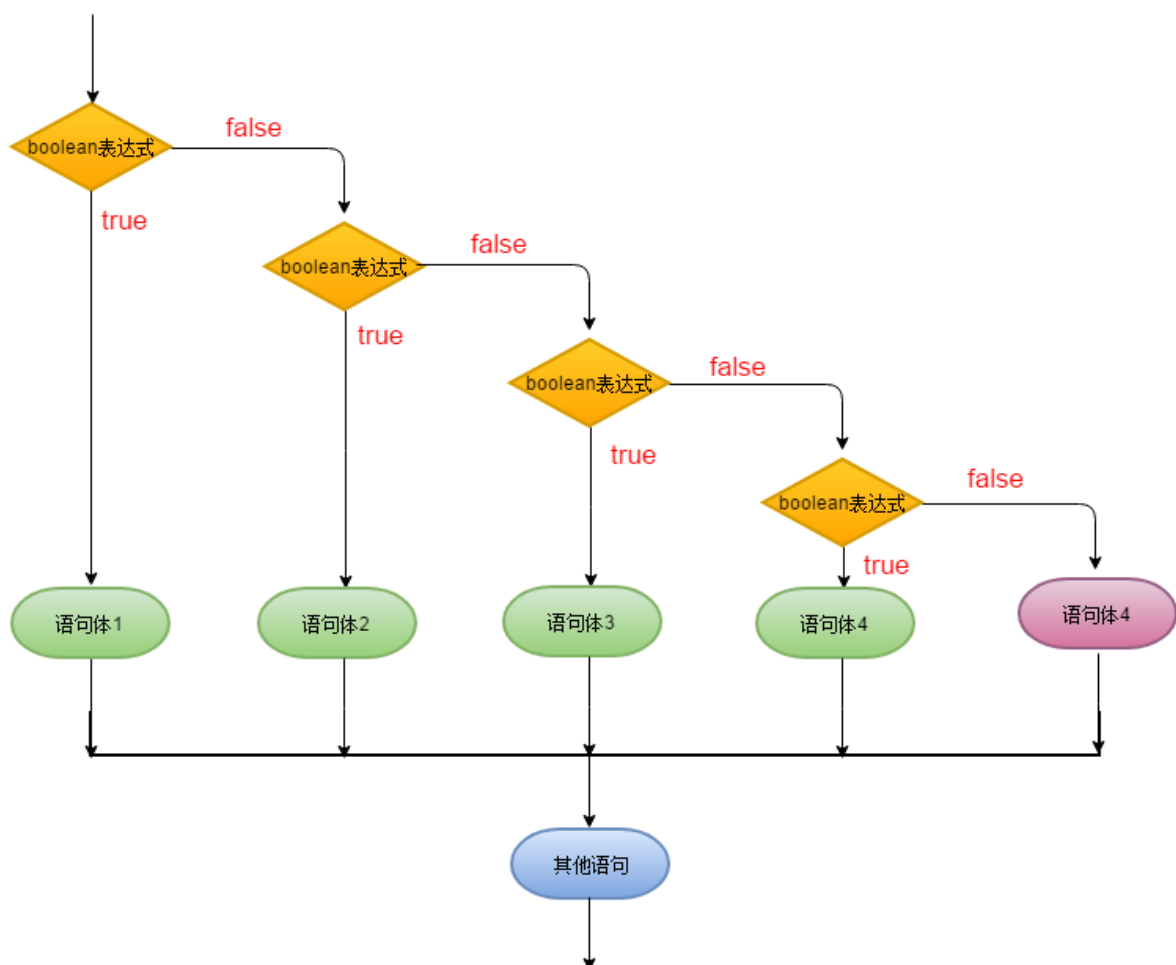
```
begin...  
a是偶数  
and...  
b是奇数  
ending...
```

3.2.3 if else if else (重点)

语义：如果。。。那么。。。如果。。。那么。。。否则。。。

多选一

```
if (boolean表达式1) {  
    语句体1  
}  
else if (boolean表达式2) {  
    语句体2  
}  
//...可以有多个else if  
else {  
    上述条件都为false, 执行该语句体  
}
```



```
public class IfElseIfElseDemo1 {
```



```

public static void main(String[] args) {
    System.out.println("begin...");
    int a = 10;
    int b = 20;
    if (a > b) {
        System.out.println("a > b");
    } else if (a < b) {
        System.out.println("a < b");
    } else {
        System.out.println("a == b");
    }
    System.out.println("ending...");
}
}

