

Day5

1、方法定义

完成特定功能的代码块

函数在java中称为方法

方法概述

- 方法概述

- 假设有一个游戏程序，程序在运行过程中，要不断地发射炮弹(植物大战僵尸)。发射炮弹的动作需要编写100行的代码，在每次实现发射炮弹的地方都需要重复地编写这100行代码，这样程序会变得很臃肿，可读性也非常差。为了解决代码重复编写的问题，可以将发射炮弹的代码提取出来放在一个{}中，并为这段代码起个名字，这样在每次发射炮弹的地方通过这个名字来调用发射炮弹的代码就可以了。上述过程中，所提取出来的代码可以被看作是程序中定义的一个方法，程序在需要发射炮弹时调用该方法即可。

格式：

```
修饰符 返回值类型 方法名(参数类型 参数名1, 参数类型 参数名2, 参数类型 参数名3){  
    方法体语句;  
    return 返回值;  
}
```

修饰符：目前就用public static，后面再讲其他的

返回值类型：就是功能结果的类型

方法名：符合命名规则即可

参数：

 实际参数：实际参与运算的

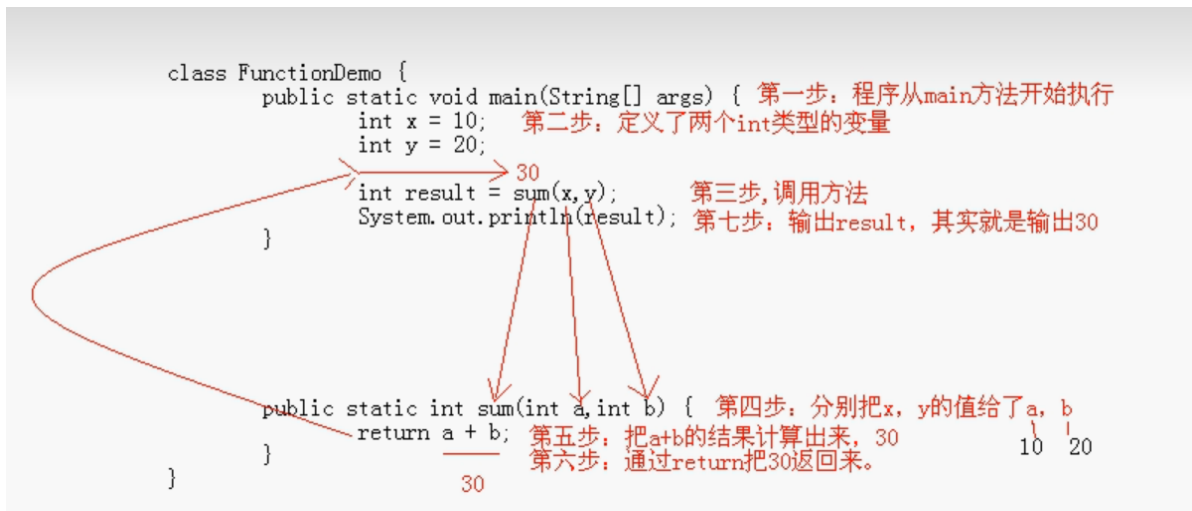
 形式参数：就是方法定义上的，用于接受实际参数的

方法体语句：就是完成功能的语句

return：返回方法的结果

有明确返回值的方法调用

- 有明确返回值的方法调用：
 - 单独调用，没有意义
 - 输出调用，有意义，但是不够好，因为我不一定非要把结果输出
 - 赋值调用，推荐方式
- 讲解完毕该案例后，画图说明方法的调用过程



有明确返回值的方法练习

- 键盘录入两个数据，返回两个数中的较大值
- 键盘录入两个数据，比较两个数是否相等
- 键盘录入三个数据，返回三个数中的最大值

注意事项：

方法不执行不调用
方法与方法是平级关系，不能嵌套定义
方法调用时候不用再传递数据类型
方法如果有明确的返回值，要用return返回

```

class FunctionDemo4 {
    public static void main(String[] args) {
        //for循环输出图形
        for(int x = 0;x < 4;x++){
            for(int y = 0;y < 5;y++){
                System.out.print('*');
            }
            System.out.println();
        }
    }
}

```

2、方法重载

在同一个类中允许定义多个同名的方法

方法名相同，参数列表不同，与返回值无关

可以参数个数不同，也可以参数类型不同

我们的需求不断的发生改变，我们就对应的提供了多个求和的方法。
 但是呢，他们的名字是不一样的。
 而我们又要求方法命名做到：见名知意。
 但是，很明显，现在没有做到。
 那么，肿么办呢？
 针对这种情况：方法的功能相同，参数列表不同的情况，为了见名知意，Java允许它们起一样的名字。
 */

方法重载

● 方法重载概述

- 在同一个类中，允许存在一个以上的同名方法，只要它们的参数个数或者参数类型不同即可。

● 方法重载特点

- 与返回值类型无关，只看方法名和参数列表
- 在调用时，虚拟机通过参数列表的不同来区分同名方法

方法重载案例

方法重载案例

- 比较两个数据是否相等。参数类型分别为两个byte类型，两个short类型，两个int类型，两个long类型，并在main方法中进行测试
- 方法递归在IO之前详细讲解

