# Java程序的基本语法

多一句没有，少一句不行，用更短的时间，教会更实用的技术

# 功能的最小单元



**AI Prompt** 用AI完成几个小需求，快速熟悉一下程序的基本语法知识

```java
// 根据天气温度区间输出穿什么衣服
public static void getClothes(int temp) {
    if (temp < 0) {
        System.out.println("穿大衣");
    } else if (temp >= 0 && temp <
        System.out.println("穿毛衣");
    } else if (temp >= 10 && temp <
        System.out.println("穿T恤");
    } else if (temp >= 20 && temp < 30) {
        System.out.println("穿短裤");
    } else{
        System.out.println("穿短裙");
    }
}
```

方法

```java
// 计算任意两个整数的和
public static int sum(int a, int b) {
    return a + b;
}
```

方法

```java
// 打印10行Hello World
public static void printHelloWorld() {
    for (int i = 0; i < 10; i++) {
        System.out.println("Hello World");
    }
}
```

方法

```java
public static void main(String[] args) {
    printHelloWorld();
    getClothes( temp: 26);
}
```

方法

**Java开发的软件，功能的最小单位是一个一个的方法**

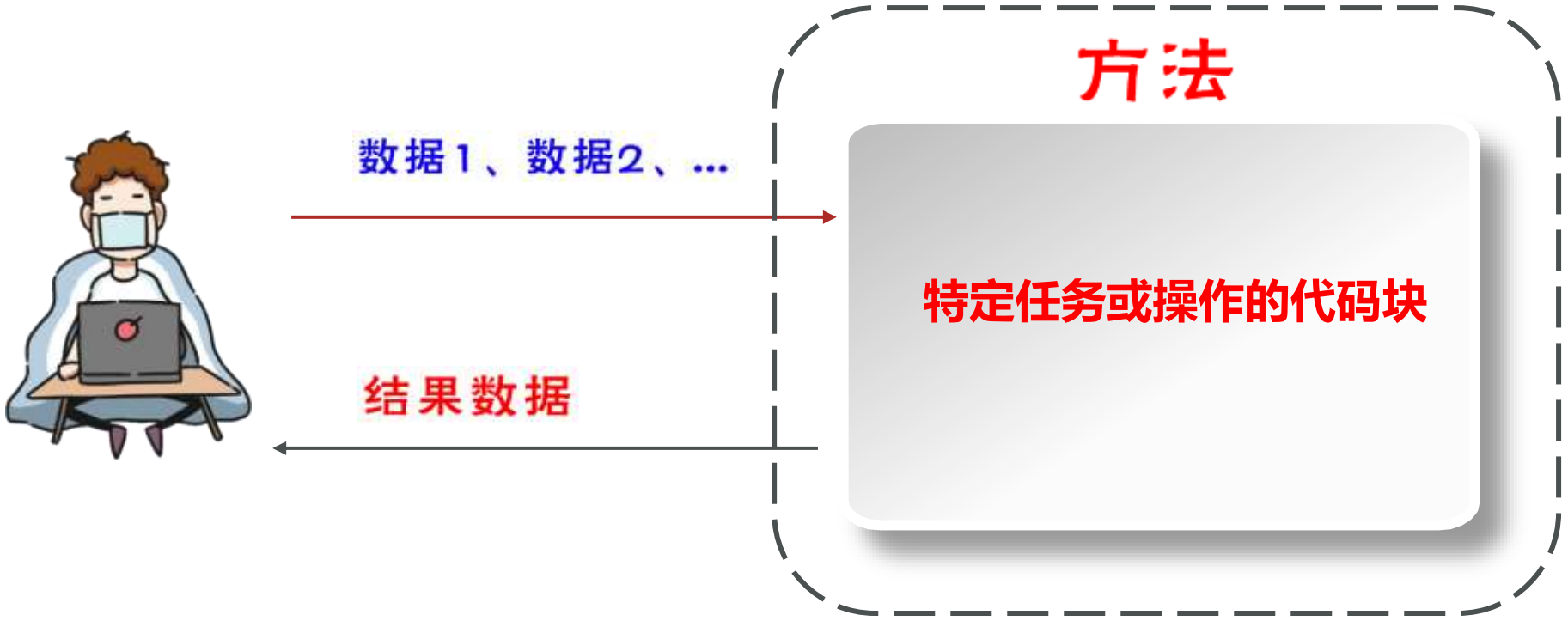
CONTENT

# Java基础语法

- **方法**
  - 方法详解
  - 方法的其他注意事项
- **类型转换**
- **输入输出**
- **运算符**
- **综合小案例**

# 方法是啥?

- 方法是一种用于执行特定任务或操作的代码块，代表一个功能，它可以接收数据进行处理，并返回一个处理后的结果。




**目标：学会定义满足需求的方法格式，学会调用方法**

# 方法是啥?



- 方法是一种用于执行特定任务或操作的代码块，代表一个功能，它可以接收数据进行处理，并返回一个处理后的结果。

## 方法的完整定义格式

求任意两个整数的较大值

```
修饰符  返回值类型  方法名( 形参列表 ){
        方法体代码(需要执行的功能代码)
        return 返回值;
}
```

```
方法的修饰符    返回值类型  方法名称  形参列表
public static   int       max ( int a , int b ){
    int max = a > b ? a : b;
    return max;
}
```

方法的执行代码(处理数据)

返回处理后的结果

## 方法如何使用

- 方法必须被调用才能执行，调用格式：方法名称(数据)。

# 方法的其他形式



1、方法是否需要接收数据处理?

2、方法是否需要返回数据?

```
// 打印3行Hello World（使用方法）：无参数无返回值的方法
public static void print(){
        System.out.println("Hello World");
        System.out.println("Hello World");
        System.out.println("Hello World");
}
```

- 如果方法不需要返回数据，返回值类型必须声明成void（无返回值声明）方法内部不可以使用return返回数据。