```

判断QQ在线的等级：

级别	图标	天数
0级		0天
1级	☆	5天
2级	☆☆	12天
3级	☆☆☆	21天
4级	☾	32天

```

/**
 * 需求:根据天数输出qq等级
 * [0,5) 无等级
 * [5,12) ☆
 * [12,21) ☆☆
 * [21,32) ☆☆☆
 * [32,~) ☾
 */

import java.util.Scanner;
public class IfElseIfElseDemo2 {
    public static void main(String[] args) {
        System.out.println("begin...");
        //从控制台中读取输出的数据
        Scanner s = new Scanner(System.in);
        System.out.println("请输入QQ在线天数: ");

        //获取控制台输入的天数（不要输入非数字）
        int days = s.nextInt();
        if (days >= 0 && days < 5) {
            System.out.println("没有等级");
        } else if (days >= 5 && days < 12) {
            System.out.println("☆");
        } else if (days >= 12 && days < 21) {
            System.out.println("☆☆");
        } else if (days >= 21 && days < 32) {
            System.out.println("☆☆☆");
        } else {
            System.out.println("终于有月亮了");
        }
    }
}

```

```

        System.out.println("ending...");
    }
}

```

从上面的代码中，你能发现什么？（自己验证总结）

```

import java.util.Scanner;

public class Demo08IFElseIFElse3 {
    public static void main(String[] args) {
        // if 多分支
        // 需求:根据天数输出qq等级
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入qq在线天数:");
        int days = sc.nextInt();

        if(days >= 0 && days < 5){
            System.out.println("无等级");
        }else if( days < 12){
            System.out.println("☆");
        }else if (days < 21){
            System.out.println("☆☆");
        }else if (days < 32){
            System.out.println("☆☆☆");
        }else{
            System.out.println("C");
        }
    }
}

```

需求：根据输入的一个数字，判断是星期几？

```

public static void main(String[] args) {
    System.out.println("begin...");
    Scanner s = new Scanner(System.in);
    System.out.println("请输入一个数字: ");
    //获取控制台输入的数字（不要输入非数字）
    int weekday = s.nextInt();
    if (weekday == 1) {
        System.out.println("周一");
    } else if (weekday == 2) {
        System.out.println("周二");
    } else if (weekday == 3) {
        System.out.println("周三");
    } else if (weekday == 4) {
        System.out.println("周四");
    } else if (weekday == 5) {
        System.out.println("周五");
    } else if (weekday == 6) {
        System.out.println("周六");
    } else if (weekday == 7) {
        System.out.println("周日");
    } else {
        System.out.println("错误数据");
    }
}

```

```

    }
    System.out.println("ending...");
}

```

3.3.2 switch语句（掌握）

上面练习使用if-elseif-else完全没问题，对于这种判断条件是否等于某一个数值的，我们有另一种更简单的分支语句——switch语句，其格式如下：

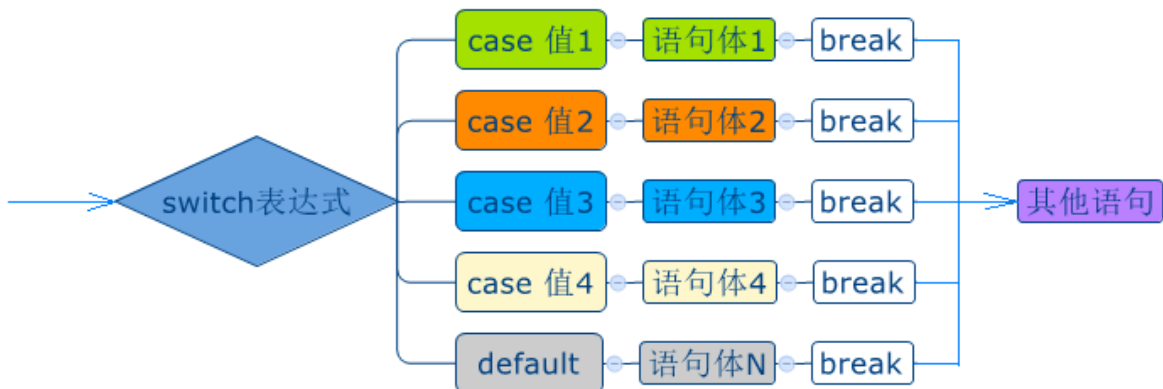
```

switch(整型表达式){
    case A值:
        语句体1; break;
    case B值:
        语句体2; break;
    case C值:
        语句体3; break;
    ...多个case语句
    [default:
        以上值都不满足时，执行此语句;
        break; // 可以省略
    ]
}

