

# Java运算符&分支流程控制

---

## 一 内容回顾（列举前一天重点难点内容）

---

### 1.1 教学重点:

1. java的执行原理
2. 配置环境变量
3. JVM, JDK, jRE关系
4. 标识符的命名规范
5. 常用数据类型基础
6. 数据类型转换
7. 运算符的使用

### 1.2 教学难点:

1. 怎么让自己拥有编程思维?
2. 对于一个0基础的小伙伴,想拥有编程思维要做到:第一:学习java的基本语法;第二:多写代码
3. 2. 原码反码补码的理解
4. 不做过多要求,有能力的小伙伴可以学习一下.

## 二 教学目标

---

- 1 1.掌握逻辑运算符的使用
- 2 2.了解位运算符的使用
- 3 3.掌握三目运算符的使用
- 4 4.掌握常见多运算符表达式的运算
- 5 5.掌握分支的使用场景
- 6 6.掌握if分支语句的使用
- 7 7.掌握switch循环语句的使用

## 三 教学导读

---

### 3.1. 运算符续

- 1 今天会接着昨天的运算符继续讲下半部分.今天要学的逻辑运算符,三目运算符是运算符教学的重点,位运算符属于了解内容.学习完单独的运算符,接下来我们要学习优先级和结合性来解决多运算符语句的运算.

### 3.2. 流程控制

- 1 1966年, 计算机科学家 Bohm 和 Jacopini 证明了这样的事实: 任何简单或复杂的算法都可以由顺序结构、选择结构和循环结构这三种基本结构组合而成。所以, 这三种结构就被称为程序设计的三种基本结构。也是 结构化程序设计 必须采用的结构。
- 2 荷兰学者Dijkstra1968年提出了“结构化程序设计”的思想, 它规定了一套方法, 使程序具有合理的结构, 以保证和验证程序的正确性, 这种方法要求程序设计者不能随心所欲地编写程序, 而要按照一定的结构形式来设计和编写程序, 它的一个重要目的是使程序具有良好的结构, 使程序易于设计, 易于理解, 易于调试修改, 以提高设计和维护程序工作的效率。
- 3 结构化程序规定了以下三种基本结构作为程序的基本单元: 以上三种基本结构可以派生出其它形式的结构. 由这三种基本结构所构成的算法可以处理任何复杂的问题. 所谓结构化程序就是由这三种基本结构所组成的程序. 可以看到, 三种基本结构都具有以下特点: ①有一个入口. ②有一个出口. ③结构中每一部分都应当有被执行到的机会, 也就是说, 每一部分都应当有一条从入口到出口的路径通过它 (至少通过一次). ④没有死循环 (无终止的循环).
- 4 下面我们就展开分支语句的学习

### 3.3. 分支结构的概念

- 1 对于要先做判断再选择的问题我们要使用分支结构。分支结构的执行是依据一定的条件选择执行路径, 而不是严格按照语句出现的物理顺序。分支结构的程序设计方法的关键在于构造合适的分支条件和分析程序流程, 根据不同的程序流程选择适当的分支语句。分支结构适合于带有逻辑或关系比较等条件判断的计算, 设计这类程序时往往都要先绘制其程序流程图, 然后根据程序流程写出源程序, 这样做把程序设计分析与语言分开, 使得问题简单化, 易于理解。

## 四 教学内容

# 4.1. 运算符(续)(会)

课程重点:

- 逻辑运算符的使用
- 三目运算符的使用
- 运算符优先级和结合性当使用

课程难点:

- 位运算符的使用

## 4.1.1. 逻辑运算符

### 4.1.1.1. 运算符分类

运算符	运算规则	范例	结果
&	理解为并且,逻辑与， 两真即为真， 任意一个为假， 结果即为假。	false&>true	False
	理解为或者,逻辑或， 两假即为假， 任意一个为真， 结果即为真。	false true	True
^	逻辑异或， 相同为假， 不同为真。	true^flase	True
!	逻辑非， 非真即假， 非假既真。	!true	Flase
&&	短路与,如果前面的结果可以决定整体的运算结果, 后面的表达式不参与运算	false&&>true	False
	短路或,如果前面的结果可以决定整体的运算结果, 后面的表达式不参与运算	false  true	True