BRIEF

# 小结

1、什么是方法？方法的完整格式是啥？

- **是用于执行特定任务或操作的代码块，可以接收数据进行处理并返回一个处理的结果。**

```
修饰符  返回值类型  方法名( 形参列表 ){
        方法体代码(需要执行的功能代码)
        return 返回值;
}
```

2、定义满足需求的方法格式，主要考虑哪两方面？

- **方法是否需要接收数据，方法是否需要返回数据。**

3、方法如何使用？

- **必须进行调用才可以执行方法；调用格式：方法名称(...)。**

CONTENT

# Java基础语法

- 方法
  - 方法详解
  - 方法的其他注意事项
- 类型转换
- 输入输出
- 运算符
- 综合小案例

# 方法的一些注意事项



## 1、方法可以重载。

- 一个类中，出现多个方法的名称相同，但是它们的形参列表是不同的，那么这些方法就称为方法重载了。

```
public static void printVariable(int a) {
    System.out.println(a);
}

public static void printVariable(String str) {
    System.out.println(str);
}

public static void printVariable(int a, String str) {
    System.out.println(a);
    System.out.println(str);
}
```

## 2、无返回值的方法中可以直接通过单独的 return; 立即结束当前方法的执行。

CONTENT

# Java基础语法

- 方法
- **类型转换**
  - 自动类型转换、强制类型转换
  - 表达式的自动类型提升
- 输入输出
- 运算符
- 综合小案例

# 类型转换-自动类型转换



## 什么是自动类型转换，为什么要进行自动类型转换?

- **类型范围小**的变量，可以**直接赋值**给**类型范围大**的变量。

byte　　int

```
byte ──→ short ──→ int ──→ long ──→ float ──→ double
                    ↑
                  char
```

```
byte a = 12 ;
```

```
printInt(a)
```

```
public static void printInt(int b){
    System.out.println(b);
}
```

## 自动类型转换在计算机中的执行原理

a　00001100　(8位)

b　00000000 00000000 00000000 00001100　(32位)

# 类型转换-强制类型转换



## 什么是强制类型转换，为什么要进行强制类型转换?

- **类型范围大**的变量， 不可以**直接赋值**给**类型范围小**的变量，会报错，需要强制类型转换过去

```
数据类型 变量2 = (数据类型)变量1、数据
```

```
int a = 20;
byte b =  (byte)a;
System.out.println(b); // 20
```

a 00000000 00000000 00000000 00010100
(32位)

b 00010100
(8位)

```
int i = 1500;
byte j =  (byte)i;
System.out.println(j); // -36
```

i 00000000 00000000 00000101 11011100
(32位)

j 11011100
(8位)

## 注意事项

- 强制类型转换**可能**造成数据(丢失)溢出；浮点型强转成整型，直接丢掉小数部分，保留整数部分返回

BRIEF

# 小结

为什么要进行类型转换?

- 存在不同类型的变量赋值给其他类型的变量

什么是自动类型转换?

- **类型范围小**的变量，可以**直接赋值**给**类型范围大**的变量。

什么是强制类型转换?

