数组的定义和使用

# 引言

1. 数组的基本定义；
2. 数组的使用；
3. 数组与方法间的互操作；
4. 数组有关的操作类库支持。

# 具体内容

所有的开发之中都一定要使用到数组，但是一般都不会用得太复杂。

## 数组的基本概念

数组指的是一组相关变量的集合。如果说现在要求定义100个整型变量，那么按照最原始的方式则肯定这样定义：

|  |
| --- |
| int i1;  int i2;  …  int i100; |

这种操作可以实现要求，但是这些变量的关联实在是太麻烦了。为此在开发之中可以利用数组来解决这一问题。

在Java中数组属于引用数据类型，既然是引用数据类型就牵扯到内存的关系。对于数组的定义语法有以下两种形式：

* 声明并开辟数组： 数据类型[] 数组名称 = new 数据类型[长度]；

数据类型 数组名称[] = new 数据类型[长度]；

* 分布完成：
  + 声明数组：数据类型[] 数组名称 = null；
  + 开辟数组：数组名称 = new 数据类型[长度]。

当数组开辟空间之后那么就可以采用“数组[索引]”的形式进行数组的访问，但是需要注意的是，如果数组的长度为3，那么索引的范围：0~2（一共三个元素）。如果操作中超过了数组的允许索引范围，则呈现在运行过程之中会出现“ArrayIndexOutOfBoundsException”(数组索引超出绑定异常，数组越界)。

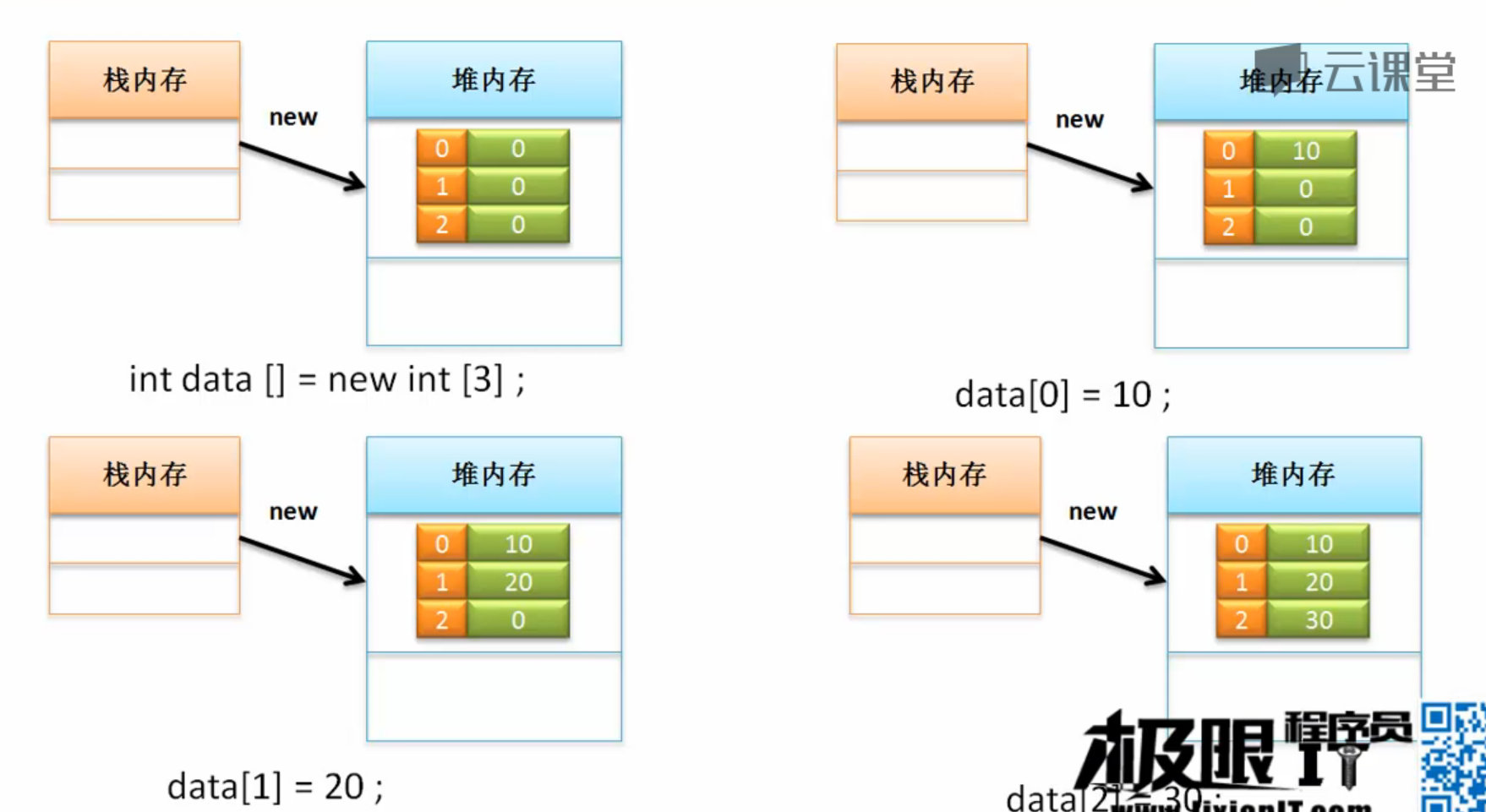
以下的操作属于数组的动态初始化，动态初始化的特点，是数组开辟空间之后，数组中每个元素的内容都其对应数据类型的默认值。

