# 方法

## 概述

Java语言中的“方法”（Method）在其他语言当中也可能被称为“函数”（Function）。对于一些复杂的代码逻辑，如果希望重复使用这些代码，并且做到“随时任意使用”，那么就可以将这些代码放在一个大括号“{}”当中，并且起一个名字。使用代码的时候，直接找到名字调用即可。

## 定义格式

首先了解关于方法的两个概念：

1. **参数**：是指进入方法中的数据，有了这些数据，方法才能执行逻辑。
2. **返回值**：是指从方法中出来的数据，也就是方法执行之后的最终结果数据。

目前定义方法的基本格式：

**修饰符 返回值类型 方法名称(参数类型 参数名称) {**

**方法体**

**return 返回值;**

**}**

对于定义格式的解释：

* 修饰符：现阶段固定为public static两个关键字。
* 返回值类型：方法最终产生的结果数据是什么类型。
* 方法名称：自定义的名称，命名规则和变量一样。
* 参数类型：进入方法的数据是什么类型。
* 参数名称：进入方法的数据对应的变量名称。
* 方法体：方法内部执行的若干行代码。
* return：结束方法的执行，并且将返回值返还给调用处。
* 返回值：方法最终产生的结果数据。

注意：

1. 返回值必须和返回值类型对应。
2. 参数如果有多个，需要使用逗号分隔。
3. 参数如果没有，小括号则可以留空。

## 定义代码

**public** **class** Demo02MethodDefine {

/\*

\* 定义方法的三要素

\* 返回值类型：最终结果肯定是一个int类型

\* 方法名称：sum

\* 参数列表：两个int数值

\*/

**public** **static** **int** sum(**int** a, **int** b) {

System.***out***.println("方法执行啦！");

**int** result = a + b; // 方法体

**return** result;

}

**public** **static** **void** main(String[] args) {

}

}

注意事项：

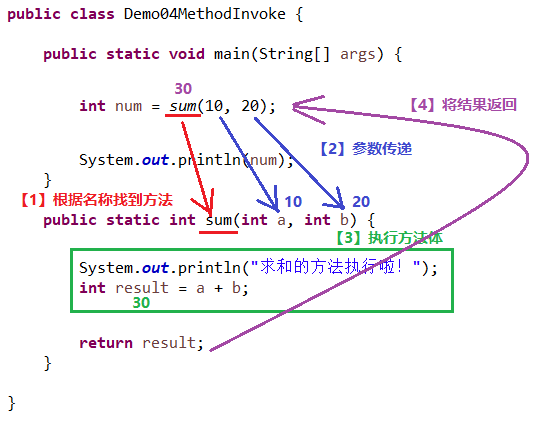
1. 多个方法的定义先后顺序无所谓。
2. 不能在一个方法内部定义方法。
3. 方法定义之后，没有调用就不会执行；要想执行，一定要调用它。

## 调用方式

调用方法的方式有常见的三种：

1. 单独调用。这种方式无法使用方法的返回值。格式：  
   **方法名称(参数值);**
2. 打印调用。这种方式可以将方法的返回值直接打印。格式：  
   **System.out.println(方法名称(参数值));**
3. 赋值调用。这种方式可以将方法的返回值赋值给一个变量，注意变量的数据类型必须和方法的返回值类型对应。格式：  
   **数据类型 变量名称 = 方法名称(参数值);**

