

Java基础(9)¹

方法的其他形式

```
1 修饰符 返回值类型 方法名(形参列表){  
2      方法体代码(需要执行的功能代码)  
3      return 返回值;  
4  }
```

设计一个合理的方法的原则如下：

- 如果方法不需要返回数据，返回值类型必须申明成void（无返回值申明），此时方法内部不可以使用return返回数据。
- 方法如果不需要接收外部传递进来的数据，则不需要定义形参，且调用方法时也不可以传数据给方法。
- 没有参数，且没有返回值类型（void）的方法，称为值无参数、无返回值方法。此时调用方法时不能传递数据给方法。

方法使用常见的问题

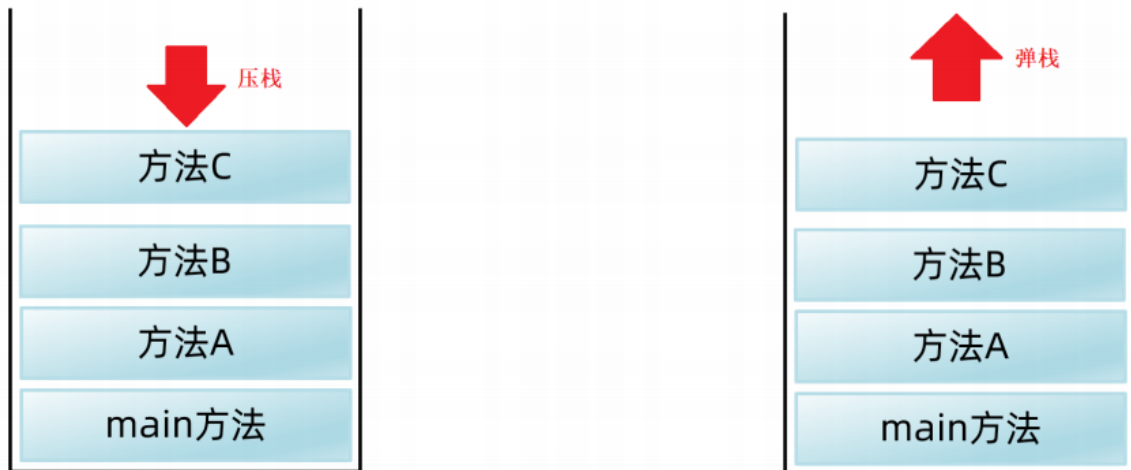
- 方法在类中没有先后顺序，但是不能把一个方法定义在另一个方法中。
- 方法的返回值类型写void（无返回申明）时，方法内不能使用 return 返回数据，如果方法的返回值类型写了具体类型，方法内部必须使用 return 返回对应类型的数据。
- return语句的下面，不能编写代码，属于无效的代码，执行不到这儿。
- 方法不调用就不会执行，调用方法时，传给方法的数据，必须严格匹配方法的参数情况。
- 调用有返回值的方法，有3种方式：
 - ① 可以定义变量接收结果
 - ② 或者直接输出调用，
 - ③ 甚至直接调用；
- 调用无返回值的方法，只有1种方式： 只能直接调用。