- 默认情况下，大范围类型的变量直接赋值给小范围类型的变量会报错!
- 可以强行将类型范围大的变量、数据赋值给类型范围小的变量

```
数据类型 变量 = (数据类型)变量、数据
```

强制类型转换有哪些需要注意的?

- 可能出现数据丢失。
- 小数强制转换成整数是直接截断小数保留整数。

CONTENT

# Java基础语法

- 方法
- **类型转换**
  - 自动类型转换、强制类型转换
  - 表达式的自动类型提升
- 输入输出
- 运算符
- 综合小案例

# 表达式的自动类型提升



## 表达式的自动类型转换

- 在表达式中，小范围类型的变量，会自动转换成表达式中较大范围的类型，再参与运算。

```
byte 、short、char ——→ int ——→ long ——→ float ——→ double
```

## 注意事项：

- 表达式的最终结果类型由表达式中的最高类型决定。
- 在表达式中，byte、short、char 是直接转换成int类型参与运算的。

BRIEF

# 小结

表达式的自动类型转换是什么样的?

- 小范围的类型会自动转换成大范围的类型运算。

表达式的最终结果类型是由谁决定的?

- 最终类型由表达式中的最高类型决定。

表达式的有哪些类型转换是需要注意的?

- byte short char是直接转换成int类型参与运算的。

CONTENT

# Java基础语法

- 方法
- 类型转换
- **输入输出**
- 运算符
- 综合小案例

# 输入输出

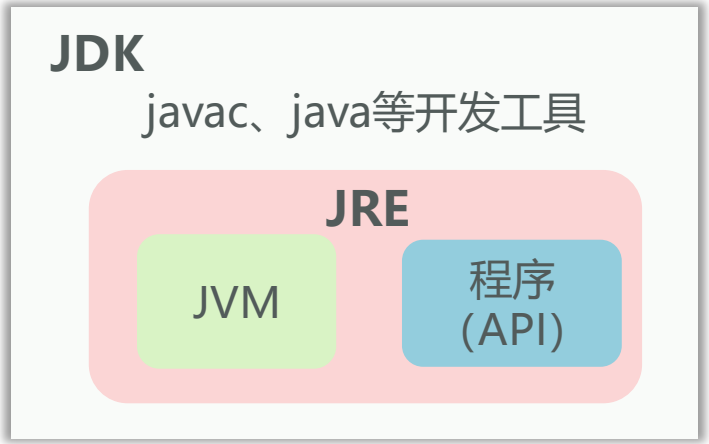


- 输出：把程序中的数据展示出来。

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

- 输入：程序读取用户键盘输入的数据。 通过Java提供的 Scanner程序来实现

**Scanner是Java提供好的API，程序员可以直接调用**




**API （Application Programming Interface：应用程序编程接口）**

- Java写好的程序，咱们程序员可以直接拿来调用。
- Java为自己写好的程序提供了相应的 程序使用说明书(API文档)。

**下载API文档：**

https://www.oracle.com/java/technologies/javase-jdk21-doc-downloads.html

# 使用Scanner接收用户键盘输入的数据，需要三个步骤：



①：导包：告诉程序去JDK的哪个包中找扫描器技术

②：抄代码：代表得到键盘扫描器对象(东西)。

③：抄代码：等待接收用户输入数据。

```
package com.itheima.scanner;
import java.util.Scanner;
public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println(“请输入您的年龄：”);
        int age = sc.nextInt();
        System.out.println(“年龄是：” + age);

        System.out.println(“请输入您的名称：”);
        String name = sc.next();
        System.out.println("欢迎：" + name);
    }
}
```

**注意：**

- **System、String在JDK中的Java.lang包下**
- **lang包不需要我们导包，是默认的包。**

CONTENT

# Java基础语法

- 方法
- 类型转换
- 输入输出
- 运算符
  - 算数运算符、+符号做连接符
  - 自增自减运算符
  - 赋值运算符
  - 关系运算符、三元运算符
  - 逻辑运算符
- 综合小案例

# 基本的算术运算符



| 符号 | 作用 | 说明 |
| --- | --- | --- |
| + | 加 | 参考小学一年级 |
| - | 减 | 参考小学一年级 |
| * | 乘 | 参考小学二年级，与“×”相同 |
| / | 除 | 与“÷”相同，注意：在Java中两个整数相除结果还是整数。 |
| % | 取余 | 获取的是两个数据做除法的余数 |

# +符号在Java中的特殊用途



- “+” 符号在有些情况下可以做连接符。
- “+”符号与字符串运算的时候是用作连接符的，其结果依然是一个字符串。

```
"abc" +  5  ---> "abc5"
```


```
int a = 5 ;
System.out.println("abc" + a);
System.out.println(a + 5);
System.out.println("itheima"+ a + 'a');
System.out.println(a + 'a' + "itheima");
```

**如何识别+符号是做运算，还是做连接，独门秘籍：**

- **能算则算，不能算就连接在一起。**

BRIEF

# 小结

算数运算符有哪些?