### 4.1.1.2. 示例代码

```
1  /**
2   * @Author 千锋大数据教学团队
3   * @Company 千锋好程序员大数据
4   * @Description 逻辑运算符
5   */
6  public class JavaSyntax {
7      public static void main(String[] args) {
8          //特点:1.运算符的两边的元素一定是true/false 2.结果一定是
9          true/false
10         //逻辑与 & 逻辑或 | 逻辑异或 ^ 逻辑非
11         !
12         //&:一假则假,全真才真
13         // |:一真则真,全假才假
14         //!:真则假,假则真
15         boolean flag2 = false&false;// & 必须是左右两边
16         都是 true 结果才是true
17         boolean a = false | false;// | 只要有一个true
18         结果就是true 两边都是false 结果才为 false
19
20         System.out.println("flag2 = "+ flag2);
21         System.out.println("a = "+ a);
22
23         // ! 逻辑非
24         boolean e = !true;
25         System.out.println("e = "+ e);
26
27         // ^ 异或 相同则为false 不同为true
28         boolean f = true ^ false;
29         System.out.println("f = "+ f);
```

```

27
28     int a = 4;
29     int c = 5;
30     boolean b4 = ++a>6 & ++c>3;
31     System.out.println(a);
32     System.out.println(c);
33
34
35     //短路与 && 和 短路或 ||
36     //注意:短路与,短路或的最终结果与逻辑与,逻辑或的一样.
37     //举例:
38     System.out.println(2 >= 3 && 3 <= 4);
39
40     /*
41     * 短路与运算规则:当一个式子中有多个连续的&&,我们只需要找出
42     第一个false,即可停止运算.因为只要有一个false,整体结果就是false
43     * 短路或运算规则:当一个式子中有多个连续的||,我们只需要找出
44     第一个true,即可停止运算.因为只要有一个true,整体结果就是true
45     */
46     System.out.println(2<4 && 4>5 && 6>3 && 4>3);
47     /*
48     * 问题总结:
49     * 以短路&&做例子
50     * 1.当遇到false的时候,是否停止运算?
51     * 是
52     * 2.对于连续的&&或者连续的||会使用短路&& 与短路||,但是如果
53     一个式子中同时出现了连续的&&和||,什么情况?
54     * 只有连续的部分遵守对应的规则,之后会用整体的结果参与后面
55     的运算
56     * 3.逻辑运算符的两边只允许使用true或false
57     */
58     int qq = 1,ww=2,xx=0;
59     boolean istru = 3<4 && 2>5 && ww++ > qq--;

```

```
56      System.out.println("ww:"+ww+"      qq:"+qq); //ww:2
      qq:1
57
58      }
59 }
```

## 4.1.2. 位运算符(了解)

### 4.1.2.1. 运算符分类

位运算符，只能作用于两个整型的变量。将两个整型变量计算出补码，然后对每一位的数字，进行类似于逻辑运算的操作。1相当于true，0相当于false。

运算符	描述
&	位与运算，对补码的每一位进行与运算。
	位或运算，对补码的每一位进行或运算。
^	位异或运算，对补码的每一位进行异或运算。
~	按位取反运算，对补码的每一位进行取反操作，包括符号位。
<<	位左移运算，将补码的每一位依次向左移动指定的位数。
>>	位右移运算，将补码的每一位一次向右移动指定的位数。左侧补符号位。
>>>	无符号位右移运算，将补码的每一位一次向右移动指定的位数。左侧补0。