方法在计算机中的执行原理

Java的方法是在 **栈内存** 区域中执行。每次调用方法，方法都会进栈执行；执行完后，又会弹栈出去。

假设在main方法中依次调用A方法、B方法、C方法，在内存中的执行流程如下：

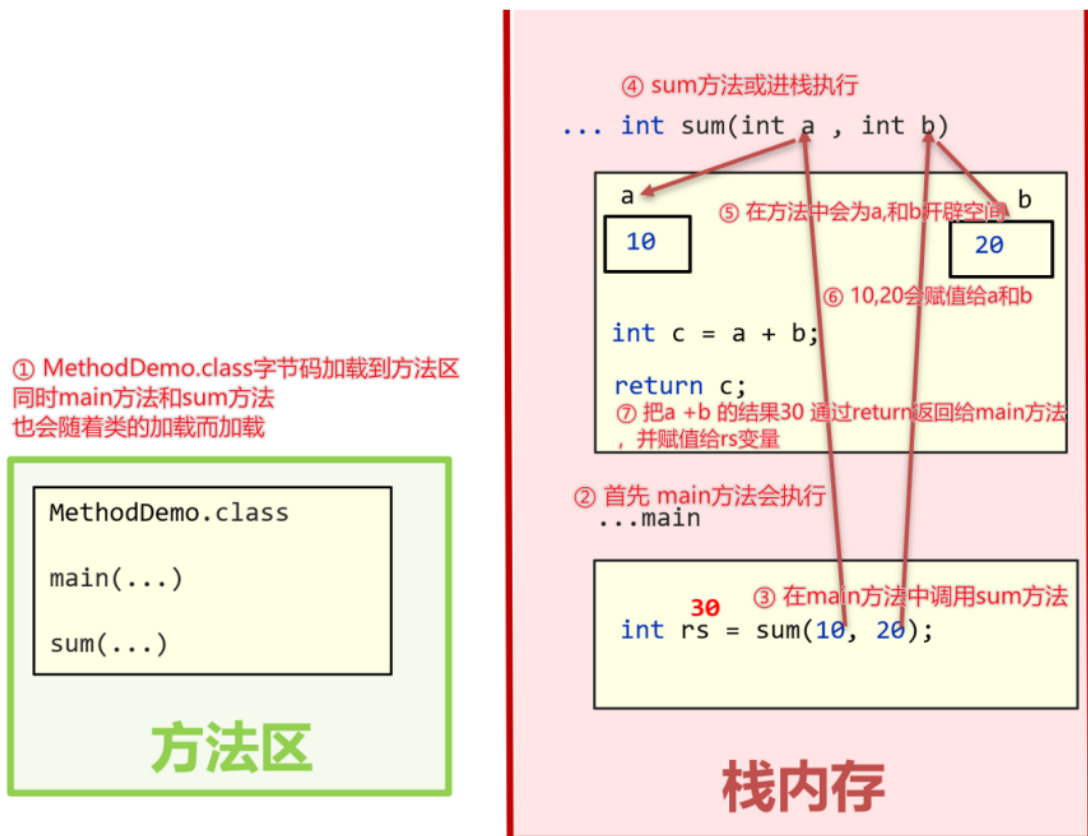
- 每次调用方法，方法都会从栈顶压栈执行没执行
- 每个方法执行完后，会从栈顶弹栈出去



有返回值的方法，内存分析

下面我们分析一下，求两个整数和的代码，在内存中的执行原理。

```
1 public class MethodDemo {
2     public static void main(String[] args) {
3         int rs = sum(10, 20);
4         System.out.println(rs);
5     }
6     public static int sum(int a, int b ){
7         int c = a + b;
8         return c;
9     }
10 }
```