```

使用注意：

- 整型表达式的类型：byte,short,char,int
- Java5开始支持枚举（后续讲解）
- Java7开始支持String
- case之后的表达式结果必须是常量
- case表示执行代码的入口，一旦进入某个入口后，代码会忽略掉后面其他case，代码会一直往下执行，直到遇到break或return。（**switch的穿透效果**）



```

public class SwitchDemo{
    public static void main(String[] args) {
        System.out.println("begin...");
        Scanner s = new Scanner(System.in);
        System.out.println("请输入一个数字: ");
        //获取控制台输入的数字（不要输入非数字）
        int weekday = s.nextInt();
        switch (weekday) {
            case 1:    System.out.println("周一");        break;
            case 2:    System.out.println("周二");        break;
            case 3:    System.out.println("周三");        break;
            case 4:    System.out.println("周四");        break;

```

```

        case 5:    System.out.println("周五");        break;
        case 6:    System.out.println("周六");        break;
        case 7:    System.out.println("周日");        break;
        default:   System.out.println("错误数据");    break;
    }
    System.out.println("ending...");
}
}

```

效果等价于if-elseif-elseif-else

15.30

switch省略break的变种语法

```

public class Demo09Switch4 {
    public static void main(String[] args) {
        // 需求: 给定一个小写字母, 是元音, 还是辅音 ?
        // 判断标准:(a e i o u)是元音, 其他字母是辅音
        char c = 'a';
        switch (c){
            case 'a' :
            case 'e' :
            case 'i' :
            case 'o' :
            case 'u' :{
                System.out.println("元音");break;
            }
            default:{
                System.out.println("辅音");break;
            }
        }
    }
}

```

使用场景: 所有的选择语句都可以使用if语句, switch语句只适合做等值判断。

4、循环结构

在java中, 如果要做重复性的操作时, 一定想到循环结构。

循环条件
循环体(重复性操作)
 在满足循环条件的情况下, 循环体一直执行。

在满足循环条件的情况下, 重复执行某一段代码, 这段被重复执行的代码被称为循环体语句。

- while语句
- do while语句
- for语句

注: 三种循环结构可以完成相同的功能, 仅仅只是语法上有差异。

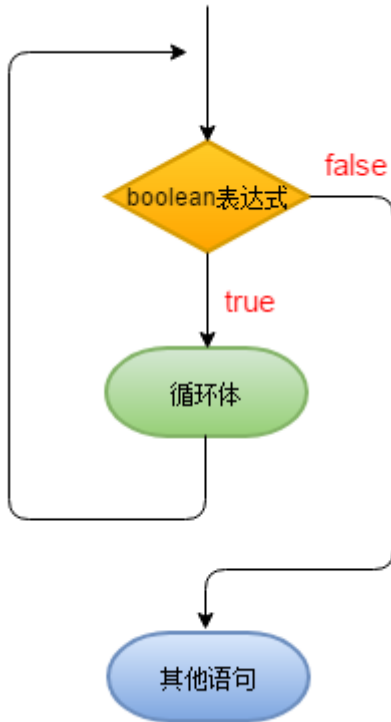
案例1: 叫500声帅哥, 打印500次帅哥

案例2: 从1数到100, 打印从1~100

案例3: 计算100以内的正整数和

4.1 while (重点)

```
while(boolean表达式) {  
    循环体;  
}
```



特点: 先判断boolean表达式:

- 若为false, 跳过循环体,
- 若为true, 执行循环体, 执行完, 再重新判断boolean表达式。

需求1: 打印10行帅哥

```
public class whileDemo1 {  
    public static void main(String[] args) {  
        System.out.println("begin...");  
        int count = 1;  
        while(count <= 10) {  
            System.out.println("帅哥");  
            count++;  
        }  
        System.out.println("ending...");  
    }  
}
```

需求2: 从1打印到100

```

public class WhileDemo2 {
    public static void main(String[] args) {
        System.out.println("begin...");
        int count = 1;
        while (count <= 100) {
            System.out.println(count);
            count++;
        }
        System.out.println("ending...");
    }
}