#### 4.1.2.2. 示例代码

```
1  /**
2   * @Author 千锋大数据教学团队
3   * @Company 千锋好程序员大数据
4   * @Description 位运算符
5   */
6  public class JavaSyntax {
7      public static void main(String[] args) {
8          //特点:直接操作的是位
9          //为什么要使用位运算符?因为位运算符的速度比普通的快
10         /* &(按位与):全1则1,有0则0:    作用:定向清0
11         * |(按位或):全0则0,有1则1
12         * ^(按位异或):相同则0,不同则1    作用:交换两个数的值
13         * ~(按位取反):1则0,0则1
14         */
15
16         /*
17         * 过程实现
18         * byte a=4    b=5
19         * a=4的补码是 00000100    简写    0100
20         * b=5的补码是 00000101    简写    0101
21         *
22         * 0100
23         * 0101    &
24         * 0100    4
25         *
26         * 0100
27         * 0101    |
28         * 0101    5
29         *
30         *
31         * 0100
```



```

32      * 0101      ^
33      * 0001      1
34      *
35      * 0100      ~
36      * 1011
37      *
38      */
39
40      //实例一:使用&实现定向清零
41      //网段:对于IPv4,两个IP地址,他们的前三个部分相同就说在同一个网段.
42      //网关:是当前网段中的一个ip地址,我们可以简单的理解成守门员.
43      //子网掩码:用来确认ip地址是否处于当前的网段
44      //IP地址:大家上网时在网上的唯一凭证(身份证),默认肯定要跟网关在同一网段.
45
46      /*
47      * 网段: 192.168.1.0
48      * 网关: 192.168.1.1
49      * IP地址:192.168.1.34
50      * 子网掩码:255.255.255.0
51      *
52      * 192.168.1.34
53      * 255.255.255.0    &
54      *
55      * 192.168.1.0 用这个值与网段比较,相同则认为在当前网段.
56      *
57      */
58
59      /*
60      * 实例二:使用^实现交换两个数的值
61      */
62      int x = 4 ,y =5;

```

```

63      //第一种方法:借助三方变量
64      int tmp = 0;
65      tmp = x;
66      x = y;
67      y = tmp;
68      //第二种方式:使用^
69      x = x ^ y;
70      y ^= x;
71      x ^= y;
72      System.out.println("x:"+x+"    y:"+y);
73
74      /*
75      * 0101    x
76      * 0100    y      ^
77      * 0001    x
78      *
79      * 0100    y
80      * 0001    x      ^
81      * 0101    y-----5
82      *
83      * 0001    x
84      * 0101    y      ^
85      * 0100    x-----4
86      */
87
88      //移位运算符:操作位的.   >>   <<   >>>
89      //特点:操作位的
90      System.out.println(5>>1);
91
92      //解析:正数的补码和原码相同
93      //补码:      00000101
94      //右移      00000010
95      //左移      00001010

```

```

96      //注意:左移后再右移或者右移后再左移都不一定能得到原来的结果
97      //作用:用于乘除法,例子:二分查找
98
99
100     //扩展:负数的运算
101     int x = -8;
102     System.out.println(x >> 2);    // 1111 1110
103     /*
104     分析:
105     //对于>> 高位补符号位,分析时可以简写10001000
106     -8的原码: 10001000
107             反码: 11110111        注意:最高位符号位不变
108             补码: 11111000
109             运算后: 11111110      注意:高位补1
110     求原码,取反: 10000001
111             加一: 10000010      -2
112             */
113
114     System.out.println("值:" + (x >>> 2)); //
00111111 11111111 11111111 11111110
115     /*
116     分析:
117
118     -8原码: 10000000 00000000 00000000 00001000
119     -8反码: 11111111 11111111 11111111 11110111
120     -8补码: 11111111 11111111 11111111 11111000
121     -8无符号左移后结果:00111111 11111111 11111111
11111110      注意:左侧补零(高位补0)
122
123     */
124     }
125 }

```

## 4.1.3. 三目运算符

### 4.1.3.1. 三目运算符描述

运算符?:

描述:三目运算符， 是一个带有些逻辑的运算符， 基础语法如下：

布尔结果表达式/布尔变量 ? 值1 : 值2

如果问号前面的布尔值为true， 整体的结果为值1。 否则整体的结果为值2。

### 4.1.3.2. 示例代码

```
1  /**
2   * @Author 千锋大数据教学团队
3   * @Company 千锋好程序员大数据
4   * @Description 三目运算符
5   */
6  public class JavaSyntax {
7      public static void main(String[] args) {
8          //X ? Y : Z
9          // 1 X 必须是boolean类型表达式
10         // 2 Y Z 必须数据类型保持一致
11
12         //练习一
13         int a = 10, b = 20;
14         int max = a > b ? a : b; // 逻辑：计算变量a和b的
           最大值
15         System.out.println(max);
16     }
```

```

17         //练习二
18
19         int score = 99;
20         //boolean falg = score>80;
21         String str = score>80? "非常优秀" : "优秀";
22         char c = '男';
23         int i = c=='男'? 5 : (int)(4.0);
24         // y 和 z 最好是保持数据类型一致
25         // 如果不一致 也必须保证 接收的 变量能够存储 y和
z的 数据类型
26
27         System.out.println(i);
28         // 需求：大于90输出"非常优秀"，大于等于60"输出及格"，
否则小于60输出"下个班见"
29         String str2 = score>90?"非常优秀":score>=60?"及
格":"下个班见";
30         System.out.println(str2);
31     }
32 }

