

# 第3章 选择

## 目录

contents



**3.1 布尔数据类型和逻辑运算符**



**3.2 条件语句**



**3.3 条件表达式**



**3.4 操作符的优先级和结合规则**

# 3.1 布尔数据类型和逻辑运算符

◆boolean类型的值有真(true)或假(false)。其字面量只有true, false。

◆关系运算符: <, <=, >, >=, ==, !=

- 关系运算符的计算结果是boolean类型

- boolean类型不能与其它数据类型混合运算 (包括类型转换)

◆布尔运算符: !, &&, ||, ^, &, |

&	真	假
真	真	假
假	假	假

	真	假
真	真	真
假	真	假

^	真	假
真	假	真
假	真	假

条件逻辑运算符和无条件逻辑运算符的区别:

A&&B: 如果A求值结果为false, B不再求值, 结果为false

A&B: 如果A求值结果为false, B还要求值, 结果为false

◆&&, ||为条件逻辑运算符: (x>0) && (x<9)

◆&, |为无条件逻辑运算符

◆^ 异或

# 3.1 位运算(~~~)

## Java支持7种位运算符:

- 按位与 &: 两位同为1, 结果为1 (用于掩码操作)
- 按位或 |: 任一位为1, 结果为1 (用于置位操作)
- 按位异或 ^: 两位不同则为1 (用于交换变量、去重)
- 按位取反 ~: 0变1, 1变0 (含符号位操作)
- 左移 <<: 符号位不变, 低位补0 (等价于 $\times 2^n$ )
- 右移 >>: 符号位填充高位 (正数补0, 负数补1)
- 无符号右移 >>>: 高位补0 (负数转为正数)

## 3.2 条件语句

### ◆if语句

- 简单的if语句
- if-else语句
- if嵌套语句

### ◆switch语句

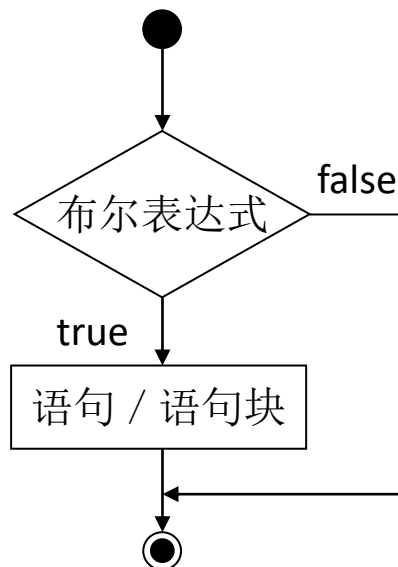
### ◆条件表达式

## 3.2 条件语句

### 简单if语句

#### ◆语法

`if (bool-expression)`  
*statement or block*



```
if (radius >= 0) {  
    area = radius * radius * PI;  
    System.out.println("The area for the circle of radius "  
        + radius + " is " + area);  
}
```

注意：if后面圆括号里必须是求值结果为**boolean**的表达式。

由于**boolean**不能和任何其它类型数据类型相互转换，因此Java里就不可能出现C++里新手常犯的下面错误

```
if(i = 0){  
  
}
```

```
if(i){  
  
}
```

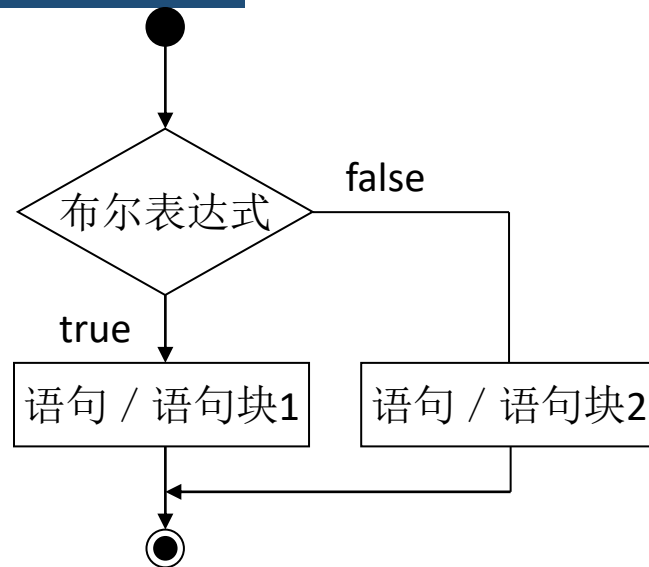
## 3.2 条件语句

### if-else语句

#### ◆语法

```
if(bool-expression)  
    statement or block 1  
else  
    statement or block 2
```

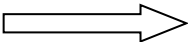
```
if (radius >= 0) {  
    area = radius * radius * 3.14159;  
    System.out.println("The area for the circle of radius "  
        + radius + " is " + area);  
}  
else { System.out.println("Negative input"); }
```