```

需求3: 1 + 2 + 3 + 4 + 5 + ... + 10 的和

```

public class WhileDemo3 {
    public static void main(String[] args) {
        System.out.println("begin...");

        int total = 0; // 最终之和, 初始为0
        int count = 1;
        while (count <= 10) {
            total = total + count;
            count++;
        }
        System.out.println(total);

        System.out.println("ending...");
    }
}

```

综合案例

```

public class Test11While5 {
    public static void main(String[] args) {
        // 需求: 求1-10之间所有偶数的和
        /**
         * 分析:
         * 1 => 问1你是不是偶数, 你就是求和的数
         * 2 => 问2你是不是偶数, 你就是求和的数 sum+=2
         * 3 => 问3你是不是偶数
         */

        int sum = 0;
        int i = 1;
        while( i <= 10 ){
            if( i % 2 == 0 ){
                sum += i;
            }
            i++;
        }
        System.out.println("sum = " + sum);

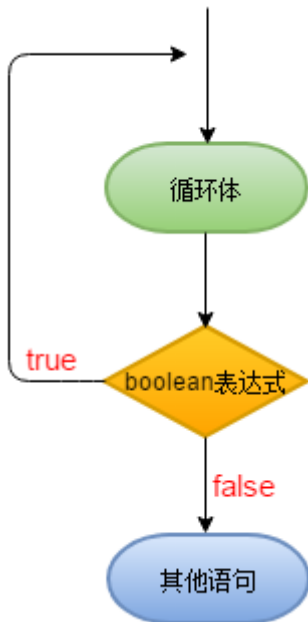
        /**
         * while和if可以嵌套使用
         */
    }
}

```

```
}
```

4.2 do while (了解)

```
do {  
    循环体;  
} while(boolean表达式);
```



特点：先执行一次循环体，再判断表达式：

若为true，就执行循环体，再重新判断boolean表达式

若为false，跳过循环体。

while和do while的区别

do while是先执行后判断，即使判断条件为false，**该循环至少会执行一次**。

```
public class DowhileDemo1 {  
    public static void main(String[] args) {  
        System.out.println("begin...");  
        int a = 5;  
        int b = 100;  
        while (a > b) {  
            System.out.println("5大于100");  
        }  
        System.out.println("and...");  
  
        do {  
            System.out.println("5大于100");  
        } while (a > b);  
        System.out.println("ending...");  
    }  
}
```

需求1：打印10行帅哥

```

public class DowhileDemo2 {
    public static void main(String[] args) {
        System.out.println("begin...");
        int count = 0;
        do {
            System.out.println("帅哥");
            count++;
        } while (count < 10);
        System.out.println("ending...");
    }
}

```

需求2: 从1打印到100

```

public class DowhileDemo3 {
    public static void main(String[] args) {
        System.out.println("begin...");
        int count = 1;
        do {
            System.out.println(count);
            count++;
        } while (count <= 100);
        System.out.println("ending...");
    }
}

```

需求3: 计算100以内正整数之和

```

public class DowhileDemo4 {
    public static void main(String[] args) {
        System.out.println("begin...");
        int total = 0; //最终之和, 初始为0
        int count = 1;
        do {
            total = total + count;
            count++;
        } while (count <= 100);
        System.out.println(total);
        System.out.println("ending...");
    }
}

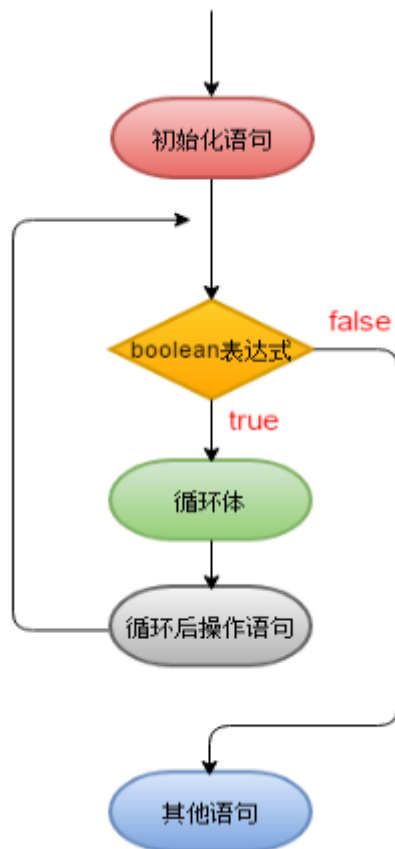
```

4.3 for (重点)

```

for(①初始化语句; ②boolean表达式; ③循环后操作语句){
    ④循环体;
}

```

特点:

