Java数组

今天我们学习一个Java中非常重要的技术——数组。

认识数组

先来认识一下什么数组

什么数组

数组是具有相同数据类型且按一定次序排列的一组变量的集合体。即用一个变量名表示一批数据。Java为数组在内存中分配一段连续的空间，这段空间中存储数据的个数是固定的，数组就是一个容器，用来存一批同种类型的数据的。

比如：想要存储 20,10,80,60,90 这些数据。 我们可以把代码写成这样

```
int[] array = {20,10,80,60,90}; // 5 0, 1, 2, 3, 4
```

比如：想要存储 “牛二“,“西门“,“全蛋“ 这些数据。我们可以把代码写成这样

```
String[] names = {"牛二", "西门", "全蛋"}; // 3  0, 1, 2
```

- 数组元素：构成数组的每一个元素
- 数组下标：下标就是数组元素在数组中的位置。下标从0开始，依次累加1，也称为索引。
- 数组大小：数组中元素的个数，也称为数组的长度。

数组的应用场景

有变量，为什么还要有数组呢？ 比如，我们要做一个点名器

如果用变量来做的话，代码是这样子的

```
String name1 = "张三";
String name2 = "李四";
String name3 = "王五";
```

```
String name4 = "赵六";
String name5 = "田七";
// ...
String name35 = "张十五";
String name36 = "张十六";
String name37 = "张十七";



Random rand = new Random();
int number = rand.nextInt(37) + 1; // 1-37
switch (number) {
    case 1:
        System.out.println(name1);
        break;
    case 2:
        System.out.println(name2);
        break;
    // ...
}
```

- 代码繁琐：大量变量的定义。
- 实现需求繁琐。

如果用数组来做的话，代码是这样子的

```
String[] names = {"张三", "李四", "王五", "赵六", "田七",.., "张十五", "张十六", "张十七"}; // 37 0-36
Random rand = new Random();
int number = rand.nextInt(37) + 1;
System.out.println(names[number - 1] + "出来回答问题");
```

- 代码简洁
- 逻辑清晰

一对比我们发现数组的写法比变量的写法更加简洁，所以我们可以得出一个结论

结论：遇到批量数据的存储和操作时，数组比变量更适合

数组的定义和访问

我们已经知道数组是用来干什么的。那么如何使用Java语言写一个数组呢？

定义数组

Java中定义数组有两种语法格式：`数据类型 数组名[ ];`或`数据类型[ ] 数组名;`

例如：`String names[]` 或 `String[] names` 推荐第二种方式

语法解析：

- 数组是什么数据类型，数组的元素就是什么数据类型
- 数组的特征是[ ]
- 数组是引用类型

数组有两种初始化的方式，一种是静态初始化、一种是动态初始化。我们先用静态初始化来学习数组的操作。

静态初始化数组

定义数组、为数组元素分配内存、数组元素初始化，这三步可以合并在一起写，例如:

`int[] scores = new int[]{12,56,34,78};`

或

`int[] scores = {12,56,34,78};`

在定义数组时直接给数组中的数据赋值这种方式称为静态初始化。标准格式是

```
数据类型[] 变量名 = new 数据类型[]{元素1,元素2,元素3};

简化为:
数据类型[] 变量名 = {元素1,元素2,元素3};

String[] names = {"金文涛", "李瑶瑶", "小樊同学"...};
System.out.println(names[1]);
```

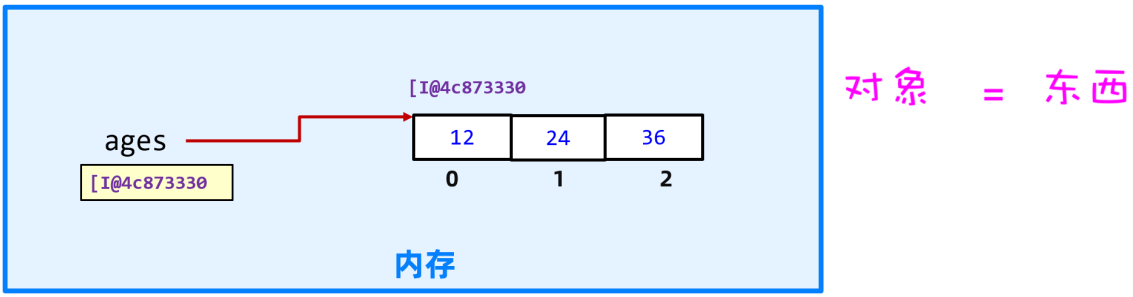
数组在计算机中的基本原理

我们知道数组是怎么定义的之后，那么接下来看一下数组在计算机中的基本原理。

我们以`int[] ages = {12,24,36};`这句话为例，看一下这句话到底在计算机中做了那些事情。

- 首先，左边`int[] ages`表示定义了一个数组类型的变量，变量名叫ages
- 其次，右边`{12,24,36}`表示创建一个数组对象，你完全可以把它理解成一个能装数据的东西。这个对象在内存中会有一个地址值`[I@4c873330`，每次创建一个数组对象都会有不用的地址值。
- 然后，把右边的地址值`[I@4c873330`赋值给左边的ages变量
- 所以，ages变量就可以通过地址值，找到数组这个东西。

```
int[] ages = {12, 24, 36};
```