## 3.2 条件语句

### 嵌套if语句和else if

```
if (score > 90.0)
    grade = 'A' ;
else
    if (score >= 80.0)
        grade = 'B' ;
    else
        if (score >= 70.0)
            grade = 'C' ;
        else
            if (score >= 60.0)
                grade = 'D' ;
            else
                grade = 'F' ;
```

等价于  


```
if (score > 90.0)
    grade = 'A' ;
else if (score >= 80.0)
    grade = 'B' ;
else if (score >= 70.0)
    grade = 'C' ;
else if (score >= 60.0)
    grade = 'D' ;
else
    grade = 'F' ;
```

## 3.2 条件语句

### 注意

#### ◆else语句与同一块中最近的if语句匹配

```
int i = 1, j = 2, k = 3;  
if (i > j)  
    if(i > k)  
        System.out.println( "A" );  
else  
    System.out.println( "B" );
```

相当于

```
int i = 1, j = 2, k = 3;  
if (i > j)  
    if(i > k)  
        System.out.println( "A" );  
else  
    System.out.println( "B" );
```



## 3.2 条件语句

### 提示

◆ 尽量避免使用if语句将值(布尔值)赋值给布尔变量，应直接将值赋给这个变量。

```
if (number % 2 == 0)
    even = true;
else
    even = false; //新手
```

等价于

```
even = (number % 2 == 0); //高手
```

## 3.2 条件语句

### 提示

◆避免在条件表达式中使用比较操作符判断布尔变量的真假

```
if (even == true)
    System.out.println( "It is even. " );
```

等价于

```
if (even)
    System.out.println( "It is even. " );
```

## 3.2 条件语句

### 例 税款计算问题

◆美国的个人所得税根据纳税人情况和须纳税收入进行计算。编写程序，用户输入纳税人情况和须纳税收入，计算出2002年的所得税。

Tax rate	Single filers	Married filing jointly or qualifying widow/widower	Married filing separately	Head of household
10%	Up to \$6,000	Up to \$12,000	Up to \$6,000	Up to \$10,000
15%	\$6,001 - \$27,950	\$12,001 - \$46,700	\$6,001 - \$23,350	\$10,001 - \$37,450
27%	\$27,951 - \$67,700	\$46,701 - \$112,850	\$23,351 - \$56,425	\$37,451 - \$96,700
30%	\$67,701 - \$141,250	\$112,851 - \$171,950	\$56,426 - \$85,975	\$96,701 - \$156,600
35%	\$141,251 - \$307,050	\$171,951 - \$307,050	\$85,976 - \$153,525	\$156,601 - \$307,050
38.6%	\$307,051 or more	\$307,051 or more	\$153,526 or more	\$307,051 or more

Example:见教材程序清单3-5

## 3.2 条件语句

### switch语句

#### ◆语法

```
switch(expression) {  
    case value1:  
        statement(s)  
        break;  
    case value2:  
        statement(s)  
        break;  
    ...  
    default :  
        statement(s)  
}
```

- switch语句的判断条件expression的计算结果只能是 **byte, char, short, int, enum**等不大于int的类型。
- value1-valueN必须与判断条件expression类型相同, 不能用逗号分割且为常量表达式, 不能是变量。
- 每个case一个判断值, 后面可以跟多条语句, 这些语句可以不用大括号括起来。
- 程序将从第一个匹配的case子句处开始执行后面的所有代码 (包括后面case子句中的代码)。可以使用break语句跳出switch语句。
- default语句是可选的。当所有case子句条件都不满足时执行。default不一定在最后, 匹配后执行后面的所有代码。

# 3.2 switch语句(~~~~~)

## Java7+支持类型

支持String、Enum类型，这使switch语句的应用范围更广。

如使用String类型，可直接根据字符串值进行分支判断，简化代码逻辑。

## Java 12+新特性

- ->箭头语法：
  - 简化了switch语句，使代码更简洁
  - 可以匹配多个值（多个值逗号分隔）
  - 无需break，避免危险穿透
  - 返回->后面表达式的值，switch语句变成switch表达式（可以是void类型，表示无返回值）
  - **和传统的switch结构不能混用**
- yield 返回值
  - 表示switch语句的返回值，变成switch表达式
  - 可以在传统switch结构和->结构中使用(传统结构无break，也不会穿透)
  - Yield必须返回值
- 每个分支返回值类型相同，或者抛出异常

## Java23增加支持所有的原始类型

// 箭头语法示例

```
String season = switch (month) {  
    case 12, 1, 2 -> "冬季";  
    case 3, 4, 5 -> "春季";  
    case 6, 7, 8 -> "夏季";  
    case 9, 10, 11 -> "秋季";  
    default -> "无效月份";  
};
```

