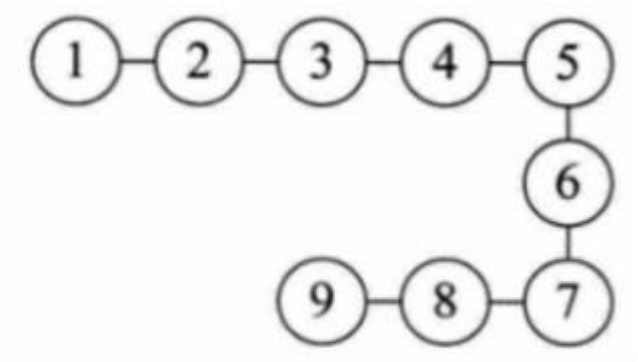
动态数组

什么是数据结构?

- 数据结构是计算机存储、组织数据的方式



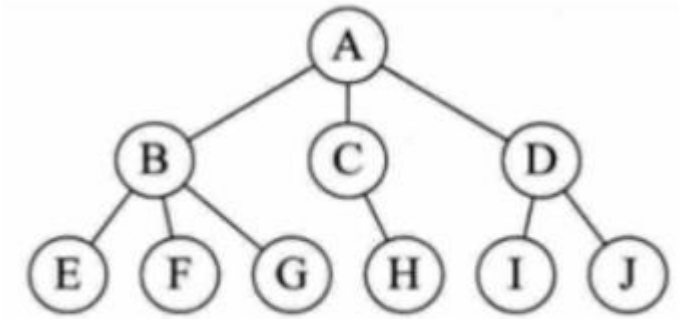
线性结构

线性表

(数组、链表、

栈、队列、

哈希表)