- + 、-、 *、 /、 %

/ 需要注意什么，为什么?

- 如果两个整数做除法，其结果一定是整数，因为最高类型是整数。

+ 除了做基本数学运算，还有哪些功能?

- 与字符串做+运算时会被当成连接符，其结果还是字符串。
- 识别技巧：能算则算，不能算就在一起。

CONTENT

# Java基础语法

- 方法
- 类型转换
- 输入输出
- 运算符
  - 算数运算符、 +符号做连接符
  - 自增自减运算符
  - 赋值运算符
  - 关系运算符、三元运算符
  - 逻辑运算符
- 综合小案例

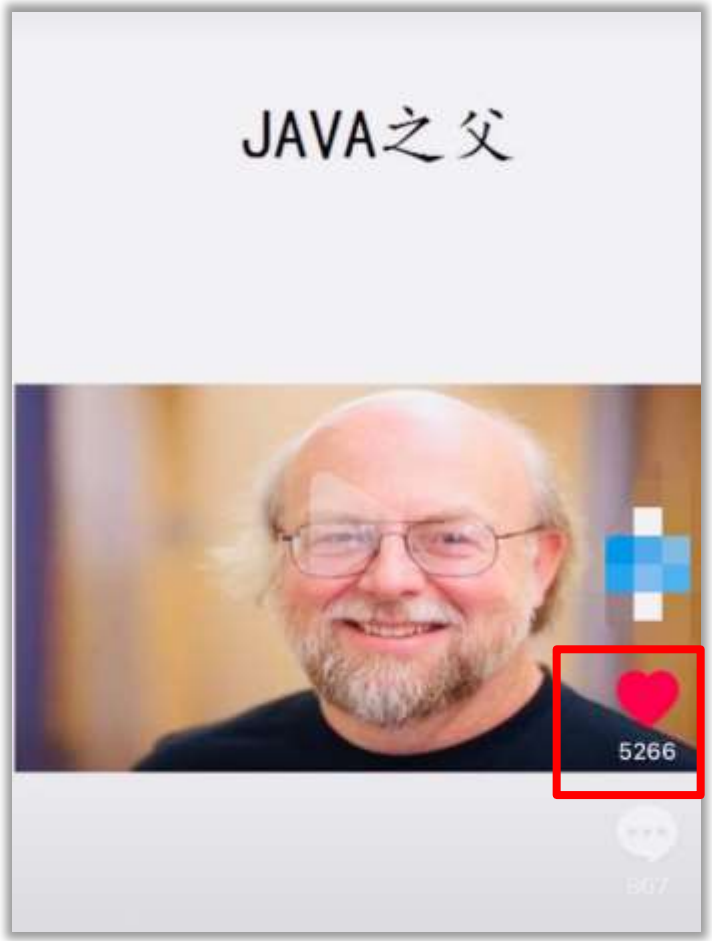
# 自增自减运算符



| 符号 | 作用 |
| --- | --- |
| 自增：++ | 放在某个变量前面或者后面，对变量自身的值加1 |
| 自减：-- | 放在某个变量前面或者后面，对变量自身的值减1 |

**注意：**

- ++ 、-- 只能操作变量，不能操作字面量的。



# 自增自减运算符



- 自增自减的使用注意事项
- ++、-- 如果在变量前后单独使用是没有区别的。

```
int a = 10;
++a;
a++;
```

- ++、-- 如果不是单独使用（如在表达式中、或同时有其它操作），放在变量前后会存在明显区别
  - 在变量的前面，先对变量+1、-1，再拿变量的值运算。

```
int a = 10;
int rs = ++a;  （先加再用）
```

  - 在变量的后面，先拿变量的值运算，再对变量的值+1、-1

```
int b = 10;
int rs = b++;  （先用再加）
```

BRIEF

# 小结

自增、自减运算符是什么，有什么作用，需要注意什么？

- ++、--；对当前变量值+1、-1
- 只能操作变量，不能操作字面量

自增、自减运算符放在变量前后有区别吗？

- 如果单独使用放前放后是没有区别的。
- 非单独使用：在变量前 ，先进行变量自增/自减，再使用变量。
- 非单独使用：在变量后 ，先使用变量，再进行变量自增/自减。