- 初始化语句：只在循环开始时执行一次，一般是定义一个变量，并赋值，用于控制循环次数。
- boolean表达式：表达式为false时，循环终止，为true,才执行循环体。
- 循环后操作语句：循环体执行后会调用该语句，**一般是变量的递增或递减操作。**

```
for(①初始化语句;②boolean表达式; ④循环后操作语句){  
    ③循环体;  
}
```

执行顺序:

①、初始化语句->②、boolean表达式:

- 若为false: 跳过循环体
- 若为true: ③、执行循环体

④、循环后操作语句->②、boolean表达式:

需求: 计算100以内正整数之和

```

public class ForDemo {
    public static void main(String[] args) {
        System.out.println("begin...");
        int total = 0; // 最终之和, 初始为0
        for (int count = 1; count < 101; count++) {
            total = total + count;
        }
        System.out.println(total);
        System.out.println("ending...");
    }
}

```

死循环：表示循环的boolean表达式一直为true，重复执行循环体。

while循环死循环

```

while (true) {
    //TODO
}

```

do while循环死循环

```

do {
    //TODO
} while (true);

```

for循环死循环

```

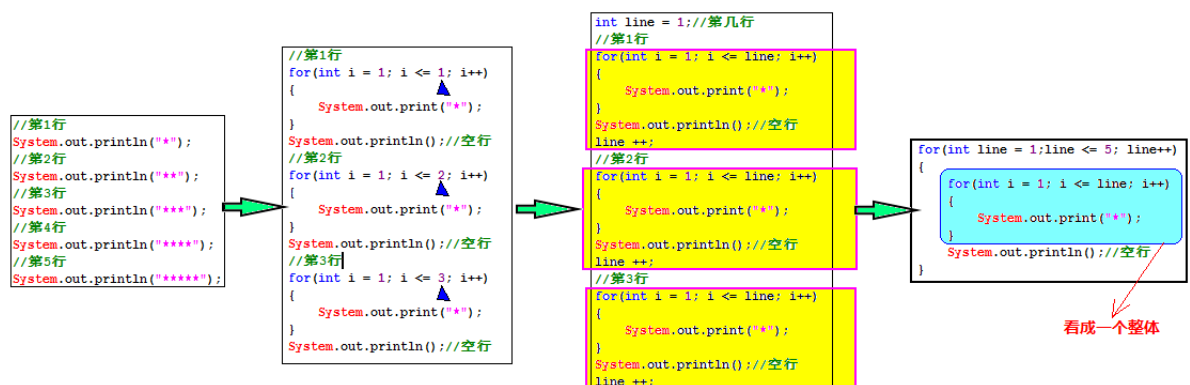
for (;;) {
    //TODO
}

```

小结：三种循环，先掌握任意一种就可以了，都可以完成相对的功能，一般循环变量名称：i, j, k, m, n, p, q。

4.4 嵌套循环（掌握）

循环解决的是：某一个操作需要重复执行，如果一个重复的操作需要做N次，此时得使用嵌套循环。



注：可以把内层循环看成是一个整体。

打印直角三角形

```
public class LoopInLoopDemo{
    public static void main(String[] args) {
        for (int line = 1; line <= 5; line++) {
            for (int i = 1; i <= line; i++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

输出结果：

```
*
**
***
****
*****
```

打印九九乘法表

```
public class Table99Demo{
    public static void main(String[] args) {
        for (int row=1; row<=9; row++){
            for(int col=1; col<=row; col++){
                System.out.print(col + "x" + row + "=" + (row*col) + " ");
            }
            System.out.println();
        }
    }
}
```

输出结果为

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

4.5 控制循环（重点）

4.5.1 break（重点）

break表示结束当前所在的循环。

需求：从1输出到10，当迭代变量为7，就停止循环

```

public class App {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if(i == 7){
                break;//结束当前循环
            }
            System.out.println("第" + i);
        }
    }
}

```

输出结果：

```

第1
第2
第3
第5
第6

```

注意：break之后的语句执行不了，所以不能编写。

4.5.2 continue (重点)

continue表示跳过当前这一次循环，直接进入下一次循环操作。

需求：从1输出到10，不要输出4。

```

public class ContinueDemo {
    public static void main(String[] args) {
        for (int i = 1; i <=10; i++) {
            if(i == 4) {
                continue;
            }
            System.out.println("第" + i);
        }
    }
}

```

输出结果：

```

第1
第2
第3
第5
第6
第7
第8
第9
第10

```

