第五章 数组

本章内容

- 一维数组的声明和初始化
- 数组元素的引用
- 二维数组的声明和使用
- 数组拷贝

数组概述

- 数组可以看成是多个相同类型数据组合，对这些数据的统一管理。
- 数组变量属引用类型，数组也可以看成是对象，数组中的每个元素相当于该对象的成员变量。
- 数组中的元素可以是任何数据类型，包括基本类型和引用类型。

一维数组的声明

- 一维数组的声明方式：

  type var[]; 或 type[] var;

- 例如：

  int a1[]; int[] a2;

  double b[];

  Person[] p1; String s1[];

- Java语言中声明数组时不能指定其长度(数组中元素的个数)，例如：

  int a[5]; //非法

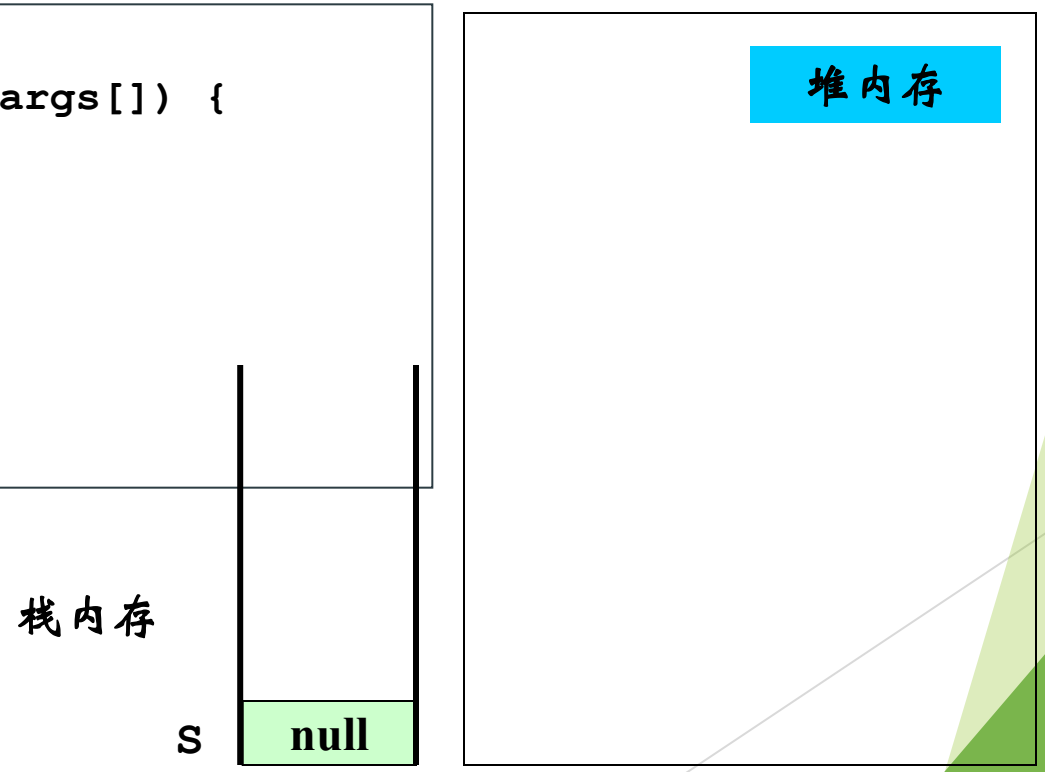
数组对象的创建

- Java中使用关键字 new 创建数组对象，格式为：

  数组名 = new 数组元素的类型 [数组元素的个数]

- 例如：

```
public class Test {
  public static void main(String args[]) {
    int[] s;
    s = new int[5];
    for (int i = 0; i < 5; i++) {
      s[i] = 2 * i + 1;
    }
  }
}
```



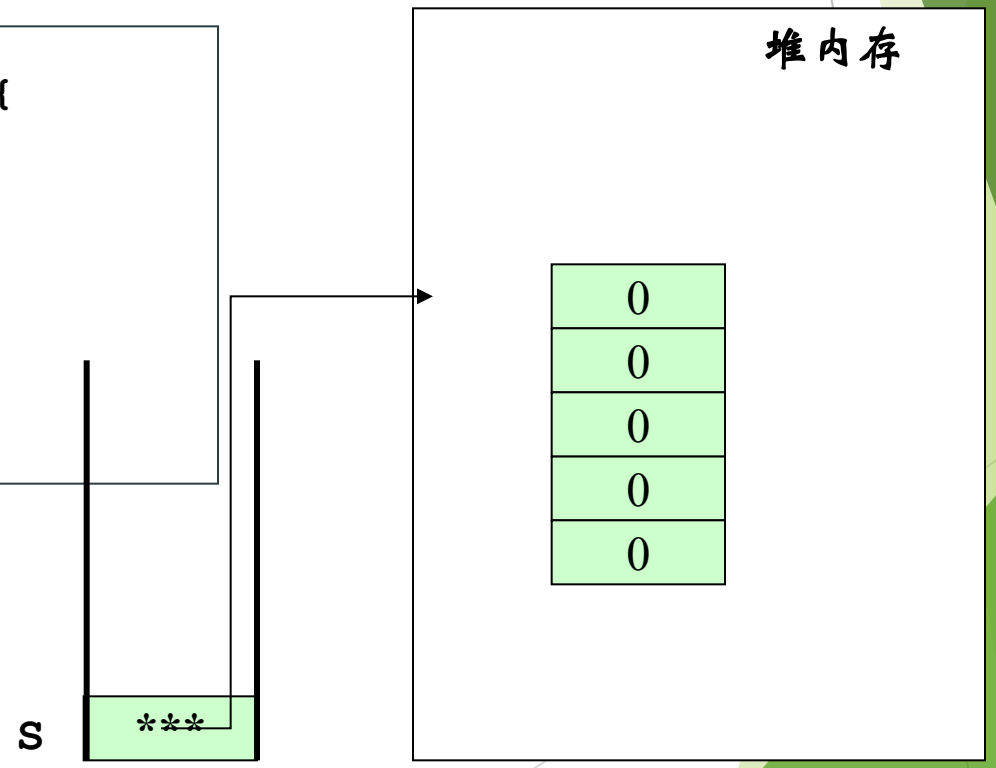
数组对象的创建

- Java中使用关键字 new 创建数组对象，格式为：

  数组名 = new 数组元素的类型 [数组元素的个数]