## 调用流程

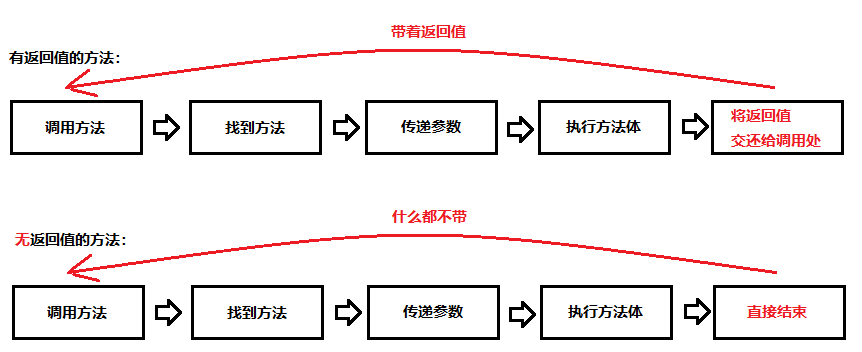


## 重名问题

1. 变量的名称是否可以与方法名称重名？可以。
2. 两个不同的方法中，能否各自有一个重名的变量？可以。

# 无返回值的方法

## 场景举例



1. 概念：如果方法只是执行方法体操作，并没有最终结果必须交还给调用处，那么这就是一个无返回值的方法。
2. 有返回值：大哥叫小弟去杀人，杀了之后提人头来见，人头就是返回值。
3. 无返回值：大哥叫小弟去杀人，杀了之后回报一声即可，不要人头，这就没有返回值。

## 定义格式

**修饰符 void 方法名称(参数类型 参数名称) {**

**方法体**

**return;**

**}**

修饰符：与普通方法一样

返回值类型：**固定为void**

方法名称：与普通方法一样

参数类型：与普通方法一样

参数名称：与普通方法一样

方法体：与普通方法一样

return：后面**不能写返回值**，直接分号结束。而且**最后一行return**通常情况下会**省略不写**。

## 代码使用

**public** **class** Demo01MethodVoid {

**public** **static** **void** main(String[] args) {

// 单独调用

*printHelloWorld*();

System.***out***.println("===============");

// 单独调用

*printHelloWorldCount*(15);

System.***out***.println("===============");

// 错误写法！void的方法不能使用打印调用

// System.out.println(printHelloWorld());

// System.out.println(void);

// 错误写法！void的方法不能使用赋值调用

// int num = printHelloWorld();

// int num = void;

}

/\*

\* 定义一个方法，用来打印输出固定10次HelloWorld。

\* 三要素

\* 返回值类型：只是希望进行打印输出，屏幕显示而已，并没有最终的数据结果产生，所以使用void

\* 方法名称：printHelloWorld

\* 参数列表：不需要任何条件数据，就可以完成任务。所以参数留空。

\*/

**public** **static** **void** printHelloWorld() {

**for** (**int** i = 0; i < 10; i++) {

System.***out***.println("Hello, World! " + (i + 1));

}

// return; // 最后一行return语句可以省略

// return 100; // 错误写法！void就不能有具体的返回值

}

/\*

\* 定义一个方法，用来打印输出指定次数的HelloWorld。

\* 三要素

\* 返回值类型：仍然只是进行打印输出，仍然没有最终的数据结果产生，所以还是使用void

\* 方法名称：printHelloWorldCount

\* 参数列表：需要知道到底是几次，才能打印。需要一个int数字代表次数。

\*/

**public** **static** **void** printHelloWorldCount(**int** count) {

**for** (**int** i = 0; i < count; i++) {

System.***out***.println("Hello, World! " + (i + 1));

}

}

}

## 遍历数组

遍历数组，是指对数组当中的每一个数据元素进行逐一、挨个儿处理。默认的遍历就是打印输出。

**public** **class** Demo02PrintArray {

**public** **static** **void** main(String[] args) {

// 1. 首先准备一个数组

**int**[] array = { 10, 20, 30, 40, 50, 65, 75 };

// 3. 调用方法，传入参数数据（一个数组）

*printArray*(array);

}

/\*

\* 2. 定义一个方法，目的是遍历数组

\* 三要素

\* 返回值类型：只是进行屏幕的打印显示而已，没有最终数据结果要还给调用处，所以用void

\* 方法名称：printArray

\* 参数列表：一定要得到一个数组，才能打印。int[]

\*/

**public** **static** **void** printArray(**int**[] array) {

**for** (**int** i = 0; i < array.length; i++) {

System.***out***.println(array[i]);