案例

# 自增自减拓展案例



```
int c = 10;

int d = 5;

int rs3 = c++ + ++c - --d - ++d + 1 + c--;

System.out.println(rs3);

System.out.println(c);

System.out.println(d);
```

CONTENT

# Java基础语法

- 方法
- 类型转换
- 输入输出
- 运算符
  - 算数运算符、+符号做连接符
  - 自增自减运算符
  - 赋值运算符
  - 关系运算符、三元运算符
  - 逻辑运算符
- 综合小案例

# 赋值运算符



## 基本赋值运算符

- 就是“=”，从右边往左看。

```
int a = 10;  // 先看“=”右边，把数据10赋值给左边的变量a存储。
```

## 扩展赋值运算符

| 符号 | 用法 | 作用 | 底层代码形式 |
| --- | --- | --- | --- |
| += | a+=b | 加后赋值 | a = (a的类型)(a + b); |
| -= | a-=b | 减后赋值 | a = (a的类型)(a - b); |
| *= | a*=b | 乘后赋值 | a = (a的类型)(a * b); |
| /= | a/=b | 除后赋值 | a = (a的类型)(a / b); |
| %= | a%=b | 取余后赋值 | a = (a的类型)(a % b); |

**注意：扩展的赋值运算符隐含了强制类型转换。**

BRIEF

# 小结

赋值运算符有哪些?

- 基本的赋值运算符：= (从右边往左看)
- 扩展的赋值运算符：+=、-=、*=、/=、%=

扩展赋值运算符的作用是什么？有什么特点

- +=可以实现数据的累加，把别人的数据加给自己。
- 扩展的赋值运算符自带强制类型转换。

CONTENT

# Java基础语法

- 方法
- 类型转换
- 输入输出
- 运算符
  - 算数运算符、+符号做连接符
  - 自增自减运算符
  - 赋值运算符
  - 关系运算符、三元运算符
  - 逻辑运算符
- 综合小案例

# 关系运算符、三元运算符



| 符号 | 例子 | 作用 | 结果 |
| --- | --- | --- | --- |
| > | a>b | 判断a是否大于b | 成立返回true、不成立返回false |
| >= | a>=b | 判断a是否大于或者等于b | 成立返回true、不成立返回false |
| < | a<b | 判断a是否小于b | 成立返回true、不成立返回false |
| <= | a<=b | 判断a是否小于或者等于b | 成立返回true、不成立返回false |
| == | a==b | 判断a是否等于b | 成立返回true、不成立返回false |
| != | a != b | 判断a是否不等于b | 成立返回true、不成立返回false |

- 判断数据是否满足条件，最终会返回一个判断的结果，这个结果是布尔类型的值：true或者false。

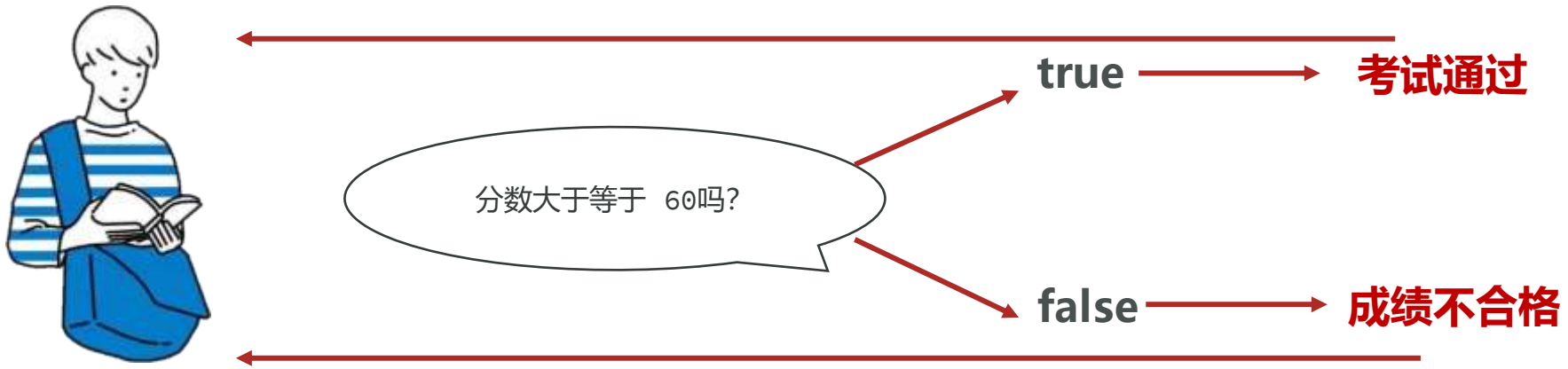
**注意：在 Java 中判断是否相等一定是“==”，千万不要把“==”误写成“=”。**

