

# 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?  
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
2. 原码反码补码的理解  
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. 掌握逻辑运算符的使用
2. 了解位运算符的使用
3. 掌握三目运算符的使用
4. 掌握常见多运算符表达式的运算
5. 掌握分支的使用场景
6. 掌握if分支语句的使用
7. 掌握switch循环语句的使用

## 三 教学导读

### 3.1. 运算符续

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

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。

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。

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）。

下面我们就展开分支语句的学习

### 3.3. 分支结构的概念

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。 <b>真真为真, 其余为假</b>	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。 <b>假假为假, 其余为真</b>	false true	True
^	逻辑异或, <b>相同为假, 不同为真。</b> <b>以前用于密码加密</b>	true^false	True
!	逻辑非, 非真即假, 非假既真。	!true	False
&&	短路与,如果前面的结果可以决定整体的运算结果, <b>后面的表达式不参与运算</b>	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

```

/**
 * @Author 干锋大数据教学团队
 * @Company 干锋好程序员大数据
 * @Description 逻辑运算符
 */
public class JavaSyntax {
    public static void main(String[] args) {
        //特点:1.运算符的两边的元素一定是true/false 2.结果一定是true/false

        //逻辑与 & 逻辑或 | 逻辑异或 ^ 逻辑非 !
        //&:一假则假,全真才真
        // |:一真则真,全假才假
        //!:真则假,假则真
        boolean flag2 = false&false;// & 必须是左右两边都是 true 结果才是true
        boolean a = false | false;// | 只要有一个true 结果就是true 两边都是false 结果才
为 false

        System.out.println("flag2 = "+ flag2);
        System.out.println("a = "+ a);

        // ! 逻辑非
        boolean e = !true;
        System.out.println("e = "+ e);

        // ^ 异或 相同则为false 不同为true
        boolean f = true ^ false;
        System.out.println("f = "+ f);

        int a = 4;

        int c = 5;

```

```

        boolean b4 = ++a>6 & ++c>3;
        System.out.println(a);
        System.out.println(c);

//短路与 && 和 短路或 ||
//注意:短路与,短路或的最终结果与逻辑与,逻辑或的一样.
//举例:
        System.out.println(2 >= 3 && 3 <= 4);

        /*
        * 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
        * 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
        */
        System.out.println(2<4 && 4>5 && 6>3 && 4>3);
        /*
        * 问题总结:
        * 以短路&&做例子
        * 1.当遇到false的时候,是否停止运算?
        * 是
        * 2.对于连续的&&或者连续的||会使用短路&& 与短路||,但是如果一个式子中同时出现了连续的&&和||,什么情况?
        * 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的运算
        * 3.逻辑运算符的两边只允许使用true或false
        */
        int qq = 1,ww=2,xx=0;
        boolean istru = 3<4 && 2>5 && ww++ > qq--;
        System.out.println("ww:"+ww+"    qq:"+qq); //ww:2    qq:1

    }
}

```

## 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

```
/**
 * @Author 干锋大数据教学团队
 * @Company 干锋好程序员大数据
 * @Description 位运算符
 */
public class JavaSyntax {
    public static void main(String[] args) {
        //特点:直接操作的是位
        //为什么要使用位运算符?因为位运算符的速度比普通的快
        /* &(按位与):全1则1,有0则0:    作用:定向清0
         * |(按位或):全0则0,有1则1
         * ^(按位异或):相同则0,不同则1    作用:交换两个数的值
         * ~(按位取反):1则0,0则1
         */

        /*
         * 过程实现
         * byte a=4    b=5
         * a=4的补码是 00000100    简写    0100
         * b=5的补码是 00000101    简写    0101
         *
         * 0100
         * 0101    &
         * 0100    4
         *
         * 0100
         * 0101    |
         * 0101    5
         *
         *
         * 0100
         * 0101    ^
        */
    }
}
```

```

        * 0001    1
        *
        * 0100    ~
        * 1011
        *
        */

/*
 * 实例:使用^实现交换两个数的值
 */
int x = 4 ,y =5;
//第一种方法:借助三方变量
int tmp = 0;
tmp = x;
x = y;
y = tmp;
//第二种方式:使用^
x = x ^ y;
y ^= x;
x ^= y;
System.out.println("x:"+x+"    y:"+y);

/*
 * 0101    x
 * 0100    y    ^
 * 0001    X
 *
 * 0100    y
 * 0001    X    ^
 * 0101    y-----5
 *
 * 0001    X
 * 0101    y    ^
 * 0100    x-----4
 */

//移位运算符:操作位的.   >>   <<
//特点:操作位的
System.out.println(5>>1);

//解析:正数的补码和原码相同
//补码:    00000101
//右移    00000010
//左移    00001010
//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

}

}