```

## 4.1.4. 运算符的其他

### 4.1.4.1. 表达式

1 表达式：符合一定语法规则的运算符和操作数的序列

3 i % 10

4 a = 0

5 a==0

6 5.0 + a

7 (a - b) \* c - 4

8 i < 30 && i % 10 != 0

10 表达式的值和类型

11 \* 对表达式中操作数进行运算得到的结果称为表达式的值

12 \* 表达式的值的数据类型即为表达式的类型

#### 4.1.4.2. 运算符的分类

按照运算符可以操作的数据的数量， 可以将运算符分为：一元运算符、二元运算符、三元运算符

**一元运算符:** 只能操作一个数据， 例如: + - ++ -- ! ~

**二元运算符:** 可以操作两个数据， 例如: + - \* /

**三元运算符:** 可以操作三个数据， 只有一个， 即三目运算符

#### 4.1.4.3. 优先级和结合性

- 为什么要考虑优先级和结合性？

当我们的一个表达式中出现了多个运算符时,运算符的运算顺序对最终结果的产生有决定性的作用.

- 什么是运算符的优先级？

在完成运算时处理运算符的先后顺序(见下表)

- 什么是结合性?

运算符处理操作数的顺序,分成左结合性和右结合性.

一个表达式中可能出现多个同优先级的运算符,这个时候就要通过结合性判断先算那个运算符.

- 注意:先考虑优先级,再考虑结合性

附运算符的优先级表

运算符的优先级

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	?:	三目运算符	由右至左
14	=	赋值运算符	由右至左

课上练习

```

1      int a = 2;
2      int b = 3;
3      int c = 1;
4      /**
5       * 推测:
6       * 规则:
7       * 1.先找优先级最低的    &&,将表达式分成两部分 a>b    和
a<c
8       * 2.考虑&&的结合性,左结合性,所以先算 a > b    ,结果
false
9       * 3.判断&&左边是true还是false,如果是true,继续算右边,
如果是false,直接执行短路与,不再算右边,结果就是false
10      * 4.如果左边是true,继续算右边, a<c    false ----
由于目前左边是false,所以这步不走
11      */
12      boolean d = a > b && a < c;
13      System.out.println("a:"+a); //false

```

## 4.2. 流程控制(会)

课程重点:

- 分支 if-else 的使用
- 分支 switch-case 的使用

课程难点:

- switch语句中break的使用

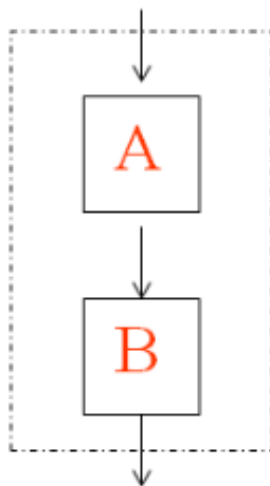


## 4.2.1. 流程控制的简介

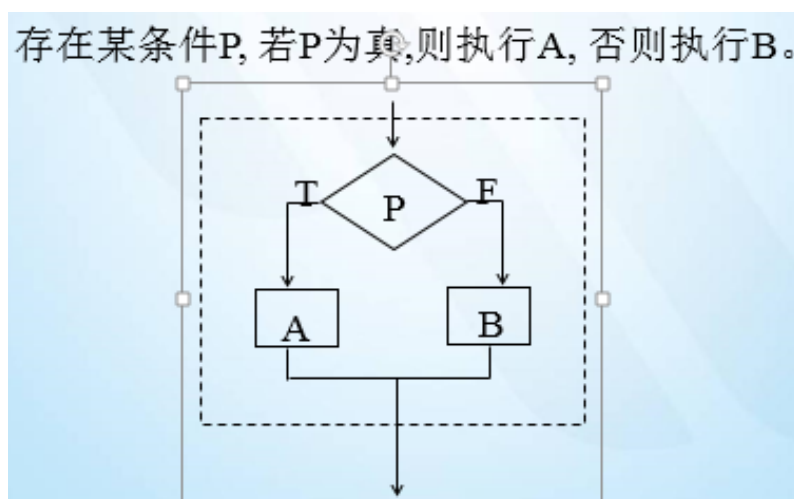
### 4.2.1.1. 程序的执行结构

在Java中，程序的执行结构分为三种

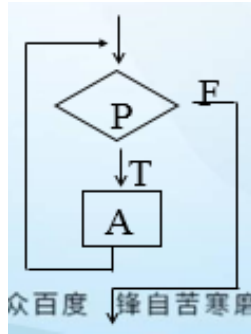
**顺序结构：**代码从上往下，逐行依次执行。是程序执行的默认结构。



**分支结构：**程序在某一个节点遇到了多种向下执行的可能性，根据条件，选择一个分支继续执行。



**循环结构：**某一段代码需要被重复执行多次。



#### 4.2.1.2. 流程控制的介绍

流程控制，就是通过指定的语句，修改程序的执行结构。按照修改的不同的执行结构，流程控制语句可以分为：

- 分支流程控制语句：
  - 将程序，由顺序结构，修改为分支结构
- 循环流程控制语句：
  - 将程序，由顺序结构，修改为循环结构

### 4.2.2. 分支流程控制 - if

#### 4.2.2.1. if流程图

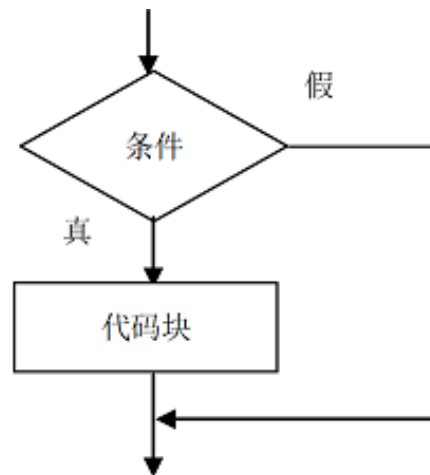
```
1  st=>start: 开始
2  e=>end: 结束
3  con=>condition: 判断条件
4  op1=>operation: 分支判断逻辑
5  s1=>operation: 分支语句1
6  s2=>operation: 分支语句2
7
8  st(right)->op1(right)->con(yes,right)->s1->e
9  con(no)->s2->e
```

if语句有四种使用形式:

- if(条件){...}
- if(条件){...}else{...}
- if(条件){...}else if(条件){...}else{...}
- if(条件){ if(条件){...} } else {...}

#### 4.2.2.2. 简单if语句

```
1  if (/* 条件判断 true or false */) {  
2      // 条件结果为true执行大括号以内的语句  
3  }  
4  /*  
5  执行流程:  
6      代码运行到if分支结构, 首先判断if之后的条件是否为true, 如果为  
7      true, 执行大括号里面的语句, 如果为false直接执行大括号之外的语句,  
8      */
```



```
1  示例1: java成绩如果大于60, 奖励一颗糖
2
3      //简单的if语句:
4      //成绩如果大于60    给奖励
5      int score = 10;
6      if(score>60){
7          System.out.println("给颗糖");
8      }
```

## 课上练习

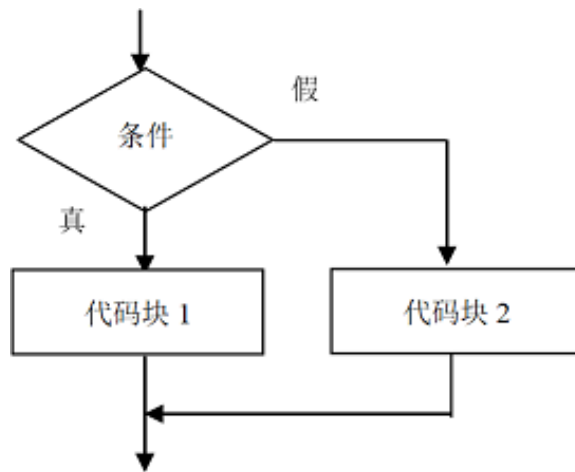
Java成绩大于98分, 而且Html成绩大于80分, 老师奖励他; 或者Java成绩等于100分, Html成绩大于70分, 老师也可以奖励他。

```
1      if((score1 >98 && score2 > 80 ) || ( score1 == 100
2      && score2 > 70 )){
3          //奖励
4      }
```