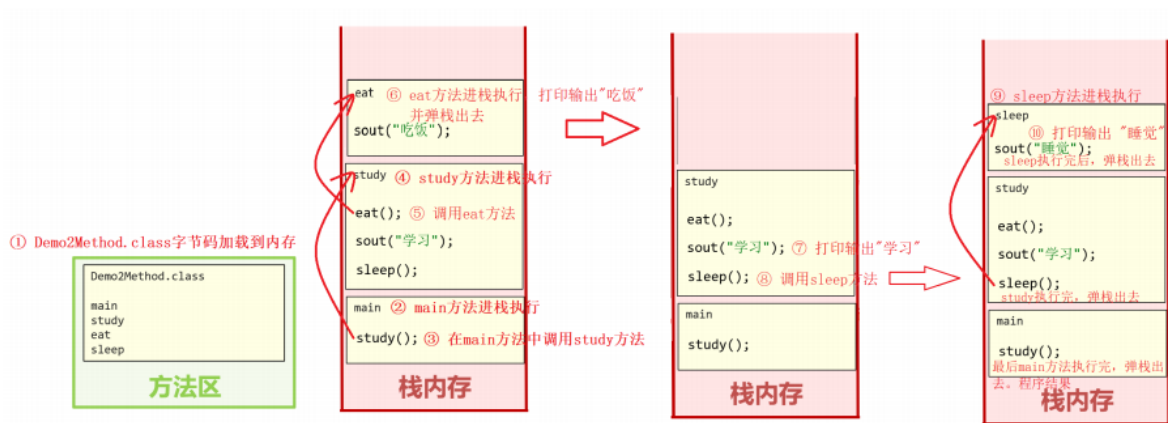
无返回值的方法，内存分析

刚才我们分析的是有参数有返回值的方法内存原理。下面再分析一个无返回值、无参数的内存原理。

```

1 public class Demo2Method {
2     public static void main(String[] args) {
3         study();
4     }
5     public static void study(){
6         eat();
7         System.out.println("学习");
8         sleep();
9     }
10    public static void eat(){
11        System.out.println("吃饭");
12    }
13    public static void sleep(){
14        System.out.println("睡觉");
15    }
16 }

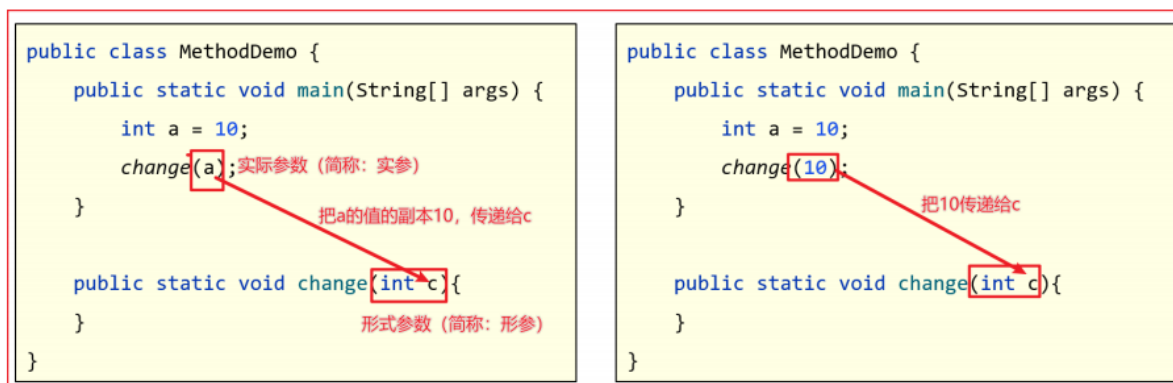
```



方法参数的传递机制

Java的参数传递机制都是：值传递

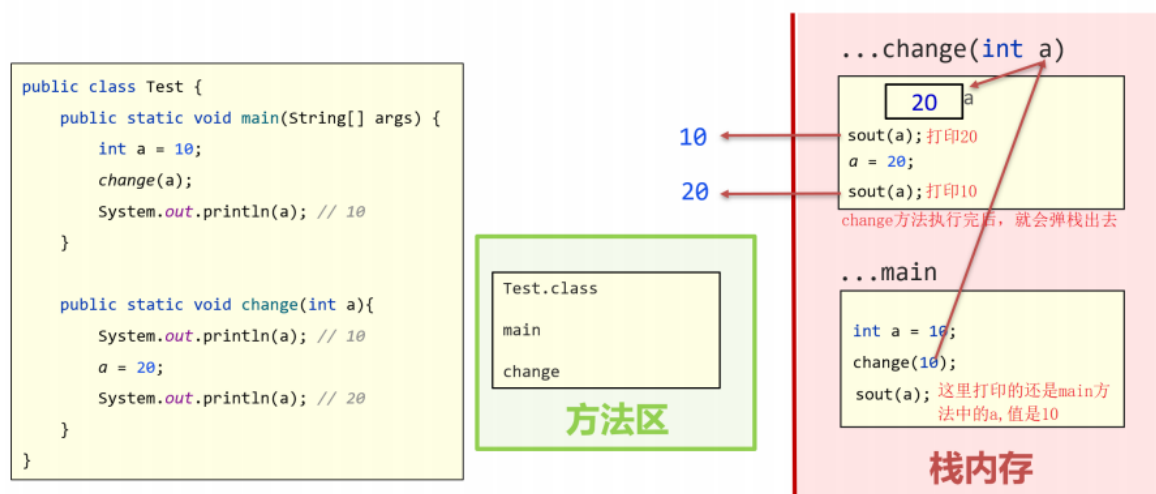
所谓值传递：指的是在传递实参给方法的形参的时候，传递的是实参变量中存储的值的副本。