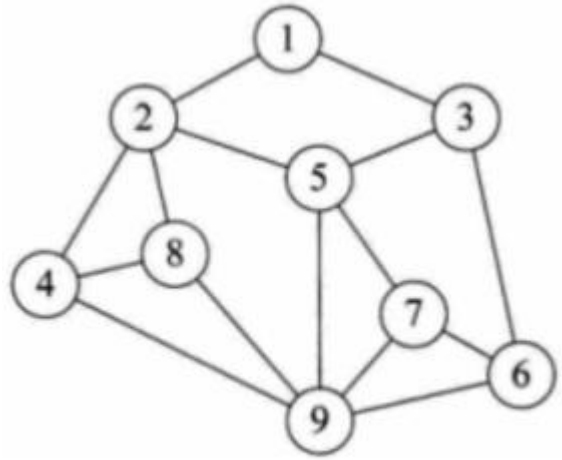
树形结构

二叉树

AVL树、红黑树

B树、堆、Trie

哈夫曼树、并查集



图形结构

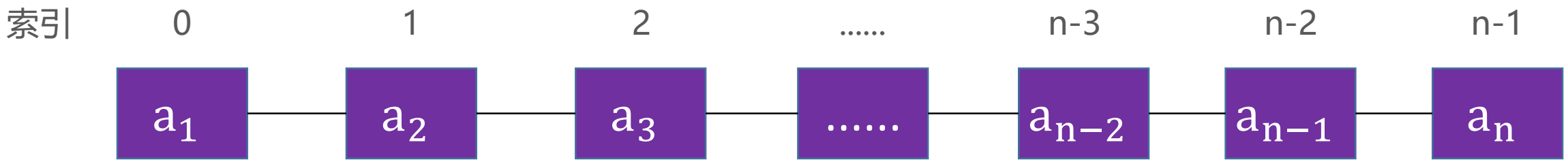
邻接矩阵

邻接表

- 在实际应用中，根据使用场景来选择最合适的数据结构

线性表

- 线性表是具有 n 个相同类型元素的有限序列（n ≥ 0）



- a_1 是首节点（首元素），a_n 是尾结点（尾元素）
- a_1 是 a_2 的前驱，a_2 是 a_1 的后继

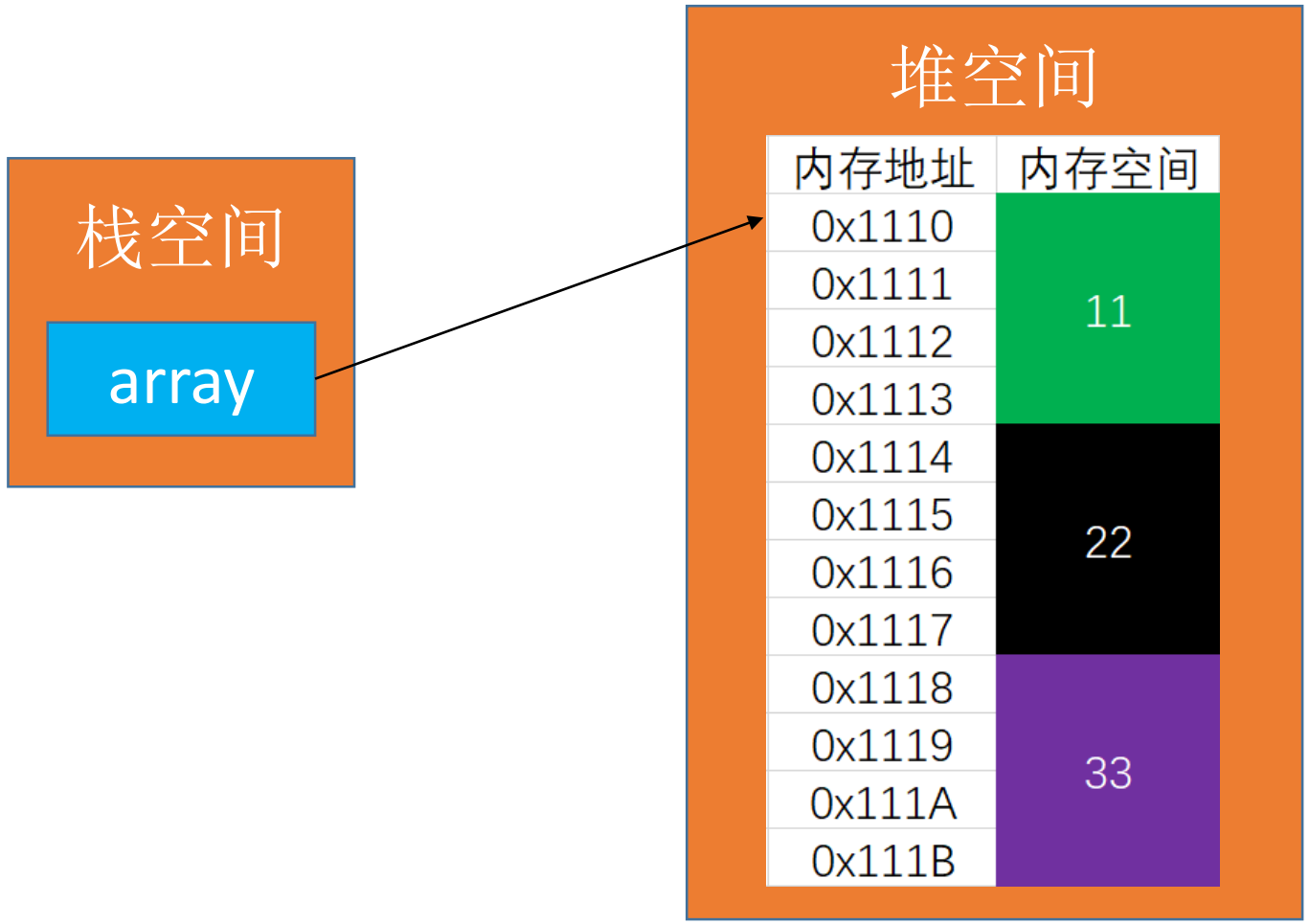
- 常见的线性表有
  - 数组
  - 链表
  - 栈
  - 队列
  - 哈希表（散列表）