```

### 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

```
/**
 * @Author 千锋大数据教学团队
 * @Company 千锋好程序员大数据
 * @Description 三目运算符
 */
public class JavaSyntax {
    public static void main(String[] args) {
        //X ? Y : Z
        // 1 X 必须是boolean类型表达式
        // 2 Y Z 必须数据类型保持一致

        //练习一
        int a = 10, b = 20;
        int max = a > b ? a : b;    // 逻辑：计算变量a和b的最大值
        System.out.println(max);

        //练习二

        int score = 99;
        //boolean flag = score>80;
        String str = score>80? "非常优秀" : "优秀";
        char c = '男';
        int i = c=='男'? 5 : (int)(4.0);
        // y 和 z 最好是保持数据类型一致
        // 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

        System.out.println(i);
        // 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"
        String str2 = score>90?"非常优秀":score>=60?"及格":"下个班见";
        System.out.println(str2);
    }
}
```

#### 4.1.4. 运算符的其他

#### 4.1.4.1. 表达式

表达式：符合一定语法规则的运算符和操作数的序列

```
i % 10
a = 0
a == 0
5.0 + a
(a - b) * c - 4
i < 30 && i % 10 != 0
```

表达式的值和类型

- \* 对表达式中操作数进行运算得到的结果称为表达式的值
- \* 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

```
int a = 2;
int b = 3;
int c = 1;
/**
 * 推测:
 * 规则:
 * 1. 先找优先级最低的 &&, 将表达式分成两部分 a>b 和 a<c