### 4.2.2.2. if-else语句

```
1  if (condition) {
2      // 代码段1
3  }
4  else {
5      // 代码段2
6  }
```

逻辑: condition是一个boolean类型的变量, 或者一个boolean结果的表达式. 如果condition的值为true, 则代码段1执行, 否则, 代码段2执行



```
1  /**
2   * @Author 千锋大数据教学团队
3   * @Company 千锋好程序员大数据
4   * @Description if-else的基础语法
5   */
6  public class JavaSyntax {
7      public static void main(String[] args) {
8          int score = 99;
9          if (score >= 60) {
10             System.out.println("成绩及格了! ");
11         }
12         else {
13             System.out.println("成绩不及格! ");
14         }
15     }
16 }
```

## 课上练习一

如果是男生就永远18岁，否则永远16岁。

```

1      // 如果是男生    就永远18岁
2      // 如果是 女生    永远16岁
3      char c = '女';
4      if(c == '男'){// boolean 结果是true  执行if中    否
则执行else中的
5          System.out.println("永远18岁");
6      }else{
7          System.out.println("永远16岁");
8      }

```

## 课上练习二

### 买彩票

如果体彩中了500万，我买车、买房、非洲旅游

如果没中，继续买。

```

1  import java.util.Scanner;
2  public static void main(String[] args){
3      //1创建input对象  作用:可以从键盘接收字符串,后面会讲
4      Scanner input=new Scanner(System.in);
5      //2提示
6      System.out.println("中500万吗?Y/N");
7      String answer=input.next();//这里是实际接收
8      //3判断
9      if(answer.equals("y")){ //字符串的判断使用equals方法
10         System.out.println("买房、买车、欧洲旅游...");
11     }else{
12         System.out.println("继续买....");
13     }
14 }

```

### 4.2.2.3. if语法进阶

```
1  if (condition1) {
2      // 代码段1
3  }
4  else if (condition2) {
5      // 代码段2
6  }
7  else {
8      // 代码段3
9  }
```

逻辑: 先判断condition1, 如果condition1成立, 执行代码段1; 如果condition1不成立, 再判断condition2, 如果condition2成立, 执行代码段2, 否则执行代码段3

```
1  /**
2   * @Author 千锋大数据教学团队
3   * @Company 千锋好程序员大数据
4   * @Description if-else的语法进阶
5   */
6  public class JavaSyntax {
7      public static void main(String[] args) {
8          int score = 99;
9          if (score < 60) {
10             System.out.println("成绩不及格! ");
11         }
12         else if (score < 80) {
13             System.out.println("良! ");
14         }
15         else {
16             System.out.println("优! ");
17         }
```

```
18     }
19 }
```

## 课上练习

如果成绩大于90并且是男生就送个女朋友，成绩大于90并且是女生送个男朋友，否则...

```
1  char c = '女';
2  int score = 10;
3  if(score>90 && c=='男'){
4      System.out.println("给送个女朋友");
5  }else if(score>90 && c=='女'){
6      System.out.println("给送个男朋友");
7  }else{
8      System.out.println("啥都没有，自己买");
9  }
```

### 4.2.2.4. 嵌套if语句

```
1  if(条件1) {
2
3      if(条件2) {
4
5          代码块1
6
7      } else {
8
9          代码块2
10
11     }
12
13 } else {
14
```

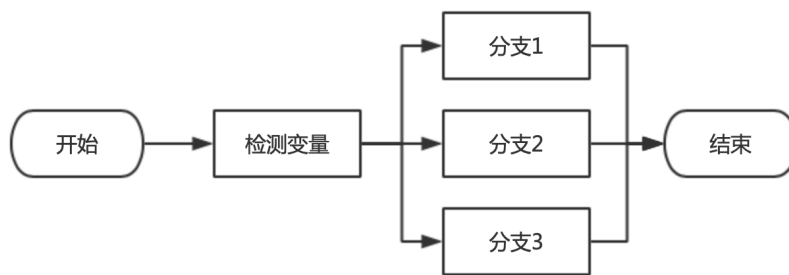


```
15      代码块3
16
17  }
```

```
1  /**
2   * @Author 千锋大数据教学团队
3   * @Company 千锋好程序员大数据
4   * @Description if-else的语法进阶
5   */
6   如果成绩大于90    如果是男生 送个女朋友,    如果是女生送个男朋友
7
8   // 如果成绩大于90    如果是男生    如果是女生
9       int score = 10;
10      if(score>90){
11          if(c=='男'){
12              System.out.println("给送个女朋友");
13          }else{
14              System.out.println("给送个男朋友");
15          }
16      }
```