范例：定义数组

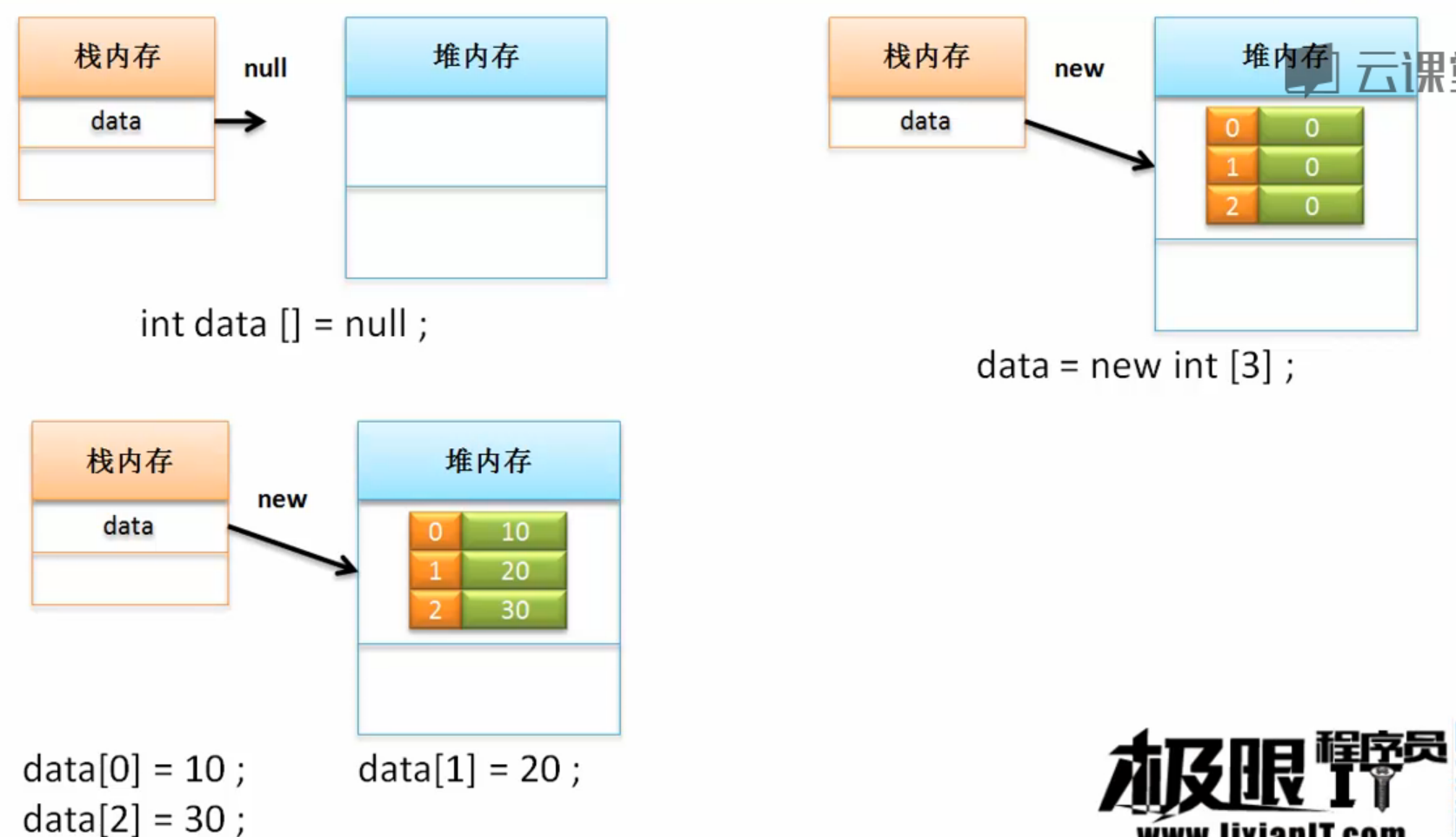
|  |
| --- |
| public class ArrayDemo {  public static void main(String[] args) {  int[] data = new int[3];  System.out.println(data[0]);  System.out.println(data[1]);  System.out.println(data[2]);  }  } |