- 例如：

```
public class Test {
  public static void main(String args[]) {
    int[] s;
    s = new int[5];
    for (int i = 0; i < 5; i++) {
      s[i] = i;
    }
  }
}
```



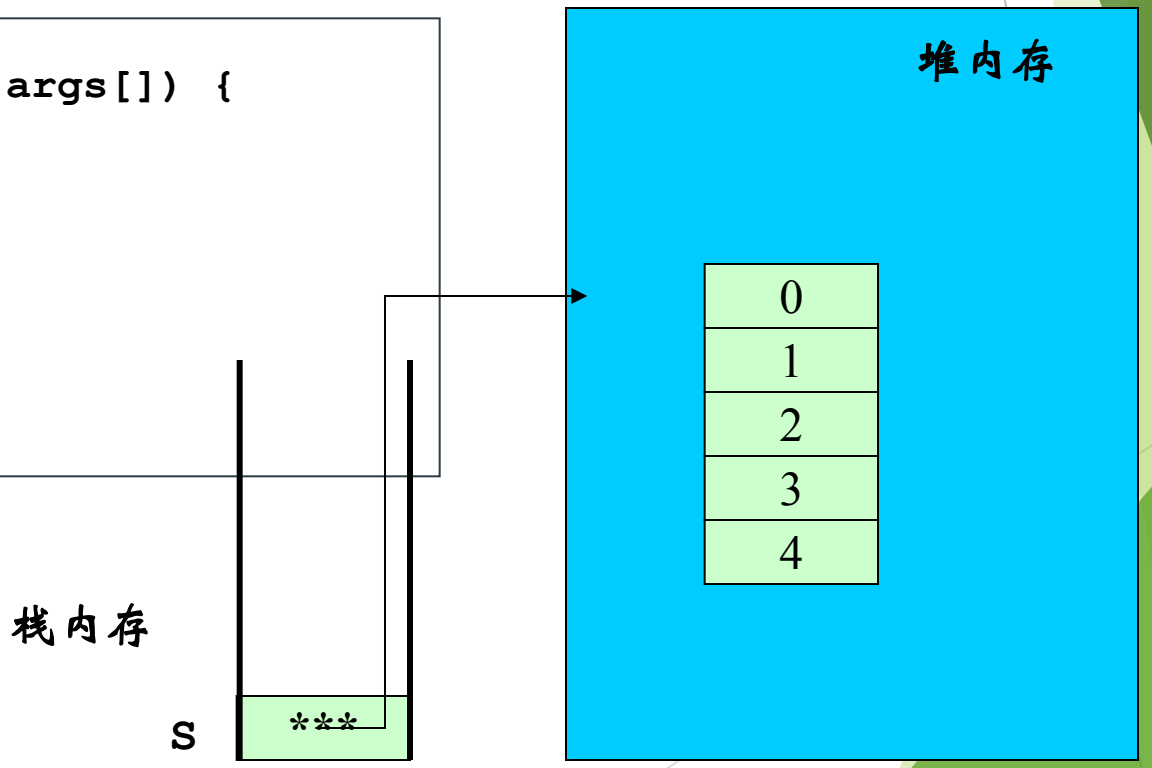
数组对象的创建

- Java中使用关键字 new 创建数组对象，格式为：

  数组名 = new 数组元素的类型 [数组元素的个数]

- 例如：

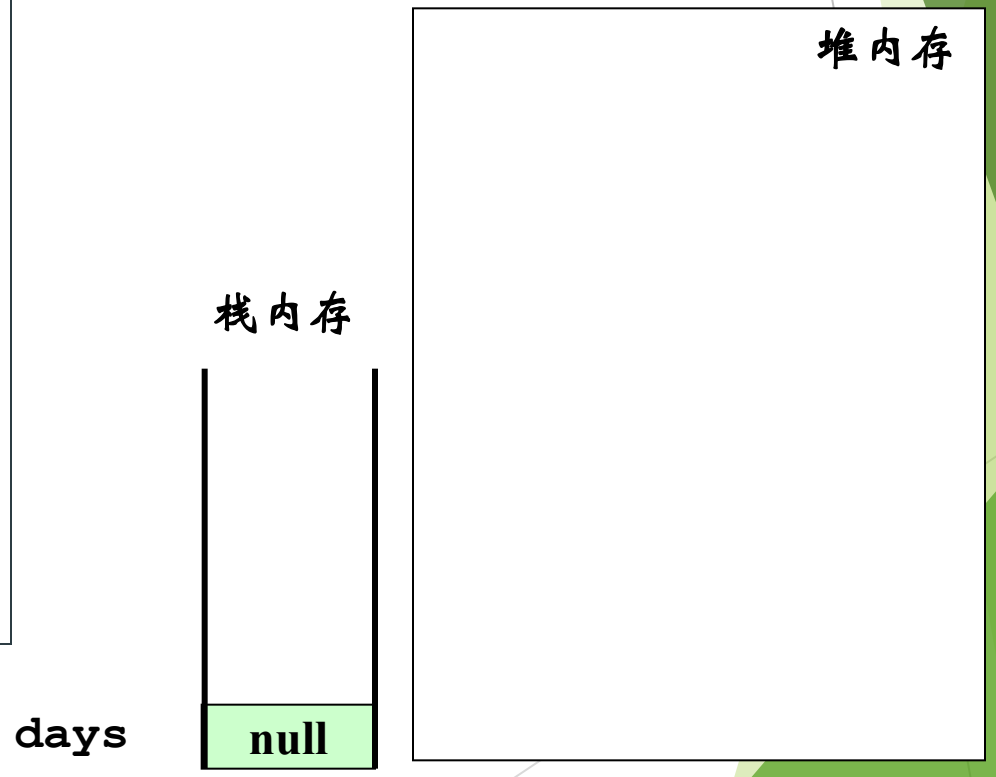
```
public class Test {
  public static void main(String args[]) {
    int[] s;
    s = new int[5];
    for (int i = 0; i < 5; i++) {
      s[i] = i;
    }
  }
}
```