## 4.2.3. 分支流程控制 - switch

### 4.2.3.1. switch流程图



#### 4.2.3.2. switch的基础语法

```
1  switch(表达式expr){ //int,byte,short,char,enum,String
2      case const1:
3          statement1;
4          break;
5      case const2:
6          statement2;
7          break;
8      ... ..
9      case constN:
10         statementN;
11         break;
12     default:
13         statement_dafault;
14         break;
15 }
```

#### 程序逻辑：

- 检测某一个变量的值， 从上往下依次与每一个case进行校验、匹配
- 如果变量的值和某一个case后面的值相同，则执行这个case后的语句
- 如果变量的值和每一个case都不相同，则执行default后的语句

### 4.2.3.3. switch的语法规则

- 1 1. 表达式expr的值必须是下述几种类型之一:
- 2
- 3       byte、short、int、char、enum(枚举); java7之后可以是  
String。
- 4
- 5 1. case子句中的值const 必须是常量值(或final的变量), case中的值  
不能是一个范围
- 6 2. 所有case子句中的值应是不同的, 否则会编译出错;
- 7 3. default子句是可选的 (不是必须的)
- 8 4. break语句用来在执行完一个case分支后使程序跳出switch语句块; 否  
则会继续执行下去

### 课上练习一

#### 简单实现switch语句

```
1      int i = 1;
2      switch(i){
3          case 1:
4              System.out.println("Hello World!");
5              break;
6          case 2:
7              System.out.println("Hello World!2");
8          case 3:
9              System.out.println("Hello World3");
10             break;
11         default:
12             System.out.println("Haaaa");
13             break;
14     }
```

### 课上练习二

## 判断 春夏秋冬

```
1 Scanner sc = new Scanner(System.in);
2     String str = sc.next();
3     switch(str){
4         case "春天":
5             System.out.println("春暖花开");
6             break;
7         case "夏天":
8             System.out.println("闷热");
9             break;
10        case "秋天":
11            System.out.println("秋高气爽");
12            break;
13        case "冬天":
14            System.out.println("滴水成冰");
15            break;
16        default:
17            System.out.println("火星的");
18            break;
19    }
```

### 4.2.3.4. case穿透

```
1 /**
2  * @Author 千锋大数据教学团队
3  * @Company 千锋好程序员大数据
4  * @Description switch结构
5  */
6 public class JavaSyntax {
7     public static void main(String[] args) {
8         int season = 1;
```

```
9      switch (season) {
10          case 1:
11              System.out.println("春天");
12          case 2:
13              System.out.println("夏天");
14          case 3:
15              System.out.println("秋天");
16          case 4:
17              System.out.println("冬天");
18          default:
19              System.out.println("错误季节");
20      }
21  }
22 }
```

上述代码中， switch结构捕获变量season的值。 变量的值和第一个case是匹配的， 应该输出的结果是 "春天"。 但实际上的输出结果却是从春天开始的每一个输出。

因为在switch结构中有“穿透性”。


### 穿透性:

指的是， 当switch的变量和某一个case值匹配上之后， 将会跳过后续的case或者default的匹配， 直接向后穿透。

```

public class Program {
    public static void main(String[] args) {
        int season = 1;
        switch (season) {
            case 1:
                System.out.println("春天");
            case 2:
                System.out.println("夏天");
            case 3:
                System.out.println("秋天");
            case 4:
                System.out.println("冬天");
            default:
                System.out.println("错误季节");
        }
    }
}

```




为了杜绝穿透， 可以使用关键字 **break**:

```

public class Program {
    public static void main(String[] args) {
        int season = 1;
        switch (season) {
            case 1:
                System.out.println("春天");
            case 2:
                System.out.println("夏天");
                break;
            case 3:
                System.out.println("秋天");
            case 4:
                System.out.println("冬天");
            default:
                System.out.println("错误季节");
        }
    }
}

```



这一段程序， season的值， 和case 1匹配。 因此会输出“春天”， 然后向下穿透， 输出“夏天”。

此时， 遇到了关键字 break。 将会结束穿透， 直接结束switch结构。

因此， 此时的输出结果是：

春天

夏天