// 返回值的 Switch 表达式

```
int days = switch (month) {  
    case 1, 3, 5, 7, 8, 10, 12 -> 31;  
    case 4, 6, 9, 11 -> 30;  
    case 2 -> {  
        if (isLeapYear) yield 29;  
        else yield 28;  
    }  
    default -> throw new IllegalArgumentException();  
};
```

## 3.2 switch语句(~~~~~)

- 始终包含 default 分支
- default建议放在最后
- Java建议包含null的处理
- 枚举处理：推荐覆盖所有枚举值（可以没有default，但仍然建议包含null的分支）

```
Test1.java x
1 public class Test1 {
2     public static void main(String[] args) {
3         Integer i=null;
4         switch(i){
5             case 1->System.out.println("1");
6             case null->System.out.println("错误: 空值");
7             default -> System.out.println("0");
8         }
9     }
10 }
```

运行 Test1 x

```
D:\Applications\Develop\Kits\jdk2101\bin\java.exe "-javaager
错误: 空值
```

```
Test1.java x
1 public class Test1 {
2     public static void main(String[] args) {
3         Integer i=null;
4         switch(i){
5             case 1->System.out.println("1");
6             //case null->System.out.println("错误: 空值");
7             default -> System.out.println("0");
8         }
9     }
10 }
```

运行 Test1 x

```
D:\Applications\Develop\Kits\jdk2101\bin\java.exe "-javaager
Exception in thread "main" java.lang.NullPointerException Cre
at Test1.main(Test1.java:4)
```

## 3.3 条件表达式

### ◆语法

*bool-expression* ? *expression1* : *expression 2*

当bool-expression为真时，表达式的结果为expression1，否则结果为expression2

例如：求num1和num2的最大值

```
max = (num1 > num2) ? num1 : num2;
```

## 3.4 操作符的优先级和结合规则

- ◆ 括号优先级最高，如果括号有嵌套，内部括号优先执行。
- ◆ 如果没有括号，则根据操作符的优先级和结合规则确定执行顺序。
- ◆ 如果相邻的操作符有相同的优先级，则根据结合规则确定执行顺序。
  - 除赋值运算符之外的二元运算符都是左结合的。
  - 赋值运算符和?:运算符是右结合的。

例如：

$a+b-c+d$  等价于  $((a+b)-c)+d$

$a=b+=c=5$  等价于  $a=(b+=(c=5))$



## 3.4 操作符的优先级和结合规则

Level	Operator	Description	Associativity
16	[] . ()	access array element access object member parentheses	left to right
15	++ --	unary post-increment unary post-decrement	not associative
14	++ -- + - ! ~	unary pre-increment unary pre-decrement unary plus unary minus unary logical NOT unary bitwise NOT	right to left
13	() new	cast object creation	right to left
12	* / %	multiplicative	left to right
11	+ - +	additive string concatenation	left to right
10	<< >> >>>	shift	left to right

Level	Operator	Description	Associativity
9	< <= > >= instanceof	relational	not associative
8	== !=	equality	left to right
7	&	bitwise AND	left to right
6	^	bitwise XOR	left to right
5		bitwise OR	left to right
4	&&	logical AND	left to right
3		logical OR	left to right
2	?:	ternary	right to left
1	= += -= *= /= %= &= ^=  = <<= >>= >>>=	assignment	right to left

## 3.4 操作符的优先级和结合规则

### 操作数的运算次序

- ◆操作符的优先级和结合规则只规定了操作符的执行顺序。操作数从左至右进行运算。
- ◆二元操作符左边的操作数比右边的操作数优先运算。例如：

```
int a = 0;  
int x = a + (++a);  
x的结果为1
```

```
int a = 0;  
int x = (++a) + a;  
x的结果为2
```

## 3.4 操作符的优先级和结合规则

### %运算符

- ◆Java允许对浮点数（如double或float）使用%运算符，C++的%运算符仅支持整数类型，浮点数取余需使用fmod()等函数。
- ◆Java中余数的符号与被除数一致，C++中余数的符号由实现定义，通常与除数一致。

```
public static void main(String[] args) {  
    System.out.println("5%3="+5%3);           //5%3=2  
    System.out.println("5.7%3="+5.7%3);       //5.7%3=2.7  
    System.out.println("5%3.3="+5%3.3);       //5%3.3=1.7  
    System.out.println("5.7%3.3="+5.7%3.3);   //5.7%3.3=2.4  
    System.out.println("-5%3="+-5%3);         //-5%3=-2  
    System.out.println("5%-3="+5%-3);        //5%-3=2  
    System.out.println("-5%-3="+-5%-3);      //-5%-3=-2  
}
```