元素为引用数据类型的数组

- 注意：元素为引用数据类型的数组中的每一个元素都需要实例化。

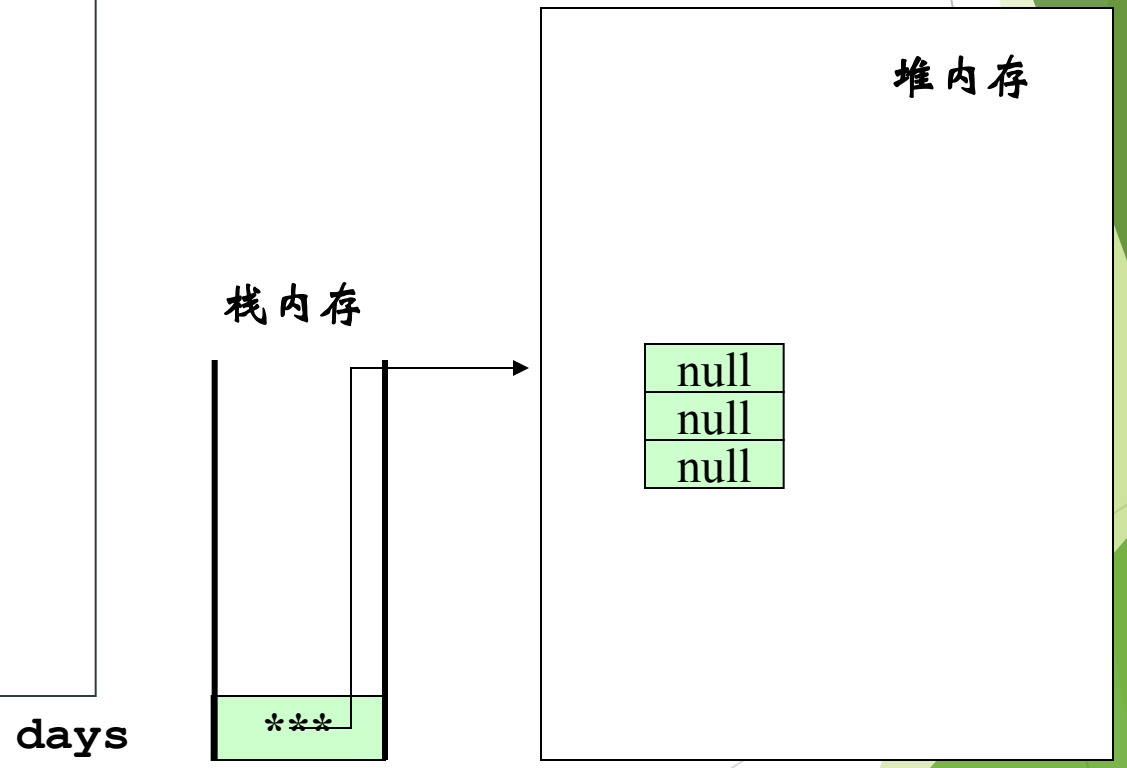
```
public class Test {
  public static void main(String args[]) {
    Date[] days;
    days = new Date[3];
    for (int i = 0; i < 3; i++) {
      days[i] = new Date(2004,4,i+1);
    }
  }
}
class Date {
  int year; int month; int day;
  Date(int y,int m,int d){
    year = y; month = m;
    day = d;
  }
}
```



元素为引用数据类型的数组

- 注意：元素为引用数据类型的数组中的每一个元素都需要实例化。

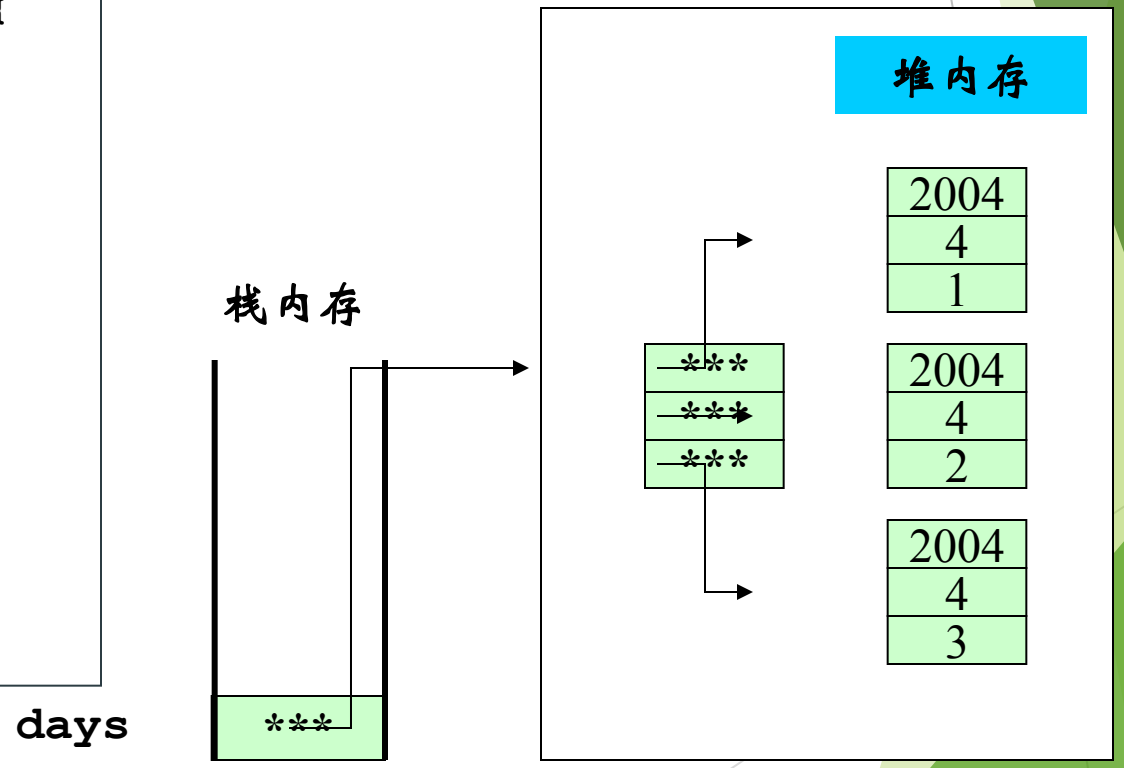
```
public class Test {
  public static void main(String args[]) {
    Date[] days;
    days = new Date[3];
    for (int i = 0; i < 3; i++) {
      days[i] = new Date(2004,4,i+1);
    }
  }
}
class Date {
  int year; int month; int day;
  Date(int y,int m,int d){
    year = y; month = m;
    day = d;
  }
}
```