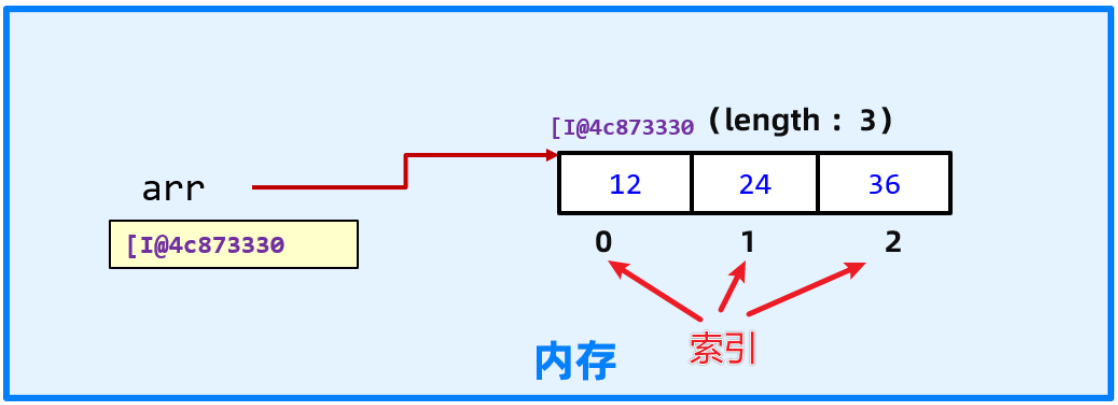
注意：数组变量名中存储的是数组在内存中的地址，数组是一种引用数据类型。

数组的元素访问

各位同学，通过刚才的学习，我们知道数组是用来存储数据的。那么数组中存储的数据又如何访问呢？这里所说的访问，意思就是获取中数组中数据的值、或者给数组中的数据赋值。

这里先给大家统一几个概念，数组中存储的数据我们叫做元素；而且数组中的每一个元素都有一个编号与之对应，我们把这个编号叫做索引，这个索引是从0依次递增的整数。如下图所示

```
int[] arr = {12, 24, 36};
```



要想访问数组中的元素，格式如下

```
// 数组名可以找到数组对象的地址，再通过索引就可以定位到具体的元素了
数组名[索引]    // 索引 0 --> 长度 - 1
```

接下来用代码来演示一下

```
// 索引:      0   1   2
int[] arr = {12, 24, 36}; // 静态初始化
// 1、访问数组的全部数据
System.out.println(arr[0]); // 12
System.out.println(arr[1]); // 24
System.out.println(arr[2]); // 36
// 下面代码没有3索引，会出现ArrayIndexOutOfBoundsException 索引越界异常
// System.out.println(arr[3]);

// 2、修改数组中的数据
arr[0] = 66;
arr[2] = 100;
System.out.println(arr[0]); //66
System.out.println(arr[1]); 0
System.out.println(arr[2]); //100
```

除了访问数组中的元素，我们可以获取数组中元素的个数，后面我们统称为数组的长度。

```
// 3、访问数组的元素个数：数组名.length
System.out.println(arr.length);

// 技巧：获取数组的最大索引：arr.length - 1(前提是数组中存在数据)
System.out.println(arr.length - 1);

int[] arr2 = {};
// arr2[arr2.length - 1]
System.out.println(arr2.length - 1);
```

数组的遍历

各位同学，接下来我们学习一个对数组最最最常见的操作——数组遍历。所谓遍历意思就是将数组中的元素一个一个的取出来。

我们刚才学习了数组中元素的访问，访问元素必须用到索引，如下列代码。

```java
int[] ages = {12, 24, 36};
System.out.println(ages[0]);
System.out.println(ages[1]);
System.out.println(ages[2]);
```

