# 函数

方法是一段能完成独立功能的代码块。我们只需要写一次方法，可以被多次调用，提高了代码的复用性。

<public> <static> 返回值类型[void] 方法名（[数据类型 变量名, 数据类型1 变量名1，…..]）{

//方法体

[return 结果值];

}

方法的特点：

* <public> <static>:后续详解
* 返回值类型[void]：方法给调用者返回的数据类型，如果方法没有返回值，我们就使用void
* 如果方法的返回值是void，那么return;是代表程序的终止，如果方法要是有返回值那么就不能有return;
* 方法名: 见名之意，使用驼峰模式来规范方法名：xxxYyyZzz
* 参数列表：方法的参数可有可无，参数列表中可以有多个参数，先指定参数数据类型，再指定参数的变量名，多个参数用逗号分隔。
* Return：在方法有返回值的情况，返回的具体数据, 程序的返回值的值一定要和返回值类型匹配



误区：

要区分好变量的作用域问题

栈

main主方法

compareNum( ..)

int a int b

int a int b

199

10

19

10

数据共享区

Demo6.class

public static void main(String[] args)

public static int compareNum(int a, int b)

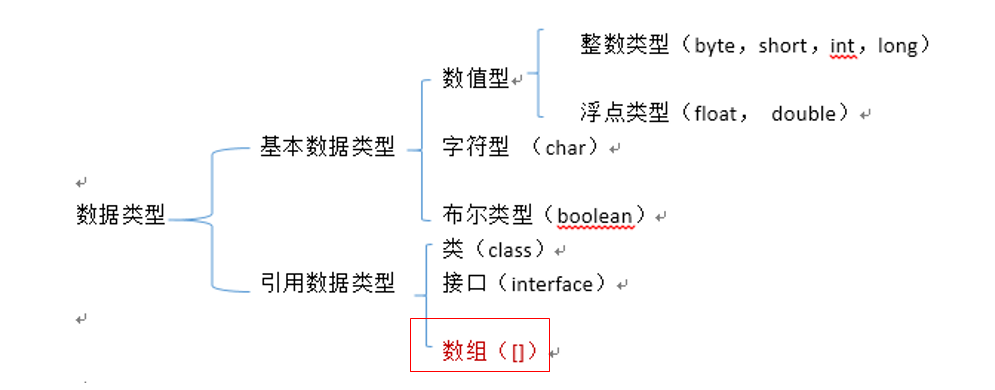
# 方法的重载

**在同一个类中函数（方法）名字，参数列表不一样（1参数个数不一样，或者对应索引位类型不一样）的两个方法是重载关系，跟返回值一点关系没有。**

目的：为了节约方法的功能命名和提高代码的可读性。

|  |
| --- |
| class Demo8{  public static void main(String[] args){    int a = 10;  int b = 19;    int result = add(a, b);  System.out.println("调用的是2个参数求和的方法: "+result);    int c = 20;  int result1 = add(a, b, c);  System.out.println("调用的是3个参数求和的方法: "+result1);    double result2 = add(a, 12.5);  System.out.println("调用的是2个参数（第二个参数是double类型）求和的方法: "+result2);  }    public static int add(int a, int b){  return a + b;  }    public static int add(int a, int b, int c){  return a + b + c;  }    public static double add(int a, double b){  return a + b;  }        } |

# 数组



数据类型[] 数组名 = new 数据类型[整数];

整数：数组的长度

画图理解：

栈 堆

（0x98）new int[8]

|  |  |
| --- | --- |
| 0 | 22 |
| 1 | 55 |
| 2 | 6 |
| 3 | 8 |
| 4 | 99 |
| 5 | 9 |
| 6 | 13 |
| 7 | 0 |

int i

int[] arr

10

0x98

基本数据类型的变量都存储在栈中，栈的特点是存储空间小，但是存取速度快，先进后出。

引用数据类型所占的空间比较大，存储在堆中，堆的特点，空间大，存取的速度比较慢。

整数类型（byte, short, int, long）的默认值都是 0

小数(float, double)的默认值都是 0.0

布尔数据类型(boolean)的默认值是 false

字符数据类型(char)的默认值是 ‘ ’

int[] arr = new int[8];

arr是数组类型的变量（如果是引用数据类型习惯上叫引用）， new int[8]真正的数据数据存储在堆中，新创建的数据的每一个位置上的值都是其所属数据类型的默认值，数组的索引是从0开始。

数组定义的第二种方式:

语法：数据类型[] 数组变量名 = {值1， 值2，……}

Int [] array = {12, 34, 56,…..};

数组的长度由value的数量来决定

数组的定义的第三种方式：

语法：数据类型[] 数组变量名 = new 数据类型[]{值1，值2，值3，……}

数组的长度由value的数量来决定

冒泡排序

第0轮4 次 第1轮3 次 第2轮2次 第3轮1次

23 25 25 25 51

25 23 23 51 25

12 12 51 23 23

7 51 12 12 12

51 7 7 7 7

相邻的两个数来做比较，如果前一个数比后一个数大，那么就要交换位置，最终每一轮的比较会产出一个最小值

轮数：数组的长度 – 1

每一轮的比较次数： 数组的长度 – 轮号 – 1

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 12 | 45 | 6 | 788 | 89 | 90 | 1 |

Length = 7

获得中间索引：length/2

获得前半段的对称索引：length – 1 – i

0 1 2 3 4 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12 | 45 | 6 | 89 | 90 | 1 |

Length = 6

获得中间索引：length/2

获得前半段的对称索引：length – 1 - i