生活中的线性表

数组（Array）

- 数组是一种顺序存储的线性表，所有元素的内存地址是连续的

```
int[] array = new int[]{11, 22, 33};
```

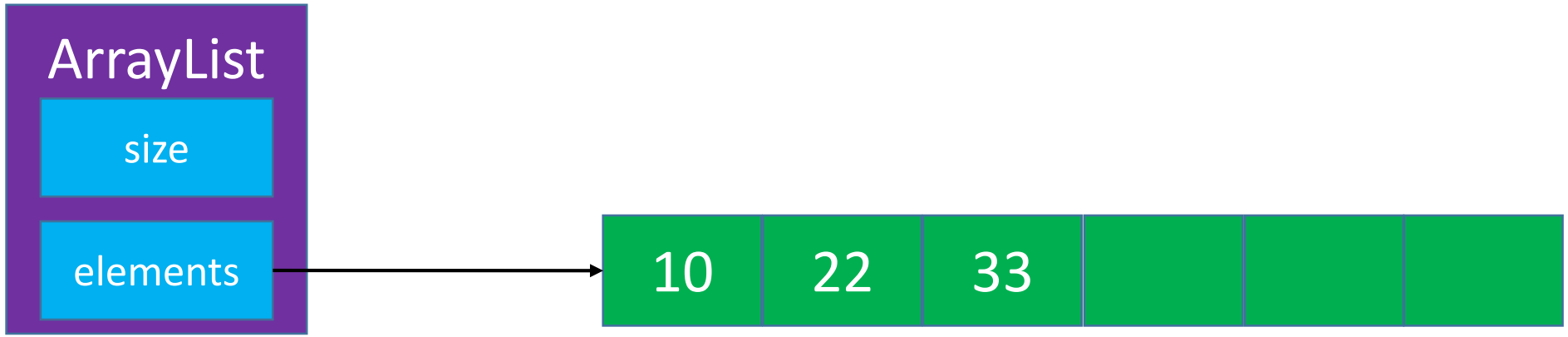


- 在很多编程语言中，数组都有个致命的缺点
  - 无法动态修改容量
- 实际开发中，我们更希望数组的容量是可以动态改变的

动态数组（Dynamic Array）接口设计

- `int size();  // 元素的数量`
- `boolean isEmpty();  // 是否为空`
- `boolean contains(E element);  // 是否包含某个元素`
- `void add(E element);  // 添加元素到最后面`
- `E get(int index);  // 返回index位置对应的元素`
- `E set(int index, E element);  // 设置index位置的元素`
- `void add(int index, E element);  // 往index位置添加元素`
- `E remove(int index);  // 删除index位置对应的元素`
- `int indexOf(E element);  // 查看元素的位置`
- `void clear();  // 清除所有元素`