3、数组概述

数组概述

- 数组概述
 - 需求：现在需要统计某公司员工的工资情况，例如计算平均工资、找到最高工资等。假设该公司有80名员工，用前面所学的知识，程序首先需要声明80个变量来分别记住每位员工的工资，然后在进行操作，这样做会显得很麻烦。为了解决这种问题，Java就提供了数组供我们使用。
 - 那么数组到底是什么呢？有什么特点呢？通过上面的分析：我们可以得到如下两句话：
 - 数组是存储多个变量(元素)的东西(容器)
 - 这多个变量的数据类型要一致

存储同一类型多个数据的集合

数组概念

● 数组概念

- 数组是存储同一种数据类型多个元素的集合。也可以看成是一个容器。
- 数组既可以存储基本数据类型，也可以存储引用数据类型。

● 数组的定义格式

- 格式1：数据类型[] 数组名；
- 格式2：数据类型 数组名[]；
- 注意：这两种定义做完了，数组中是没有元素值的。
如何对数组的元素进行初始化呢？

数组的初始化

● 数组初始化概述：

- Java中的数组必须先初始化,然后才能使用。
- 所谓初始化：就是为数组中的数组元素分配内存空间，并为每个数组元素赋值。

● 数组的初始化方式

- 动态初始化：初始化时只指定数组长度，由系统为数组分配初始值。
- 静态初始化：初始化时指定每个数组元素的初始值，由系统决定数组长度。

初始化方式：静态初始化和动态初始化

数组的初始化

- 动态初始化：初始化时只指定数组长度，由系统为数组分配初始值。
 - 格式：数据类型[] 数组名 = new 数据类型[数组长度];
 - 数组长度其实就是数组中元素的个数。
 - 举例：
 - `int[] arr = new int[3];`
 - 解释：定义了一个 `int` 类型的数组，这个数组中可以存放3个 `int` 类型的值。

```
public class ArrayDemo {  
    public static void main(String[] args) {  
        //      int[] a; //会报错，因为没有初始化  
        //      System.out.println(a);  
        int[] a = new int[3];  
        System.out.println(a); // [I@4554617c 地址值  
    }  
}
```

```
/*  
    1  
    左边：  
        int: 说明数组中的元素的数据类型是int类型  
        []: 说明这是一个数组  
        arr: 是数组的名称  
    右边：  
        new: 为数组分配内存空间。  
        int: 说明数组中的元素的数据类型是int类型  
        []: 说明这是一个数组  
        3: 数组长度，其实也就是数组中元素的个数  
*/
```

```
System.out.println(arr); // [I@175078b 地址值。  
// 我要地址值没有意义啊，我就要数据值，怎么办呢？  
// 不用担心，java为你考虑到了。  
// 其实数组中的每个元素都是有编号的，并且是从0开始。最大编号是数组的长度-1。  
// 用数组名和编号的配合就可以获取数组中的指定编号的元素。这个编号的专业叫法：索引  
// 通过数组名访问数据的格式是：数组名[索引];
```

动态初始化默认值为0

4、Java中的内存分配

Java中的内存分配

- Java 程序在运行时，需要在内存中的分配空间。为了提高运算效率，就对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存管理方式。
 - 栈 存储局部变量
 - 堆 存储new出来的东西
 - 方法区 (后面讲)
 - 本地方法区 (和系统相关)
 - 寄存器 (给CPU使用)

后两个了解即可

Java程序为了提高程序的效率，就对数据进行了不同空间的分配。具体的划分为了如下5个内存空间：

- 栈：存放的是局部变量
- 堆：存放的是所有new出来的东西
- 方法区：(面向对象部分详细讲解)
- 本地方法区：(和系统相关)
- 寄存器：(CPU使用)

局部变量：在方法定义中或者方法声明上的变量都称为局部变量。

```
int[] arr = new int[3];
System.out.println(arr); //地址值
System.out.println(arr[0]); //0
System.out.println(arr[1]); //0
System.out.println(arr[2]); //0
```

栈内存的数据用完就释放。

```
{
    int a = 100;
    System.out.println(a);
}
```

堆内存的特点：

- A: 每一个new出来的东西都有地址值
- B: 每个变量都有默认值
 - byte, short, int, long 0
 - float, double 0.0
 - char '\u0000'
 - boolean false
 - 引用类型 null
- C: 使用完毕就变成了垃圾，但是并没有立即回收。会在垃圾回收器空闲的时候回收。

栈

```
int[] arr ← 0x0001
```

堆

```
new int[3]
0 0
1 0
2 0
```

栈内存在脱离作用域之后就释放

不要同时静态和动态初始化

```
int[] arr = new int[3]{1,2,3}
```

这样初始化是错误的

常见问题：

- 1、数组索引越界
- 2、空指针异常

```
/*  
    数组操作的两个常见小问题：  
    ArrayIndexOutOfBoundsException: 数组索引越界  
        原因：你访问了不存在的索引。  
  
    NullPointerException: 空指针异常  
        原因：数组已经不在指向堆内存了。  
*/
```