 * 2. 考虑&&的结合性, 左结合性, 所以先算 a > b , 结果 false
 * 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
 * 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
 */
boolean d = a > b && a < c;
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:

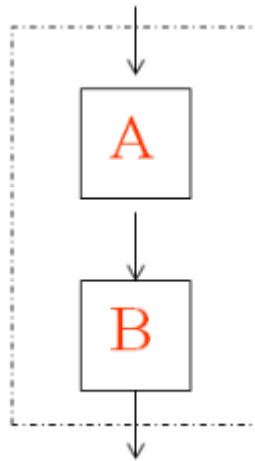
- switch语句中break的使用

## 4.2.1. 流程控制的简介

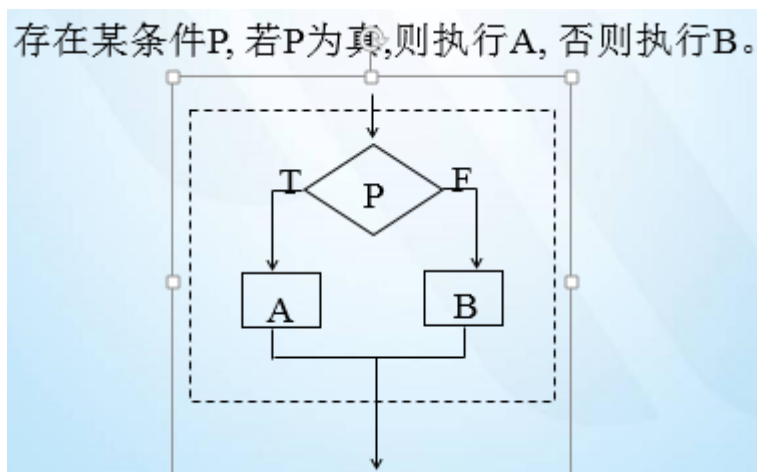
### 4.2.1.1. 程序的执行结构

在Java中，程序的执行结构分为三种

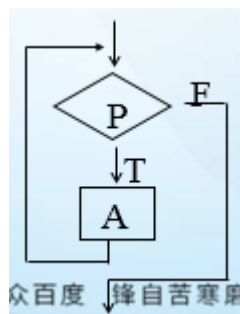
**顺序结构：**代码从上往下，逐行依次执行。是程序执行的默认结构。



**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。



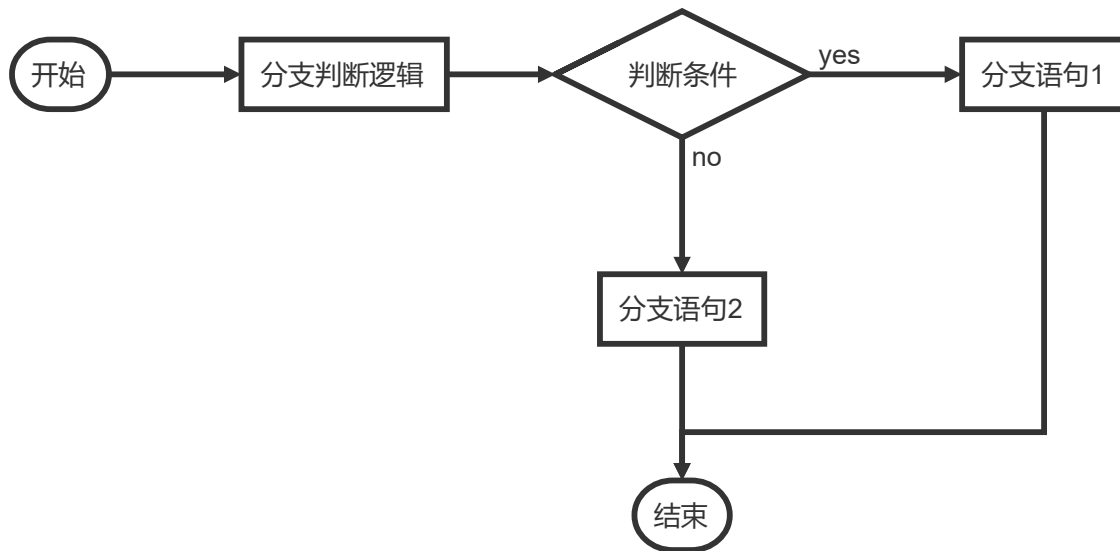
### 4.2.1.2. 流程控制的介绍

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



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

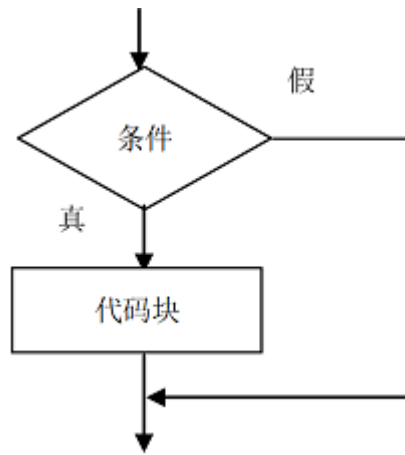
```
if (/* 条件判断 true or false */) {  
    // 条件结果为true执行大括号以内的语句  
}
```

/\*

执行流程：

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，

\*/



示例1: java成绩如果大于60, 奖励一颗糖

```
//简单的if语句:  
//成绩如果大于60 给奖励  
int score = 10;  
if(score>60){  
    System.out.println("给颗糖");  
}
```

### 课上练习

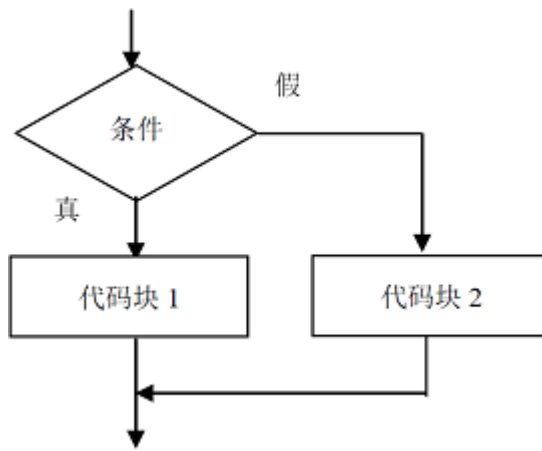
Java成绩大于98分, 而且Html成绩大于80分, 老师奖励他; 或者Java成绩等于100分, Html成绩大于70分, 老师也可以奖励他。

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

### 4.2.2.2. if-else语句

