👀

数组

Java 中的**数组 Array** 是**一种容器对象**，可以用来存储**固定大小的相同类型的数据**

不论是基本数据类型（`int`，`double` 等）还是引用类型（`String`，自定义类等），Java 都支持数组形式

▼ 基本特性

特性	说明
类型固定	数组中所有元素必须是同一种类型
长度固定	创建后长度不可改变
索引从 0 开始	第一个元素是 `arr[0]`
可以存基本类型 / 引用类型	如 `int[]`，`String[]`，`Person[]`

▼ 为什么数组可以直接用，即使没有创建

数组在 JDK 中是特殊的类型，JDK专门用虚拟机的一些特殊指令来负责创建数组

✅ 情况1：看起来没创建，其实是“隐式”创建了

在 Java 中，**数组的创建**其实是有的，只是有时候它是隐藏在语法糖中的

```
int[] arr = {1, 2, 3, 4, 5};
```

这行代码虽然没有写 `new`，但实际上 **Java 编译器会自动把它翻译成：**

```
int[] arr = new int[]{1, 2, 3, 4, 5};
```

✅ 情况2：有时候数组是作为参数传进来的

比如在方法里看到：

```
public static void printArray(int[] arr) {
    for (int i : arr) {
        System.out.println(i);
    }
}
```

虽然在方法内部没有看到数组的创建，但其实这个 `arr` 是调用者传进来的

```
printArray(new int[]{10, 20, 30});
```

```
int[] nums = {10, 20, 30};
printArray(nums);
```

数组早就已经在别的地方被创建好了

✅ 情况3：主方法参数 `String[] args`

Java 程序的主方法是这样写的：

```
public static void main(String[] args) {
    System.out.println(args.length);
}
```

你没看到创建 `args`，但它是 JVM 在调用你的程序时，**自动传入的字符串数组**（命令行参数），所以不是你创建的，但已经被传了进来

▼ 3 种方式创建数组

1️⃣

```
int[] arr = new int[3];       // 创建长度为 3 的 int 数组
```

2️⃣

```
int[] arr = new int[] {5,4,3,2,1}  // 开辟长度为 5 的空间, 将这些数字填充进去
```

3️⃣

```
int[] arr = {5,4,3,2,1};
```

▼ 初始化

✅ 静态初始化（直接赋值）：

```
int[] nums = {1, 2, 3, 4};
String[] strs = {"hello", "world"};
```

✅ 动态初始化（先分配空间，再赋值）：

```
int[] nums = new int[3];
nums[0] = 10;
nums[1] = 20;
nums[2] = 30;
```

▼ 内存结构

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

- `arr` 是引用变量，存在 **栈** 中
- `new int[3]` 创建了一个数组对象，存在 **堆** 中
- `arr` 保存的是 **堆中数组对象的地址**

▼ 多维数组

```
int[][] matrix = new int[2][3];  // 两行三列的二维数组
```

赋值方式：

```
matrix[0][0] = 1;
matrix[1][2] = 9;
```

> Java 中的数组是“固定长度的同类型变量集合”，可以容纳基本类型和对象引用，是一个对象，在内存中存在堆里，用引用变量指向