元素为引用数据类型的数组

- 注意：元素为引用数据类型的数组中的每一个元素都需要实例化。

```
public class Test {
  public static void main(String args[]) {
    Date[] days;
    days = new Date[3];
    for (int i = 0; i < 3; i++) {
      days[i] = new Date(2004,4,i+1);
    }
  }
}
class Date {
  int year; int month; int day;
  Date(int y,int m,int d){
    year = y; month = m;
    day = d;
  }
}
```



数组初始化（1）

- 动态初始化

  数组定义与为数组元素分配空间和赋值的操作分开进行，例如：

```
public class Test {
    public static void main(String args[]) {
        int a[];
        a = new int[3];
        a[0] = 3; a[1] = 9; a[2] = 8;
        Date days[];
        days = new Date[3];
        days[0] = new Date(1, 4, 2004);
        days[1] = new Date(2, 4, 2004);
        days[2] = new Date(3, 4, 2004);
    }
}
class Date {
    int year,month,day;
    Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
}
```

数组初始化（2）

- 静态初始化

  在定义数组的同时就为数组元素分配空间并赋值，例如：

```
public class Test {
    public static void main(String args[]) {
        int a[] = new int[]{ 3, 9, 8 };
        Date days[] = {
            new Date(1, 4, 2004),
            new Date(2, 4, 2004),
            new Date(3, 4, 2004)
        };
    }
}
class Date {
    int year,month,day;
    Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
}
```

数组元素的默认初始化

- 数组是引用类型，它的元素相当于类的成员变量，因此数组分配空间后，每个元素也被按照成员变量的规则被隐式初始化，如：

```
public class Test {
    public static void main(String args[]) {
        int a[] = new int[5];
        Date[] days = new Date[3];
        System.out.println(a[3]);
        System.out.println(days[2]);
    }
}
class Date {
    int year,month,day;
    Date(int y, int m, int d) {
        year = y; month = m; day = d;
    }
}
```

- 输出结果：

```
0
null
```

数组元素的引用

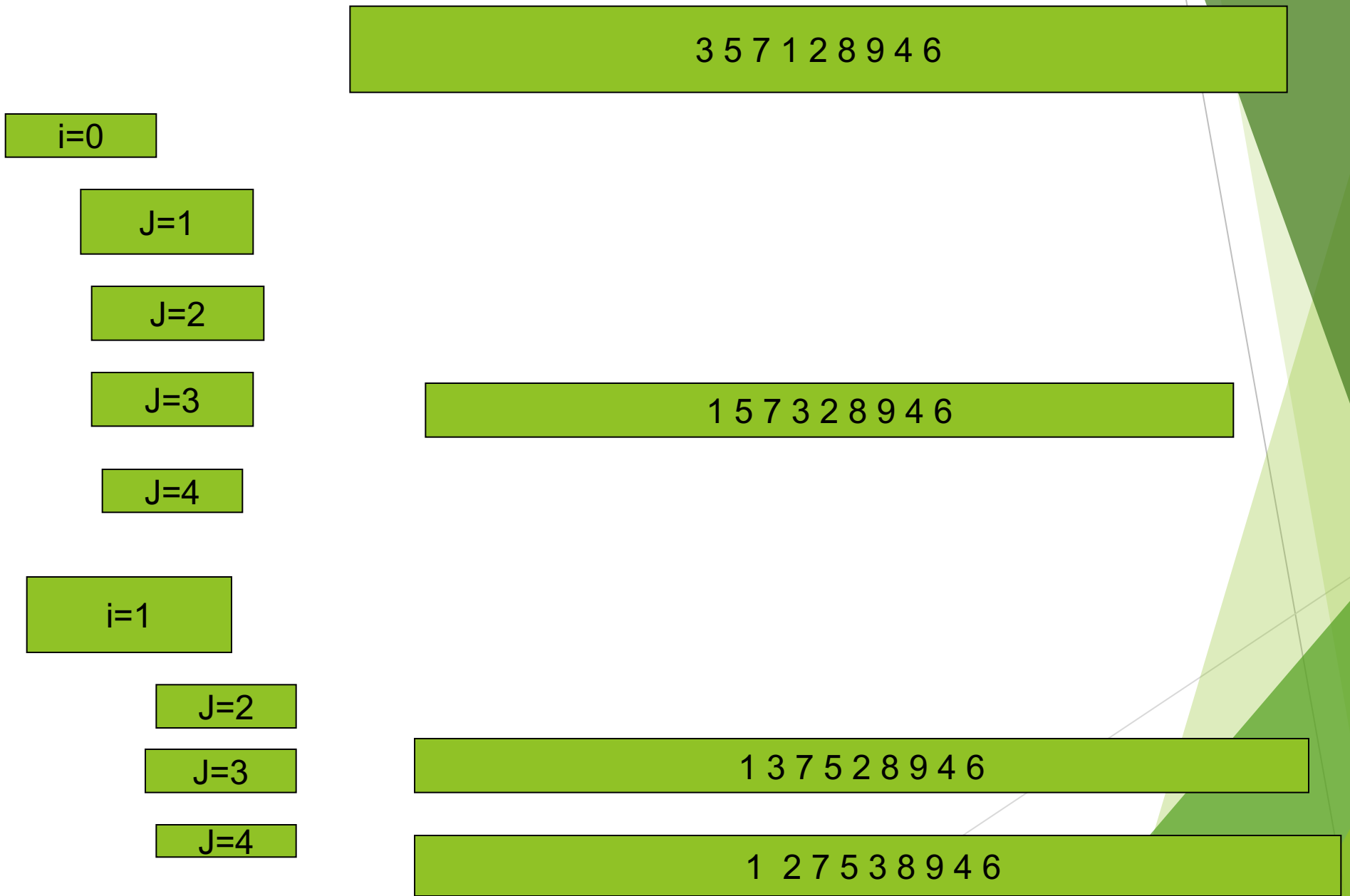
- 定义并用运算符new为之分配空间后后，才可以引用数组中的每个元素，数组元素的引用方式为：
  - arrayName[index]
    - index为数组元素下标，可以是整型常量或整型表达式。如：