```
if (condition) {  
    // 代码段1  
}  
else {  
    // 代码段2  
}
```

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



```
/**
 * @Author 千锋大数据教学团队
 * @Company 千锋好程序员大数据
 * @Description if-else的基础语法
 */
public class JavaSyntax {
    public static void main(String[] args) {
        int score = 99;
        if (score >= 60) {
            System.out.println("成绩及格了! ");
        }
        else {
            System.out.println("成绩不及格! ");
        }
    }
}
```

### 课上练习一

如果是男生就永远18岁，否则永远16岁。

```
// 如果是男生 就永远18岁
// 如果是 女生 永远16岁
char c = '女';
if(c == '男'){// boolean 结果是true 执行if中 否则执行else中的
    System.out.println("永远18岁");
}else{
    System.out.println("永远16岁");
}
```

### 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。

```
import java.util.Scanner;
```

```

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

```

#### 4.2.2.3. if语法进阶

```

if (condition1) {
    // 代码段1
}
else if (condition2) {
    // 代码段2
}
else {
    // 代码段3
}

```

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

```

/**
 * @Author 干锋大数据教学团队
 * @Company 干锋好程序员大数据
 * @Description if-else的语法进阶
 */
public class JavaSyntax {
    public static void main(String[] args) {
        int score = 99;
        if (score < 60) {
            System.out.println("成绩不及格! ");
        }
        else if (score < 80) {
            System.out.println("良! ");
        }
        else {
            System.out.println("优! ");
        }
    }
}

```

## 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...

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

### 4.2.2.4. 嵌套if语句

```
if(条件1) {

    if(条件2) {

        代码块1

    } else {

        代码块2

    }

} else {

    代码块3

}
```

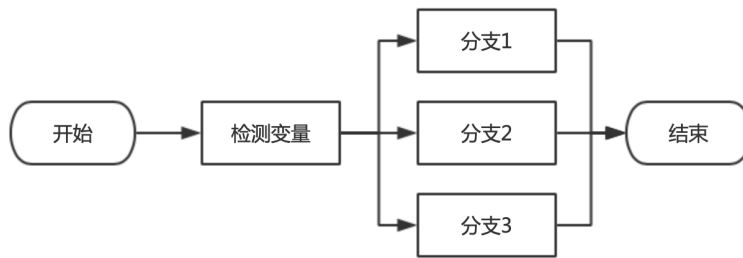
```
/**
 * @Author 干锋大数据教学团队
 * @Company 干锋好程序员大数据
 * @Description if-else的语法进阶
 */
如果成绩大于90 如果是男生 送个女朋友, 如果是女生送个男朋友

// 如果成绩大于90 如果是男生 如果是女生
int score = 10;
if(score>90){
    if(c=='男'){
        System.out.println("给送个女朋友");
    }else{
        System.out.println("给送个男朋友");
    }
}
```

```
}
```

### 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

```
switch(表达式expr){  
    case const1:  
        statement1;  
        break;  
    case const2:  
        statement2;  
        break;  
    ... ...  
    case constN:  
        statementN;  
        break;  
    default:  
        statement_dafault;  
        break;  
}
```

#### 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. 表达式expr的值必须是下述几种类型之一：

byte、short、int、char、enum(枚举); java7之后可以是String。

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
2. 所有case子句中的值应是不同的, 否则会编译出错;
3. default子句是可选的 (不是必须的)
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