参数传递的基本类型数据

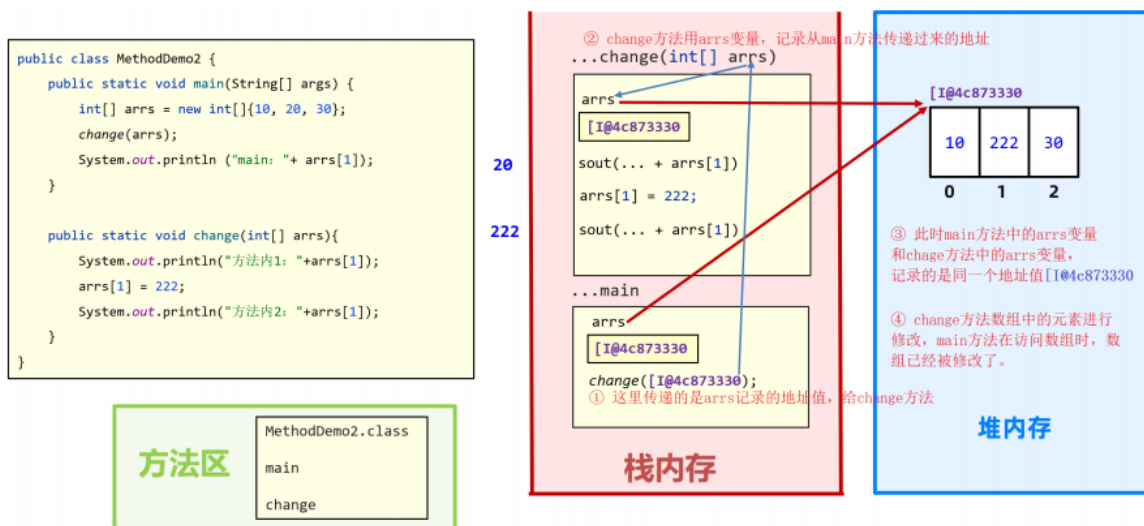
接下来，看一下方法参数传递是基本类型数据时，内存中是怎么执行的。

Java的参数传递机制都是：值传递，传递的是实参存储的值的副本



参数传递的是引用数据类型

接下来，看一下方法的参数是引用类型的数据时，内存中是怎么执行的



调用change方法时参数是引用类型，实际上也是值传递，只不过参数传递存储的地址值

基本类型和引用类型的参数在传递的时候有什么不同？

都是值传递

基本类型的参数传递存储的数据值。

引用类型的参数传递存储的地址值。

可变长度参数

在 Java 中，可变长度参数，也称为可变参数或不定参数，可以在方法声明中使用，用于接受不确定数量的参数。

使用可变长度参数时，需要在参数类型之后加上三个连续的点（...），表示这是一个可变长度参数。在方法内部，可变长度参数会被当作一个数组处理，开发者可以像操作数组一样操作这个参数。需要注意的是，**每个方法只能有一个可变长度参数，而且必须是最后一个参数。**

可变长度参数只能用于方法的最后一个参数位置，并且不能和数组作为参数同时存在，否则会编译错误。在实际开发中，可变长度参数经常用于编写简化代码的 API，以及可以接受任意数量参数的方法。


方法重载

所谓方法重载指的是：一个类中，出现多个相同的方法名，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。

- 参数列表不同的情况

- 长度不一样
- 类型不一样

方法重载需要注意什么

- 一个类中，只要一些方法的名称相同、形参列表不同，那么它们就是方法重载了，
- 其它的都不管（如：修饰符，返回值类型是否一样都无所谓）。
-  ○ 形参列表不同指的是：形参的个数、类型、顺序不同，不关心形参的名称。

方法重载应用场景

开发中我们经常需要为处理一类业务，提供多种解决方案，此时用方法重载来设计是很专业的。

return单独使用

方法中单独使用 `return` 语句，可以用来提前结束方法的执行。

变量的作用域和生命周期

变量的作用域就是指一个变量定义后，在程序的什么地方能够使用。变量的生命周期是指变量什么时候分配内存，什么时候从内存中回收。

前面学过代码块的概念，就是在程序设计的时候，一对大括号 `{ }` 包含的区域。在Java中，使用大括号的地方有：类定义、方法定义、方法中的循环、判断等，一个变量的作用域只被限制在当前变量所在的语句块中（也就包含该变量的，离该变量最近的大括号）。

方法中定义的变量，称为局部变量，方法的形式参数也是方法的局部变量。只能在当前方法中使用，包括当前方法中的判断语句块，循环语句块。在判断语句块中声明的变量只能在当前判断语句块中使用，当前判断语句块之外不能正常使用，对循环语句块也是一样。

变量的生命周期就是从变量声明到变量终结，普通变量的生命周期与作用域范围一致，一个变量在当前语句块结束时，变量被系统回收

递归

在 Java 中，递归是指一个方法或函数在执行过程中调用自身的过程。递归可以用来解决许多问题，特别是那些需要对某种数据结构进行深度优先搜索或遍历的问题。

递归函数通常包含两部分：基本情况和递归情况。基本情况是指问题可以直接解决的情况，通常是递归函数的终止条件，避免无限递归。递归情况是指问题需要进一步分解才能解决的情况，通常是递归函数中调用自身的部分。

在递归函数中，每次递归都会将问题规模减小，直到达到基本情况，然后开始回溯并合并解决方案，直到最终解决整个问题。递归在某些情况下可以让程序更加简洁和优雅，但也可能会带来性能问题和堆栈溢出等问题。

Java 中常见的递归实现包括计算阶乘、斐波那契数列等问题。递归虽然是一种强大的编程技巧，但需要注意递归深度和递归栈的使用情况，以避免出现性能问题和堆栈溢出等情况。

1. Java学习第九天 [↩](#)