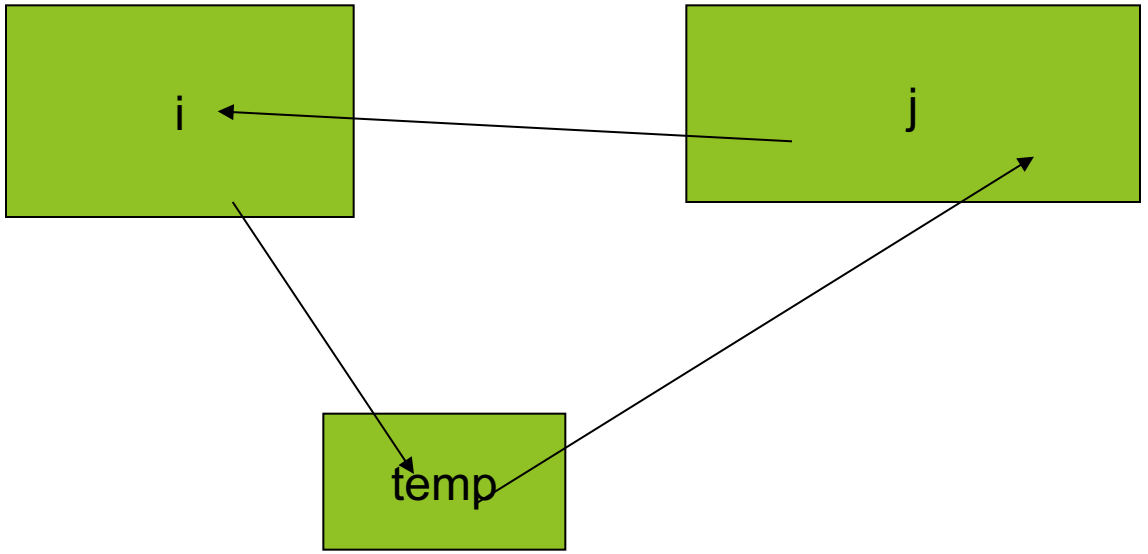
    a[3], b[i], c[6*i] 。
  - 数组元素下标从0开始；长度为n的数组的合法下标取值范围为

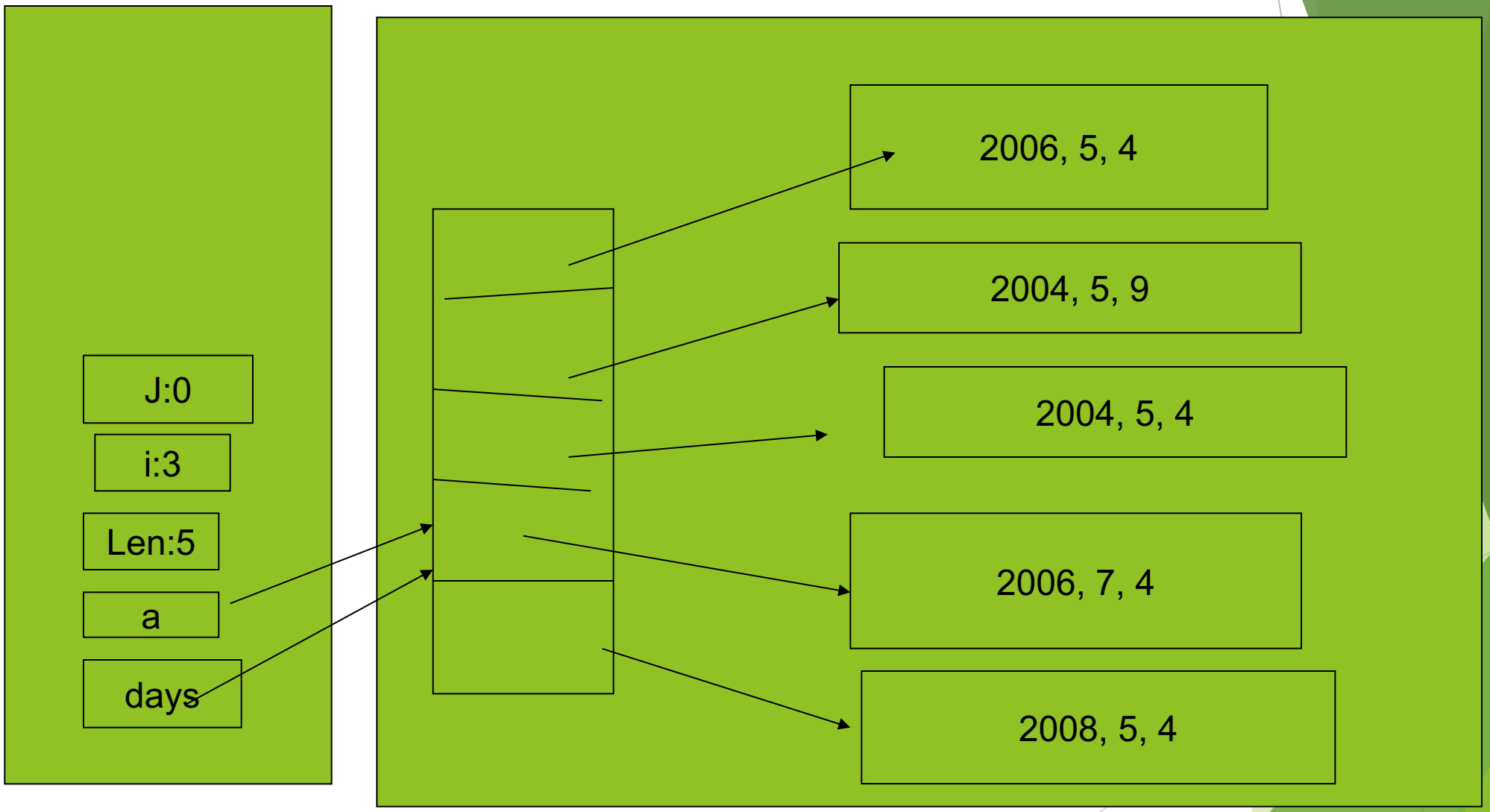
    0 ～ n-1 。

- 每个数组都有一个属性length指明它的长度，例如：
  - a.length的值为数组a的长度(元素个数)。

TestArray.java TestArgs.java NumSort.java TestDateSort.java


3 5 7 1 2 8 9 4 6
i=0
J=1
J=2
J=3
1 5 7 3 2 8 9 4 6
J=4
i=1
J=2
J=3
1 3 7 5 2 8 9 4 6
J=4
1 2 7 5 3 8 9 4 6