# 三元运算符



- 格式：`条件表达式？值1：值2;`
- 执行流程：首先计算**关系表达式的值**，如果值为**true**，返回**值1**，如果为**false**，返回**值2**。

```
90  成绩及格！
25  成绩不合格！
```



CONTENT

# Java基础语法

- 方法
- 类型转换
- 输入输出
- 运算符
  - 算数运算符、+符号做连接符
  - 自增自减运算符
  - 赋值运算符
  - 关系运算符、三元运算符
  - 逻辑运算符
- 综合小案例

# 逻辑运算符



- 把多个条件放在一起运算，最终返回布尔类型的值：true、false。

| 符号 | 叫法 | 例子 | 运算逻辑 |
| --- | --- | --- | --- |
| & | 逻辑与 | 2 > 1 & 3 > 2 | 多个条件必须都是true, 结果才是true； 有一个是false，结果就是false |
| \| | 逻辑或 | 2 > 1 \| 3 < 5 | 多个条件中只要有一个是true, 结果就是true; |
| ! | 逻辑非 | ! (2 > 1) | 就是取反：你真我假，你假我真。!true == false、!false == true |
| ^ | 逻辑异或 | 2 > 1 ^ 3 > 1 | 前后条件的结果相同，就直接返回false，前后条件的结果不同，才返回true |

| 符号 | 叫法 | 例子 | 运算逻辑 |
| --- | --- | --- | --- |
| && | 短路与 | 2 > 10 && 3 > 2 | 判断结果与“&”一样，过程不同：**左边为 false，右边则不执行。** |
| \|\| | 短路或 | 2 > 1 \|\| 3 < 5 | 判断结果与“\|”一样，过程不同：**左边为 true， 右边则不执行。** |

注意：在Java中，“&”、“|”：无论左边是 false 还是 true，右边都要执行。

由于&&、||运算效率更高、在开发中用的更！

BRIEF

# 小结

逻辑运算符有哪些，有什么特点?

- &： 有一个为false、结果是false
- &&： 一个为false、结果是false，但前一个为false,后一个条件不执行了
- |： 有一个为true、结果是true
- ||：一个为true、结果是true，但前一个为true，后一个条件不执行了
- !：!false=true、 !true=false
- ^ ：相同是false、不同是true。

- 注意：实际开发中、常用的逻辑运算符还是：&& 、 || 、 ！

CONTENT

# Java基础语法

- 方法
- 类型转换
- 输入输出
- 运算符
- 综合小案例

# 案例 健康计算器



- 开发一个简单的健康计算器应用程序，它可以接受用户的输入（如年龄、性别、体重、身高），并计算出用户的BMI（身体质量指数）和基础代谢率（BMR）。

BMI（Body Mass Index，身体质量指数）是用来评估体重是否适宜的一个常用指标。它通过体重（以千克为单位）除以身高（以米为单位）的平方来计算。

BMI的正常范围通常定义为18.5到24.9（$kg/m^2$）。具体来说：

- BMI小于18.5被认为是体重过低；
- BMI在18.5到24.9之间被认为是正常范围；
- BMI在25到29.9之间被认为是超重；
- BMI在30及以上被认为是肥胖。

BMR的计算可以使用不同的公式，其中Harris-Benedict公式是比较常见的：

- 对于男性：BMR = 88.362 + (13.397 × 体重[kg]) + (4.799 × 身高[cm]) - (5.677 × 年龄[岁])
- 对于女性：BMR = 447.593 + (9.247 × 体重[kg]) + (3.098 × 身高[cm]) - (4.330 × 年龄[岁])

基础代谢率（BMR）的具体数值因人而异，主要受年龄、性别、体重、身高、体脂比例和肌肉量等因素影响。以下是根据不同公式计算的成年男女平均BMR的大概范围：

- **男性**：大约在1500至2500卡路里/天之间。
- **女性**：大约在1200至2000卡路里/天之间。