}

}

}

扩展成为指定格式：

**public** **class** Demo03PrintArrayFormat {

**public** **static** **void** main(String[] args) {

**int**[] array1 = { 10, 20, 30, 40, 50 };

*printArrayFormat*(array1);

**int**[] array2 = { 5, 15, 25, 35, 45 };

*printArrayFormat*(array2);

}

**public** **static** **void** printArrayFormat(**int**[] array) {

System.***out***.print("["); // 1. 首先打印左括号

**for** (**int** i = 0; i < array.length; i++) {

// 2. 判断一下当前元素是不是最后一个？

**if** (i == array.length - 1) { // 如果是最后一个

System.***out***.println(array[i] + "]");

} **else** { // 如果不是最后一个

System.***out***.print(array[i] + ", ");

}

}

}

}

## 对比返回值的有无

**public** **class** Demo04VoidVsReturn {

**public** **static** **void** main(String[] args) {

**int**[] array = { 1, 2, 7, 12, 80, -14 };

// 有返回值的调用

**int** result = *sumA*(array);

System.***out***.println("结果是：" + result);

// 无返回值的调用

*sumB*(array);

}

// 求出数组当中元素的总和，有返回值

**public** **static** **int** sumA(**int**[] array) {

**int** sum = 0; // 存钱罐，这个变量用来进行累加

**for** (**int** i = 0; i < array.length; i++) {

sum += array[i];

}

**return** sum;

}

// 打印数组当中元素的总和，只是打印，就没有返回值

**public** **static** **void** sumB(**int**[] array) {

**int** sum = 0;

**for** (**int** i = 0; i < array.length; i++) {

sum += array[i];

}

System.***out***.println("总和：" + sum);

}

}

# 方法重载

## 概述

对于参数列表不一样的多个方法，如果设置不一样的方法名称会很麻烦，所以引入了方法重载（Overload）的技术。

方法重载：方法名称相同，但是参数列表不同。

参数列表的不同包括：

1. 参数的个数不同；
2. 参数的类型不同；
3. 参数的多类型顺序不同。

重载与下列因素无关：

1. 方法的返回值；
2. 参数的名称。

## 基本使用

**public** **class** Demo01MethodOverload {

**public** **static** **void** main(String[] args) {

// System.out.println(sumTwo(10, 20)); // 30

// System.out.println(sumThree(10, 20, 30)); // 60

// System.out.println(sumFour(10, 20, 30, 40)); // 100

System.***out***.println(*sum*(10, 20)); // 30

System.***out***.println(*sum*(10, 20, 30)); // 60

System.***out***.println(*sum*(10, 20, 30, 40)); // 100

// System.out.println(sum(10, 20, 30, 40, 50)); // 错误的写法！因为没有任何一种重载形式能对上

System.***out***.println("===============");

System.***out***.println(*sum*(10, 20)); // int

System.***out***.println(*sum*(10.5, 20.5)); // double

System.***out***.println("===============");

System.***out***.println(*sum*(2.5, 10)); // 先double后int

System.***out***.println(*sum*(10, 2.5)); // 先int后double

}

**public** **static** **double** sum(**int** a, **double** b) {

System.***out***.println("先int后double的方法执行！");

**return** a + b; // 注意！不同数据类型的数值相加，谁的范围大，结果就是谁

}

**public** **static** **double** sum(**double** a, **int** b) {

System.***out***.println("先double后int的方法执行！");

**return** a + b; // 注意！不同数据类型的数值相加，谁的范围大，结果就是谁

}

**public** **static** **double** sum(**double** a, **double** b) {

System.***out***.println("有2个double参数的方法执行！");

**return** a + b;

}

**public** **static** **int** sum(**int** a, **int** b) {

System.***out***.println("有2个int参数的方法执行！");

**return** a + b;

}

// 错误写法！与参数的名称无关

// public static int sum(int x, int y) {

// System.out.println("有2个int参数的方法执行！");