i
j
temp


J:0
i:3
Len:5
a
days
2006, 5, 4
2004, 5, 9
2004, 5, 4
2006, 7, 4
2008, 5, 4

课 堂 练 习

课堂练习

阅读下面程序，任意初始化days数组，然后对days的元素排序，再用折半查找法查找特定的Date对象day。

```
public class Test {
  public static void main(String args[]) {
    Date[] days = new Date[10];
    Date day = new Date(2004,4,6);
  }
}
class Date {
  int year, month, day;
  Date(int y, int m, int d) {
    year = y; month = m; day = d;
  }
  public int compare(Date date) {
    return year > date.year ? 1
           : year < date.year ? -1
           : month > date.month ? 1
           : month < date.month ? -1
           : day > date.day ? 1
           : day < date.day ? -1 : 0;
  }
}
```

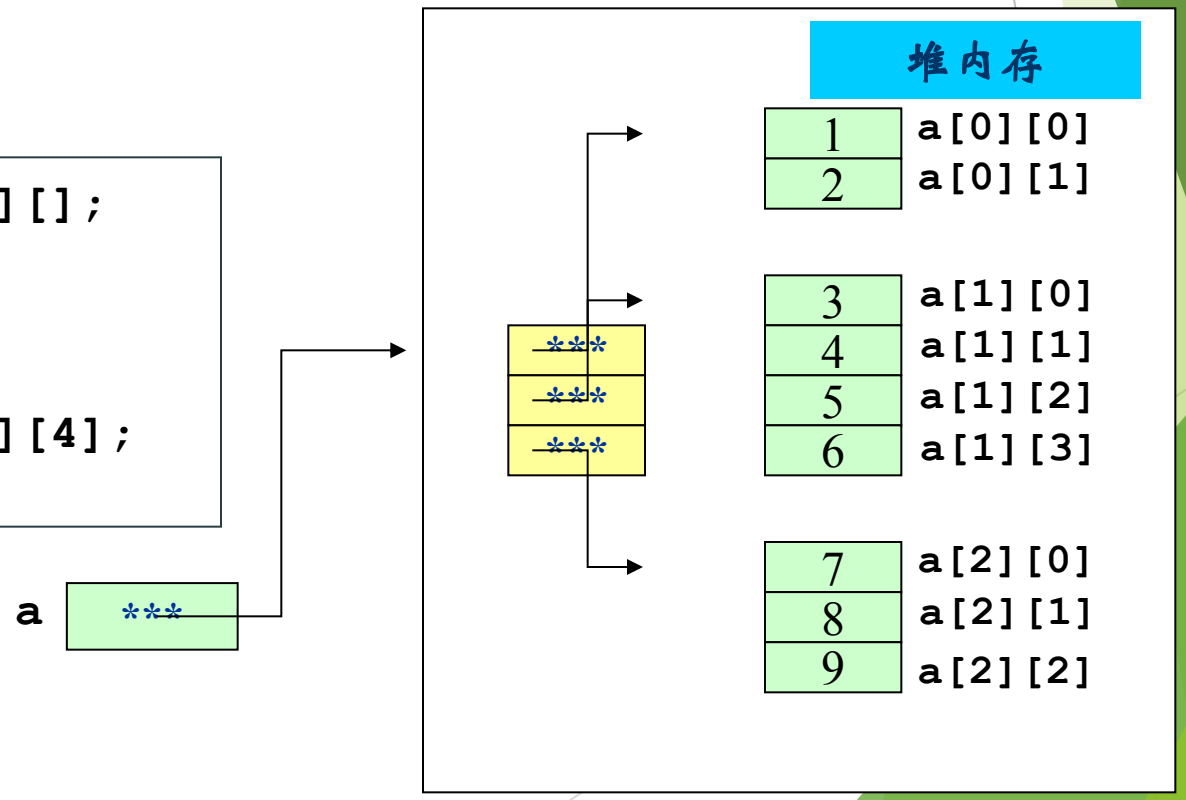
二维数组

- 二维数组可以看成以数组为元素的数组。例如：

int a[][] = {{1,2},{3,4,5,6},{7,8,9}};

- Java中多维数组的声明和初始化应按从高维到低维的顺序进行，例如：

```
int a[][] = new int[3][];
a[0] = new int[2];
a[1] = new int[4];
a[2] = new int[3]
int t1[][] = new int[][4];
//非法
```



二维数组初始化

- 静态初始化：

```
int intA[][] = {{1,2},{2,3},{3,4,5}};
int intB[3][2] = {{1,2},{2,3},{4,5}};//非法
```