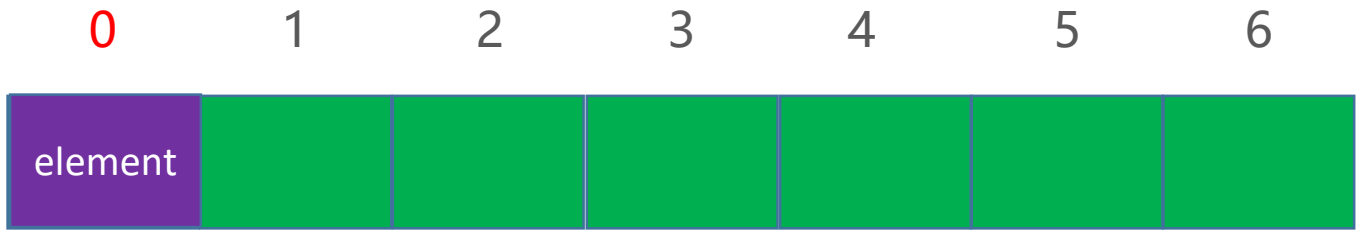
动态数组的设计



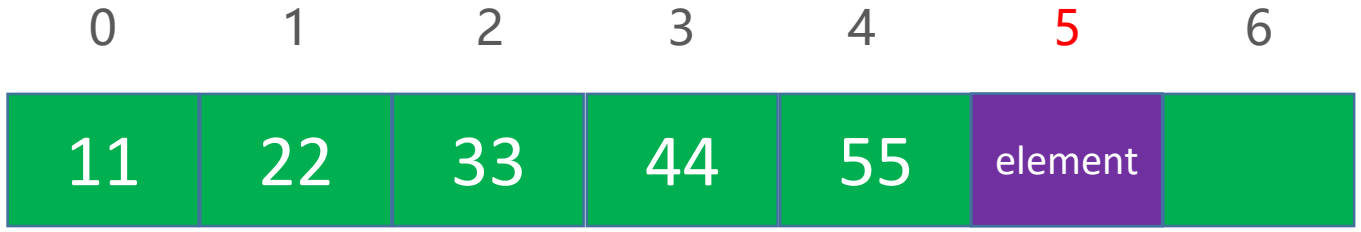
- 在Java中，成员变量会自动初始化，比如
  - `int` 类型自动初始化为 `0`
  - 对象类型自动初始化为 `null`

添加元素 - add(E element)

当 size 等于 0 时



当 size 等于 5 时




```
elements[size] = element;
size++;
```

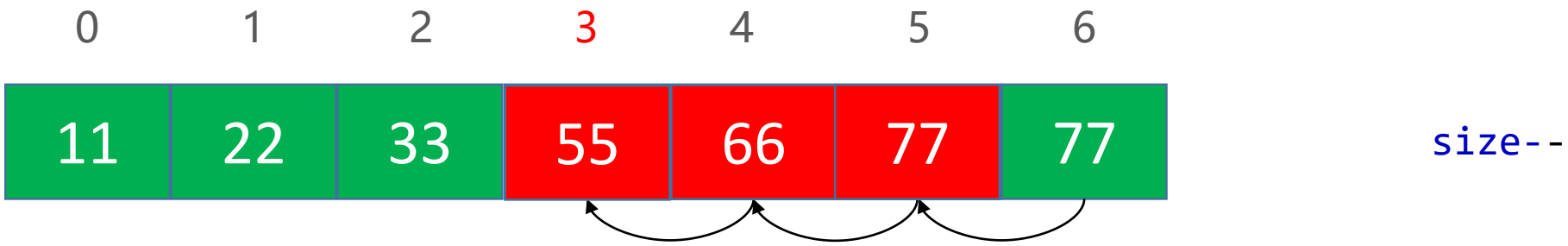
打印数组

- 重写 `toString` 方法
- 在 `toString` 方法中将元素拼接成字符串
- 字符串拼接建议使用 `StringBuilder`

删除元素 - remove(int index)

size 等于 7

index 等于 3

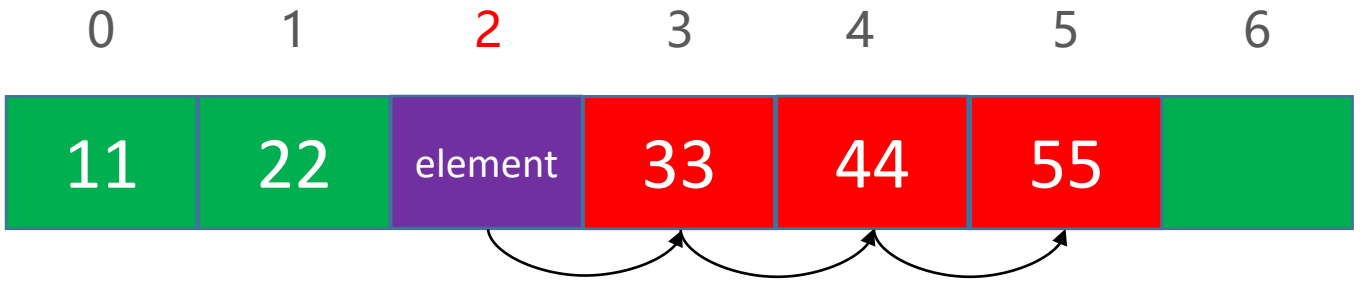


思考：最后一个元素如何处理?