```
int i = 1;
switch(i){
    case 1:
        System.out.println("Hello World!");
        break;
    case 2:
        System.out.println("Hello World!2");
    case 3:
        System.out.println("Hello World3");
        break;
    default:
        System.out.println("Haaaa");
        break;
}
```

## 课上练习二

判断 春夏秋冬

```
Scanner sc = new Scanner(System.in);
String str = sc.next();
switch(str){
    case "春天":
        System.out.println("春暖花开");
        break;
    case "夏天":
        System.out.println("闷热");
        break;
    case "秋天":
        System.out.println("秋高气爽");
        break;
    case "冬天":
        System.out.println("滴水成冰");
        break;
    default:
        System.out.println("火星的");
        break;
}
```

#### 4.2.3.4. case穿透

```
/**
 * @Author 干锋大数据教学团队
 * @Company 干锋好程序员大数据
 * @Description switch结构
 */
public class JavaSyntax {
    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");
        }
    }
}
```

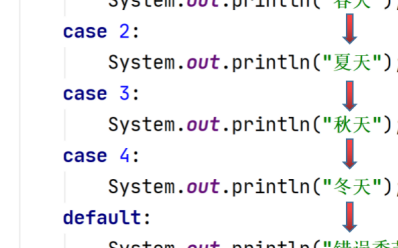
上述代码中，switch结构捕获变量season的值。变量的值和第一个case是匹配的，应该输出的结果是"春天"。但实际上的输出结果却是从春天开始的每一个输出。

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

## 五 实战应用

### 5.1 实战案例一

输入四个季节：如果春天，输出“春暖花开”，如果是夏天，输出“夏日炎炎”，如果是秋天，则输出“秋高气爽”，如果是冬天，则输出“安静祥和”。

```
// 1
import java.util.Scanner;

class Demo8
{
    public static void main(String[] args)
    {
        // == 判断基本数据类型内容是否相等
        // 引用数据类型 == 判断内存地址
        // 判断内容 String 类型 判断内容 equals

        Scanner sc = new Scanner(System.in);
        String str = sc.next();

        // boolean falg= str.equals("春天");
        if(str.equals("春天")){
            System.out.println("春暖花开");
        }
    }
}
```

```

    }else if(str.equals("夏天")){
        System.out.println("夏日炎炎");
    }else if(str.equals("秋天")){
        System.out.println("秋高气爽");
    }else if(str.equals("冬天")){
        System.out.println("安静祥和");
    }
}
}

```

## 5.2 实战案例二

利用switch实现一个计算器操作

```

Scanner sc = new Scanner(System.in);
System.out.println("请输入第一个操作数:");
int a = sc.nextInt();
System.out.println("请输入运算符");
String str = sc.next();
System.out.println("请输入第二个操作:");
int b = sc.nextInt();
switch(str){
    case "+":
        System.out.println(a+"+"+b+"="+a+b);
        break;
    case "-":
        System.out.println(a+"-"+b+"="+a-b);
        break;
    case "*":
        System.out.println(a+"*"+b+"="+a*b);
        break;
    case "/":
        System.out.println(a+"/"+b+"="+a/b);
        break;
    case "%":
        System.out.println(a+"%"+b+"="+a%b);
        break;
}

```

## 5.3 实战案例三

打印输出指定的月份的天数

```

Scanner sc = new Scanner(System.in);

int month = sc.nextInt();

switch(month){

```

```
        case 1:

        case 3:

        case 5:

        case 7:

        case 8:

        case 10:

        case 12:

            System.out.println("31天");

            break;

        case 2:

            System.out.println("28天");

            break;

        case 4:

        case 6:

        case 9:

        case 11:

            System.out.println("30天");

            break;

    }

    System.out.println("Hello World!");

}
```

## 六 教学总结

### 6.1 运算符

课程重点：

1. 逻辑运算符                      //特点:1.运算符的两边的元素一定是true/false   2.结果一定是true/false

```
//逻辑与 & 逻辑或 | 逻辑异或 ^ 逻辑非 !
//&:一假则假,全真才真
    // |:一真则真,全假才假
    //!:真则假,假则真
```