- 动态初始化：

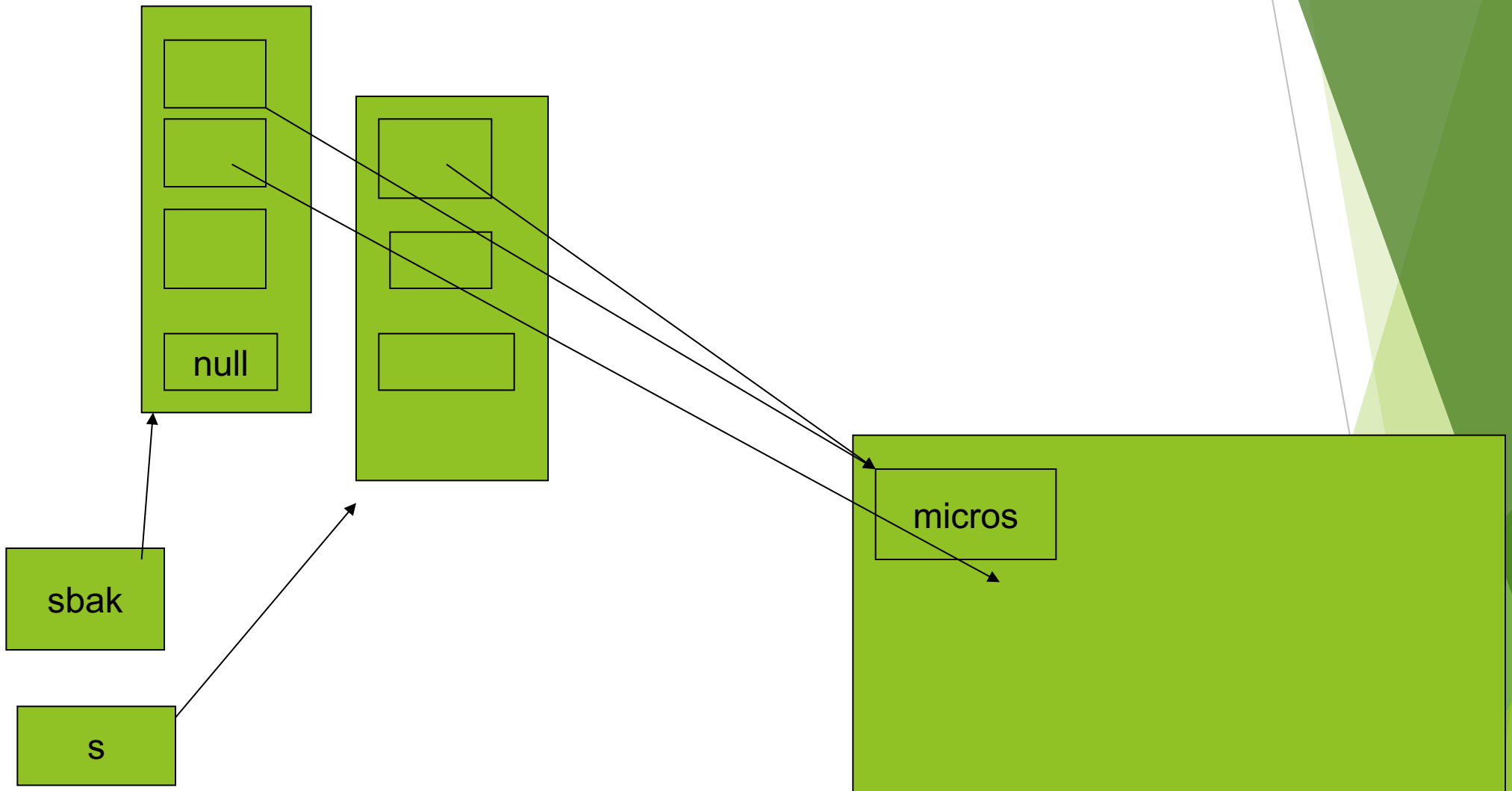
```
int a[][] = new int[3][5];

int b[][] = new int[3][]

b[0] = new int[2];

b[1] = new int[3];

b[2] = new int[5];
```

