

函数&数组

函数

函数：定义在类中，具有特定功能的一段小程序

格式：修饰符 返回值类型 函数名(形参类型 形式参数 1,形参类型 形式参数 2...)

```
{  
    函数体;  
    reutrn 返回值;  
}
```

绝逼不能嵌套定义函数

函数的特点：

A:函数与函数之间是平级关系。不能在函数中定义函数。

B:运行特点：函数只有被调用才执行。

JVM 默认调用 main 方法

函数是怎么结束的？ I
函数其实是有关键字return让他结束的。

而 void 的函数最后一行默认有 return ; （一般都不用自己写）

在 if () else 中可以用三目就用三目运算符

重载：函数名相同，参数列表不同。与返回值类型无关。

在参数传递时，有自动转换的过程，找最接近的调用

数组

1)数组是存储同一种类型的多个元素的容器。

(2)好处：数组中的元素会被自动从 0 开始编号，方便我们获取。

(3)格式：

A: `int[] arr = new int[3];`

B: `int arr[] = new int[3];`

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

D: `int[] arr = {1,2,3};` 推荐 A 和 D。

```
int[] arr = new int[3];
```

```
arr = {1,2,4}; //error,要单独赋值
```

创建一个数字数组时，所有元素都初始化为 0。boolean 数组的元素会初始化为 false。对象数组的元素则初始化为一个特殊值 null，这表示这些元素（还）未存放任何对象。初学者对此可能有些不解。例如，

```
String[] names = new String[10];
```

会创建一个包含 10 个字符串的数组，所有字符串都为 null。如果希望这个数组包含空串，可以为元素指定空串：

```
for (int i = 0; i < 10; i++) names[i] = "";
```

一旦创建数组，就不能再修改它的大小

引用类型都是存放在堆中

基本类型存放在栈内

栈：变量，或方法的执行（方法在调用时才会加载）（不可见之后释放）

堆：所有 new 出来的

整数：0 浮点数：0.0 字符：'\u0000' 布尔：false

方法区： **本地方法区：** **寄存器：**

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，有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。

栈内存

- 用于存储局部变量，当数据使用完，所占空间会自动释放。

堆内存

- 数组和对象，通过new建立的实例都存放在堆内存中。
- 每一个实体都有内存地址值
- 实体中的变量都有默认初始化值
- 实体不在被使用，会在不确定的时间内被垃圾回收器回收

方法区，本地方法区，寄存器

局部变量：定义在方法中或者

数组操作的常见问题：

A:数组越界异常。你访问了不存在的索引。

ArrayIndexOutOfBoundsException

B:空指针异常。一个实例(对象)已经不存在了，你还去访问它的内容。

NullPointerException

两个引用对象

```
int[] a = {1,2,3};
```

第二个是通过直接赋值而得

```
int[] b = a;
```

若一个改变，另一个也随之改变。 `b[1] = 4;` ----则 `a[1]` 的值为 4

注意： `a` 的值 == `b` 的值 两个变量指向同一个堆内存空间

（两个变量引用同一个数组）

在设置数组查找函数时，可以设置为查找到返回-1

二维数组

java 中实际只有一维数组，多维数组是“数组的数组”

二维数组：

格式：

A: `int[][] arr = new int[3][2];`

B: `int[][] arr = new int[3][];`


C: `int[][] arr = {{1,2,3},{4,5},{6,7,8,9}};`

在二维数组中打印：`arr[0]` 其值为一个地址值，因为第一维中存的是指向二维数组的引用。


Eg: `int[][] arr = new int[3][2];`

`arr.length` ----- 是第一维的长度

`arr[i].length` ----- 是第二维的长度

 **注释：**for each 循环语句不能自动处理二维数组的每一个元素。它是按照行，也就是一维数组处理的。要想访问二维数组 a 的所有元素，需要使用两个嵌套的循环，如下所示：

```
for (double[] row : a)
    for (double value : row)
        do something with value
```

 **提示：**要想快速地打印一个二维数组的数据元素列表，可以调用：

```
System.out.println(Arrays.deepToString(a));
```

输出格式为：

```
[[16, 3, 2, 13], [5, 10, 11, 8], [9, 6, 7, 12], [4, 15, 14, 1]]
```

数组常用方法

一、数组本身的方法：

- 1、**length** ----注意不要到括号！若为二维则为其第一维的长度

二、在 java.util.Arrays;中的 Arrays 类

调用格式 eg: `Arrays.toString(a)`

- 1、**toString(--)**; ---输出是带中括号的输出 (String)

- 2、**copyOf(--)**; --- 从数组中赋值，设置其长度

`copyOf(int[] original, int newLength);`

`copyOfRange(boolean[] original, int from, int to);`

- 3、**sort(--)**; --进行升序排序

`sort(int[] a, int fromIndex, int toIndex)`

- 4、**binarySearch()**; --二分法查找

`binarySearch(int[] a, int key)`

`binarySearch(int[] a, int fromIndex, int toIndex, int key)`

- 5、**equal ()** ---判断两个数组是否相同 (boolean)

`equals(int[] a, int[] a2)` --比较数组 a、b 是否相等

- 6、**fill ()** ---向数组中填充指定的值 (void)

`fill(数组, 填充值)` -- 向数组中所有元素填充相同的值

`fill(boolean[] a, int fromIndex, int toIndex, boolean val)`—向数组 a 中，指定开始与结尾处填充值 val

- 7、**hashCode(--)**; ---返回哈希码，并不是地址码 (int)

类

算法+数据结构=程序

类（class）是构造对象的模板或蓝图

由类构造对象的过程称为：创建类的实例（instance）

对象可以调用类中定义的方法

要想使用 OOP，一定要清楚对象的三个主要特性：

- 对象的行为（behavior）——可以对对象施加哪些操作，或可以对对象施加哪些方法？
- 对象的状态（state）——当施加那些方法时，对象如何响应？
- 对象标识（identity）——如何辨别具有相同行为与状态的不同对象？