## 数组的引用分析

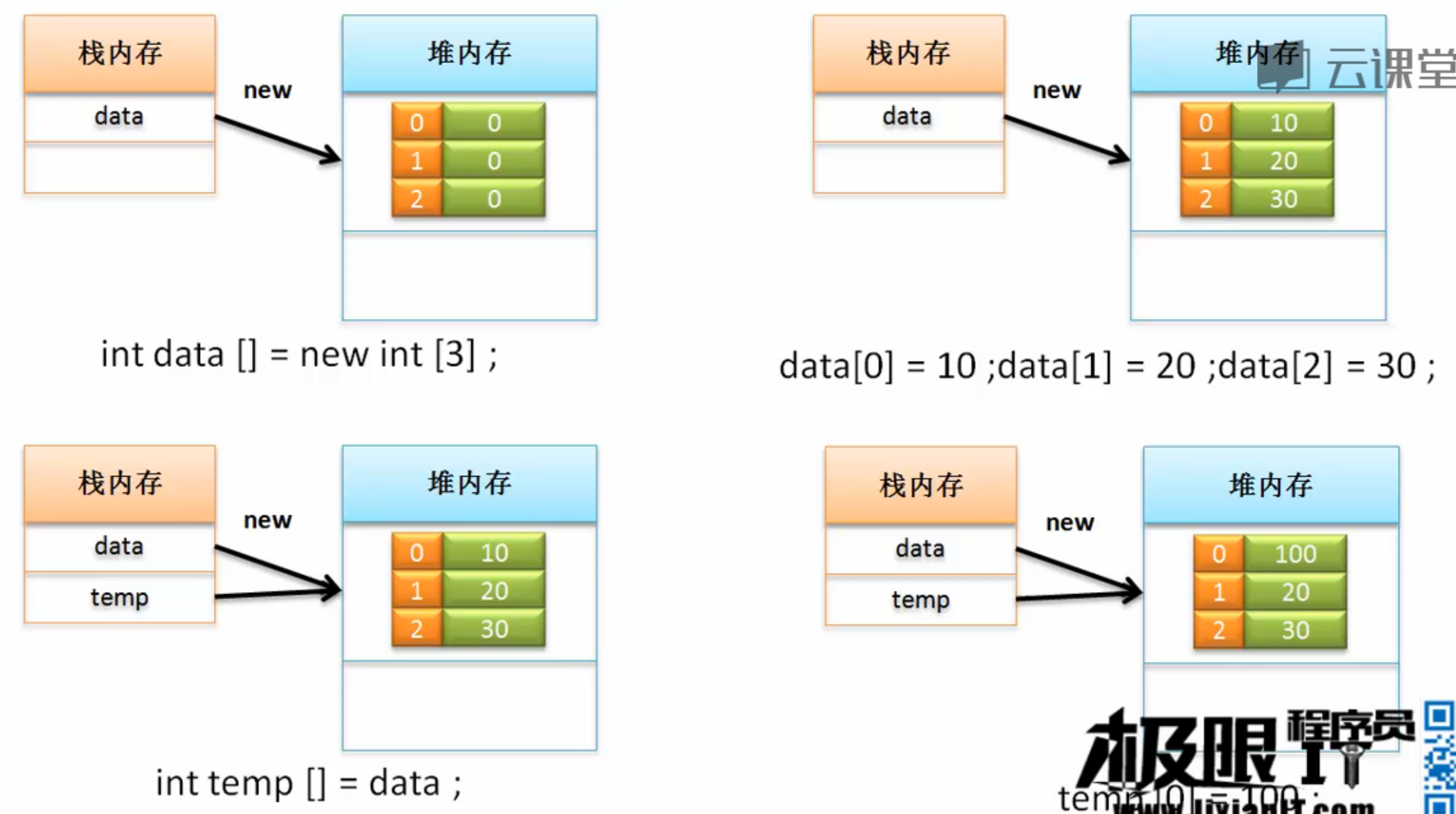
引用数据类型的分析过程几乎都是一样的，所以数组的引用分析其本质也是一样的，与对象的流程是相同的。



以上的流程与普通对象本质上是没有区别的，唯一的区别是在于普通的类对象是保存属性，利用属性名称来操作，但是数组保存的是一组内容，利用索引来操作。



那么现在如果此关系可以明白的话，就意味着可以进行数组的引用操作了。引用的本质同一块堆内存空间被不同的占内存所指向。



## 数组的静态初始化

以上是针对于数组的动态初始化操作讲解，动态初始化操作的特点在于其只能够在开辟数组空间之后进行数组内容的赋值，如果现在希望数组开辟之后就可以存在明确的内容，那么则可以使用数组的静态初始化，而语法有以下两种：

第一种：简化型

|  |
| --- |
| 数据类型[] 数据名称 = {值,值,值,…}; |

第二种：完全型

|  |
| --- |
| 数据类型[] 数据名称 = new 数据类型[] {值,值,值,…}; |

## 数组与方法

在数组与方法进行传递的操作过程之中，实际上就属于引用传递。在数组与方法间的操作中有两种：方法接收数组、方法返回数组。不管是接收还是返回，最终操作的一定是一个堆内存的使用。

## 与数组有关的类库支持

Java本身提供有一系列的开发类库帮助用户进行代码编写，那么首先给出两个基础的数组操作方法：

* 数组拷贝：System.arraycopy(原始数组，原始数组开始点，目标数组，目标数组的开始点，拷贝长度);
* 数组排序：java.util.Arrays.sort(数组名称);

# 总结

1. 数组属于引用数据类型；
2. 数组属于线性的存储结构，里面的内容可以根据索引线性操作；
3. 数组里面容易出现面试题。