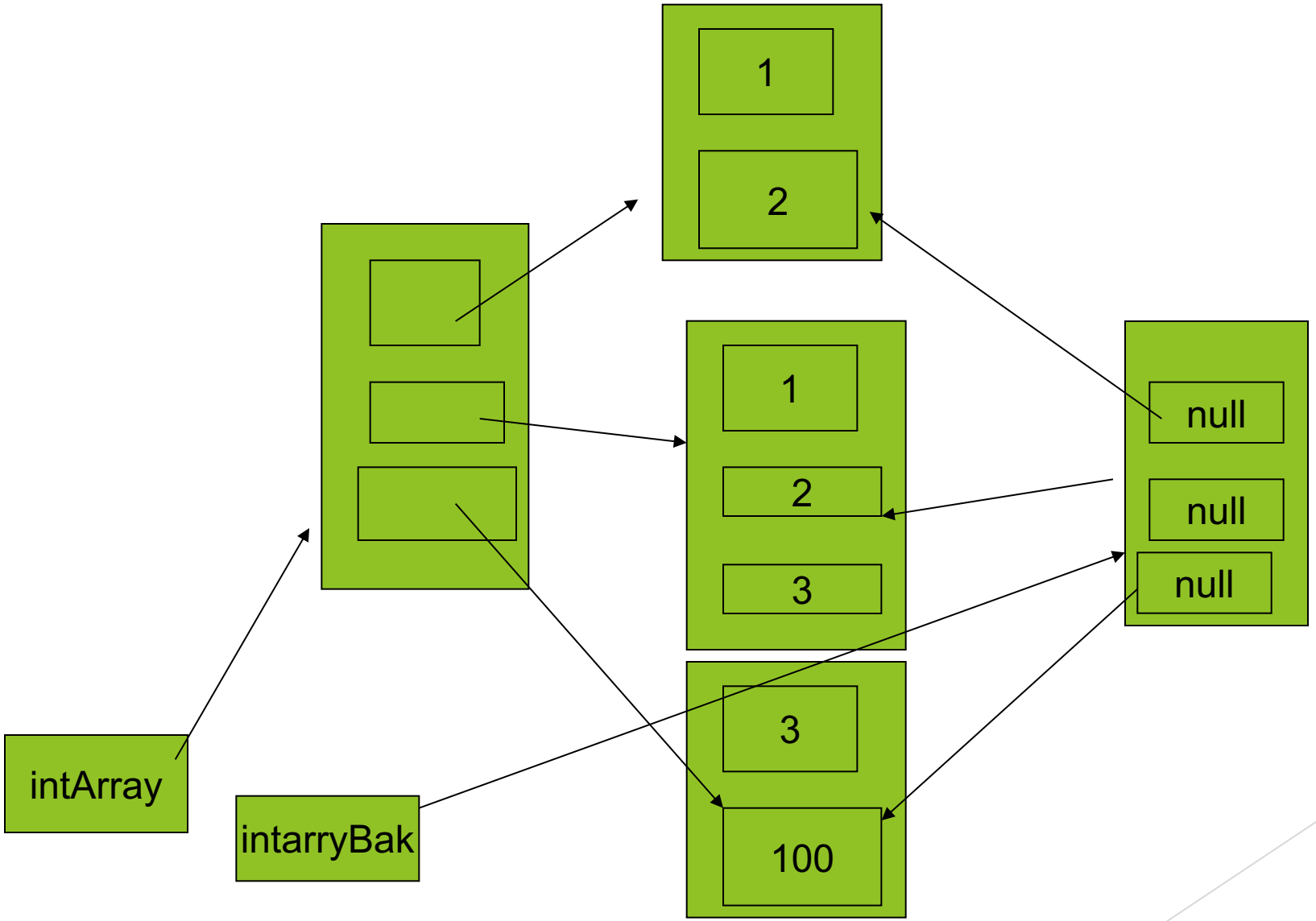
二维数组举例（1）

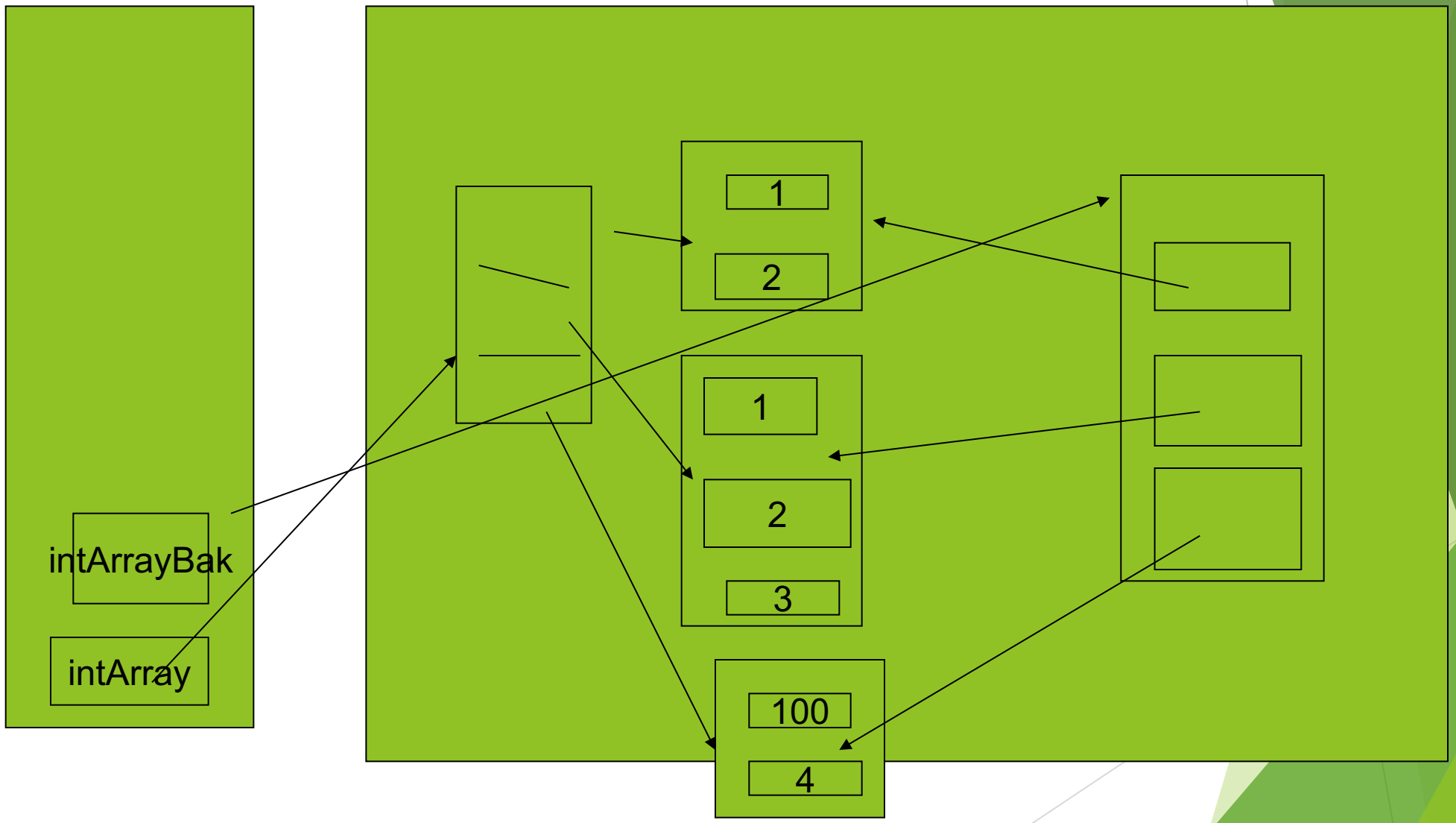
```
public class Test {
  public static void main(String args[]) {
    int a[][] = {{1,2},{3,4,5,6},{7,8,9}};
    for(int i=0;i<a.length;i++){
      for(int j=0;j<a[i].length;j++){
        System.out.print
            ("a["+i+"]["+j+"] = "+a[i][j]+"  ");
      }
      System.out.println();
    }
  }
}
```

二维数组举例（2）

```
public class Test {
  public static void main(String args[]) {
    String[][] s ;
    s = new String[3][];
    s[0] = new String[2];
    s[1] = new String[3];
    s[2] = new String[2];
    for(int i = 0;i<s.length;i++){
      for(int j = 0;j<s[i].length;j++){
        s[i][j] = new String
                  ("我的位置是："+i+","+j);
      }
    }
    for(int i = 0;i<s.length;i++){
      for(int j = 0;j<s[i].length;j++){
        System.out.print(s[i][j]+"  ");
      }
      System.out.println();
    }
  }
}
```


null
micros
sbak
s


1
2
1
2
3
null
null
null
3
100
intArray
intarryBak


intArrayBak
intArray
1
2
1
2
3
100
4

数组的拷贝

- 使用java.lang.System类的静态方法

```
public static void arraycopy
        (Object src,int srcPos,Object dest,
            int destPos,int length)
```

- 可以用于数组src从第srcPos项元素开始的length个元素拷贝到目标数组从destPos项开始的length个位置。
- 如果源数据数目超过目标数组边界会抛IndexOutOfBoundsException 异常。

TestArrayCopy.java