添加元素 - add(int index, E element)

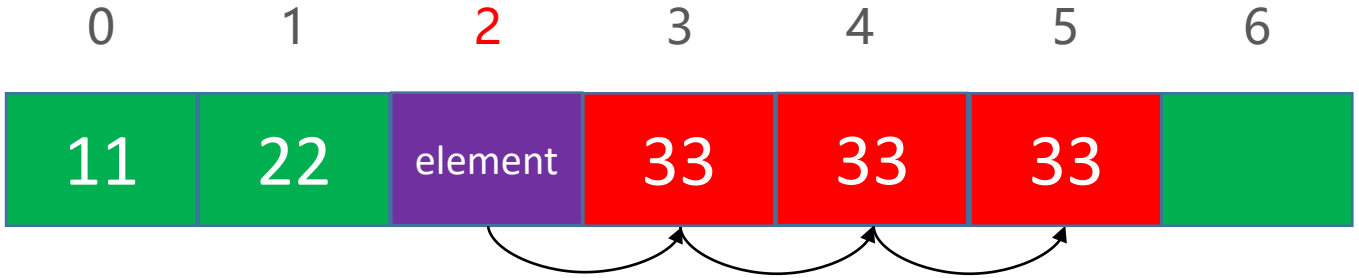
size 等于 5

index 等于 2

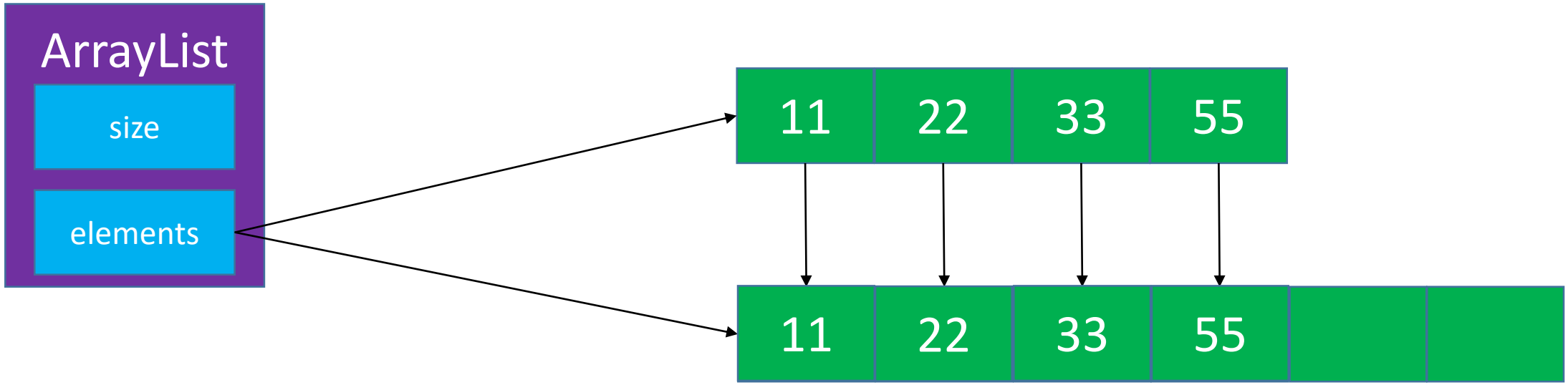


size++

错误的覆盖顺序



如何扩容



泛型

- 使用泛型技术可以让动态数组更加通用，可以存放任何数据类型

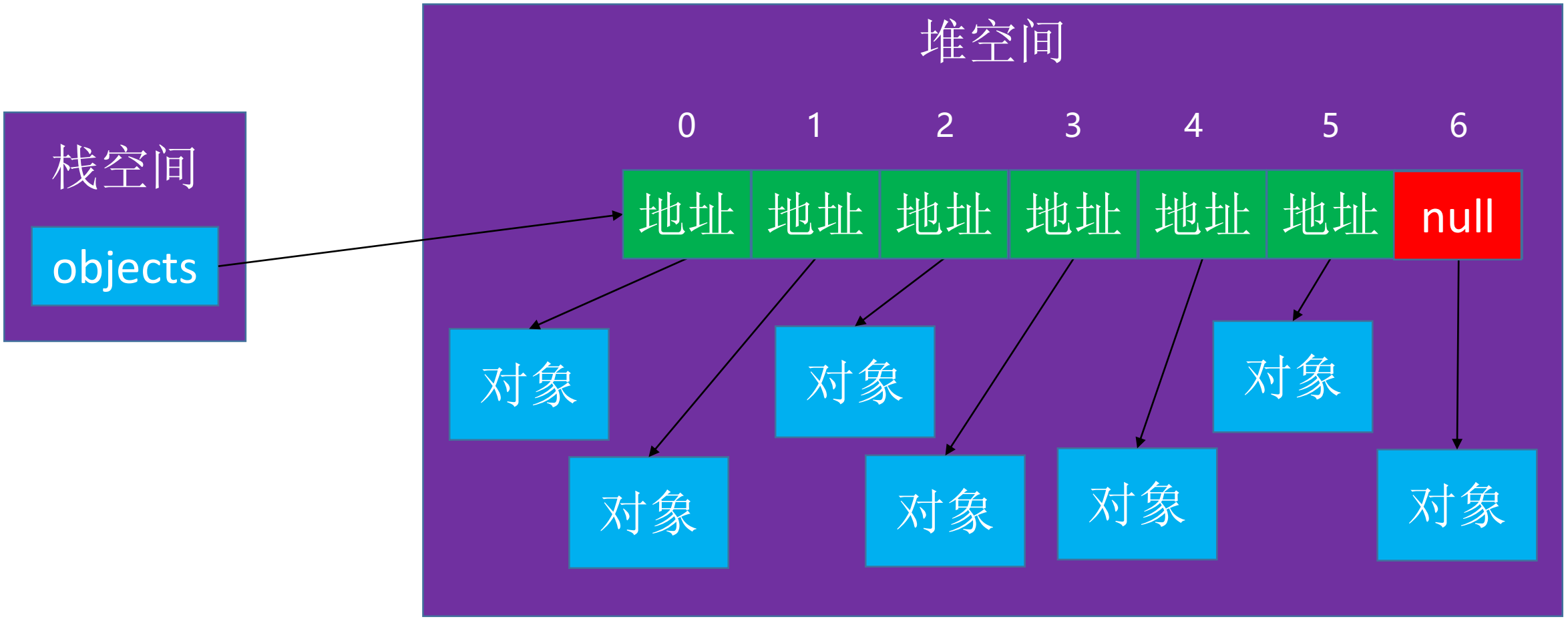
```
public class ArrayList<E> {
    private int size;
    private E[] elements;
```

```
elements = (E[]) new Object[capacity];
```

```
ArrayList<Integer> list = new ArrayList<>();
```

对象数组

```
Object[] objects = new Object[7];
```



内存管理细节

```
public void clear() {
    for (int i = 0; i < size; i++) {
        elements[i] = null;
    }
    size = 0;
}
```

```
public E remove(int index) {
    rangeCheck(index);
    E oldElement = elements[index];
    for (int i = index; i < size - 1; i++) {
        elements[i] = elements[i + 1];
    }

    elements[--size] = null;
    return oldElement;
}
```

null的处理

- 一个内部设计方面的问题
- 是否可以存储 `null` 数据?

```
public int indexOf(E element) {
    if (element == null) {
        for (int i = 0; i < size; i++) {
            if (elements[i] == null) return i;
        }
    } else {
        for (int i = 0; i < size; i++) {
            if (elements[i].equals(element)) return i;
        }
    }
    return ELEMENT_NOT_FOUND;
}
```

java.util.ArrayList

■ JDK中内置了一个动态数组类：`java.util.ArrayList`

■ 源码分析