```
        //短路与 && 和 短路或 ||
//注意:短路与,短路或的最终结果与逻辑与,逻辑或的一样。
```

```
/*
    * 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
    * 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
*/
```

## 2.三目运算符

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。

课程难点：

1.位运算符

## 6.2 流程控制概述

理解三种流程控制原理

## 6.3 流程控制之分支语句

课程重点：

1.选择结构包括两种：分别为if语句和switch语句

2.if语句有四种使用形式

第一种，只有if语句

```
if(布尔表达式){
}
```

第二种 双分支if语句（有if和else语句）

```
if(布尔表达式){
}
else{
}
```

第三种 多分支if语句（有if else if else）

```
if(布尔表达式){
}
else if(布尔表达式){
}
else{
}
```

#### 第四种 嵌套

```
if(布尔表达式){  
    if(布尔表达式){  
        }  
    else{  
        }  
    }  
else{  
    }  
}
```

3.switch语句重点

4.两种选择语句的区别

课程难点:

switch通过break实现的穿透性

## 七 课后作业

1.写出结果。

```
class Demo  
{  
    public static void main(String[] args)  
    {  
        int x=0,y=1;  
        if(++x==y--&x++==1 || --y==0) 短路结构，一经确定，终止执行  
            System.out.println("x="+x+",y="+y);  
        else  
            System.out.println("y="+y+",x="+x);  
    }  
}
```

2.

写出输出结果。

```
class Demo  
{  
    public static void main(String[] args)  
    {  
        int a=3,b=8;  
  
        int c=(a>b)?a++:b++;  
        System.out.println("a="+a+"\tb="+b+"\tc="+c);  
  
        int d=(a>b)?++a:++b;  
        System.out.println("a="+a+"\tb="+b+"\td="+d);  
  
        int e=(a<b)?a++:b++;  
        System.out.println("a="+a+"\tb="+b+"\te="+e);  
  
        int f=(a<b)?++a:++b;  
        System.out.println("a="+a+"\tb="+b+"\tf="+f);  
    }  
}
```

```
}
```

3. 写出结果。

```
class Demo
{
    public static void main(String[] args)
    {
        int m=0,n=3;
        if(m>0)
        {
            if(n>2)
                System.out.println("A");
            else
                System.out.println("B");
        }
    }
}
```

4: 输入一个年份, 判断是否是闰年(能被4整除并且不能被100整除或者能被400整除的就是闰年?)

```
import java.util.*;
class lx2
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("请输入一个年份");
        int n=sc.nextInt();
        if((n%4==0&& n%100!=0) || n%400==0)
            System.out.print(n+"是闰年");
        else
            System.out.print(n+"不是闰年");
    }
}
```

5.

已知学生成绩以100分为满分, 共分5个等级: A,B,C,D,E。

90~100为等级A, 80~89为等级B, 70~79为等级C,

60~69为等级D, 0~59为等级E。

要求定义一个成绩变量, 当成绩变化时, 可直接知道该成绩对应的等级。

例如: 当成绩为100时, 该学生的等级是A。

6.

写出输出结果。

```
class Demo
{
    public static void main(String[] args)
    {
        show(0);
        show(1);
    }
    public static void show(int i)
    {
        switch(i)
        {

```



```

        default:
            i+=2;
        case 1:
            i+=1;
        case 4:
            i+=8;
        case 2:
            i+=4;
    }
    System.out.println("i="+i);
}
}
}
7. 写出输出的结果.
class Demo
{
    public static void main(String[] args)
    {
        int x=0,y=1;

        if(++x==y-- & x++==1 || --y==0)

            System.out.println("x="+x+",y="+y);
        else
            System.out.println("y="+y+",x="+x);
    }
}

```

## 八 解决方案

### 8.1 应用场景

### 8.2 核心面试题

因为switch结构做匹配，最底层还是转化为int类型进行比对，long类型如果转为int类型，可能存在精度损失，无法进行匹配，所以不能用long类型

- 1.switch是否能作用在byte上，是否能作用在long上，是否能作用在String上？ 不可以使用long
- 2.switch中default的位置是否必须出现在最后？ 不一定

不一定，不影响