// return x + y;

// }

// 错误写法！重载与返回值无关！

// public static double sum(int a, int b) {

// System.out.println("有2个int参数的方法执行！");

// return a + b + 0.0;

// }

**public** **static** **int** sum(**int** a, **int** b, **int** c) {

System.***out***.println("有3个int参数的方法执行！");

**return** a + b + c;

}

**public** **static** **int** sum(**int** a, **int** b, **int** c, **int** d) {

System.***out***.println("有4个int参数的方法执行！");

**return** a + b + c + d;

}

}

# 参数传递

**形式参数**：在“定义方法”的时候，写在小括号之内的变量，就叫形式参数。

**实际参数**：在“调用方法”的时候，真正传入方法里的数据，叫做实际参数。

两条规则：

1. 对于基本类型（以及String）来说，形式参数的操作【不会】影响实际参数。
2. 对于引用类型（除了String）来说，形式参数的操作【会】影响实际参数。

## 基本类型传参

**public** **class** Demo01BasicParam {

**public** **static** **void** main(String[] args) {

**int** a = 10;

**int** b = 20;

System.***out***.println("a=" + a); // 10

System.***out***.println("b=" + b); // 20

System.***out***.println("======================");

*change*(a, b); // 实际参数

System.***out***.println("======================");

System.***out***.println("a=" + a); // 10

System.***out***.println("b=" + b); // 20

}

**public** **static** **void** change(**int** x, **int** y) { // 形式参数

x \*= 10; // 扩大十倍

y \*= 10; // 扩大十倍

System.***out***.println("x=" + x); // 100

System.***out***.println("y=" + y); // 200

}

}

## 引用类型传参

**public** **class** Demo02RefParam {

**public** **static** **void** main(String[] args) {

**int**[] array = { 10, 20, 30 };

System.***out***.println(array[0]); // 10

System.***out***.println(array[1]); // 20

System.***out***.println(array[2]); // 30

System.***out***.println("====================");

*change*(array);

System.***out***.println("====================");

System.***out***.println(array[0]); // 100

System.***out***.println(array[1]); // 200

System.***out***.println(array[2]); // 300

}

**public** **static** **void** change(**int**[] arr) {

arr[0] \*= 10;

arr[1] \*= 10;

arr[2] \*= 10;

System.***out***.println(arr[0]); // 100

System.***out***.println(arr[1]); // 200

System.***out***.println(arr[2]); // 300

}

}

# 知识总结

1. 方法：就是将一些代码使用大括号包起来，然后取一个名字。使用的时候，只要根据名称调用即可。而且比循环要更加灵活。
2. 如何定义一个方法：  
   修饰符 返回值类型 方法名(参数类型 参数名称) {  
    方法体  
    return 返回值;  
   }
3. 调用方法的三种方式：
   1. 单独调用：方法名称(实际参数);
   2. 打印调用：System.out.println(方法名称(实际参数));
   3. 赋值调用：数据类型 变量名称 = 方法名称(实际参数);
4. 如何定义一个没有返回值的方法：  
   修饰符 void 方法名称(参数类型 参数名称) {  
    方法体  
    // return;  
   }
5. 调用没有返回值的方法，只能使用单独调用，不能使用打印调用或者赋值调用。
6. 重载（Overload）：多个方法的名称相同，但是参数列表不同。
7. 重载中，什么叫做参数不同：
   1. 参数的个数不同
   2. 参数的类型不同
   3. 参数的多类型顺序不同
8. 重载中，与什么无关：
   1. 与返回值类型无关
   2. 与参数名称无关
9. 参数传递：也就是在调用方法的时候，向方法当中传入数据的动作。
10. 参数传递的两项规则：
    1. 对于基本类型（以及String）来说，形式参数的操作【不会】影响实际参数。
    2. 对于引用类型（除了String）来说，形式参数的操作【会】影响实际参数。