但是，如果数组中有很多很多元素，索引靠自己一个一个数肯定是不行的！我们可以使用for循环从0开始一直遍历到长度-1的位置，就可以获取所有的索引了。

当你获取到每一个索引，那么每一个元素不就获取到了吗？上代码吧

```java
int[] ages = {12, 24, 36};
for (int i = 0; i < ages.length; i++) {
    // i的取值 = 0, 1, 2
    System.out.println(ages[i]);
}
```

数组静态初始化案例

学习完数组的静态初始化之后，接下来我们做一个练习题来巩固一下。

```
需求：某部门5名员工的销售额分别是：16、26、36、6、100，请计算出他们部门的总销售额。

需求分析：
    1.看到有16、26、36、6、100这5个数据数据，而且数据值很明确；
        1)想到,可以使用数组静态初始化把这5个数据存起来

    2.请计算出他们部门的总销售额（这不就是求数组中数据的和吗？）
        2)必须先将数组中所有的元素遍历出来
        3)想要求和，得先有一个求和变量sum
        4)再将每一个元素和求和变量sum进行累加（求和思想）
```

按照分析的思路来写代码

```java
// 1、定义一个数组存储5名员工的销售额
//索引          0   1    2  3   4
int[] money = {16, 26, 36, 6, 100};

// 3、定义一个变量用于累加求和
int sum = 0;

// 2、遍历这个数组中的每个数据。
for (int i = 0; i < money.length; i++) {
    // i = 0  1  2  3  4
    sum += money[i];
}
System.out.println("员工的销售总额: " + sum);
```

数组的动态初始化

刚才我们初始化数组时，都是直接将元素写出来。但是还有另一个初始化数组的方式叫 **动态初始化**。

动态初始化不需要我们写出具体的元素，而是指定元素类型和长度就行。格式如下

数据类型[] 数组名;

`数组名 = new 数据类型[数组长度];`

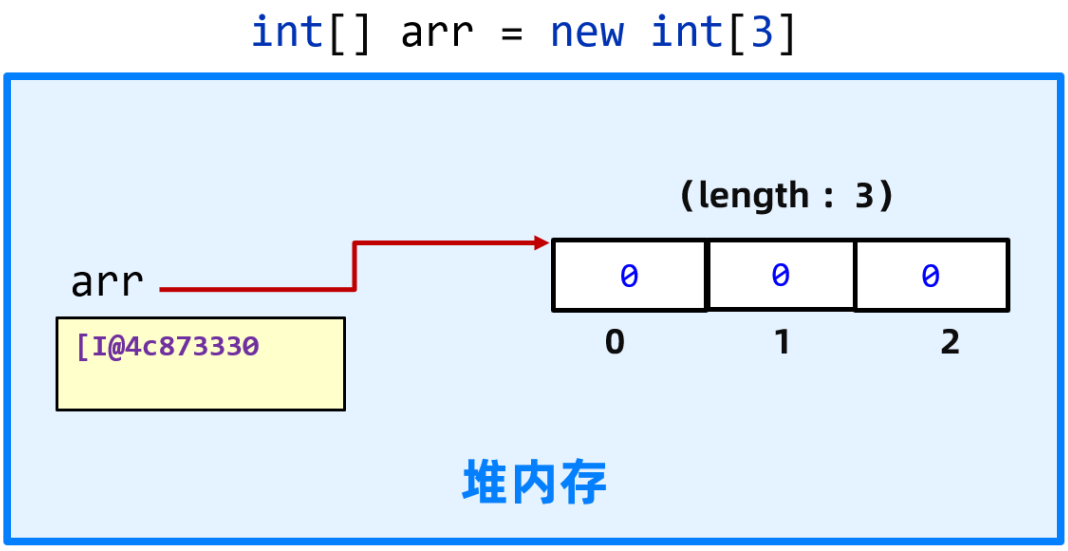
例如：`names = new String[5];`

定义数组和为数组元素分配内存，这两步可以合并在一起写，例如：

`String[] names = new String[5];`

```java
// 数据类型[]  数组名 = new 数据类型[长度];
int[] arr = new int[3];
```

下面是动态初始化数组的原理图。我们发现`int[] arr`其实就是一个变量，它记录了数组对象的地址值，而且数组中的元素默认值是0。



注意：

使用动态初始化定义数组时，根据元素类型不同，默认值也有所不同。

数组元素类型	默认初始值
byte，short，int，long	0
float，double	0.0
char	'\u0000'（空字符）
boolean	false
引用数据类型	null

数组动态初始化案例

各位同学，接下来我们做一个数组动态初始化的案例。

```
案例需求：
    某歌唱比赛，需要开发一个系统：可以录入6名评委的打分，录入完毕后立即输出平均分做
    选手得分

需求分析：
    1.需要录入6名评委的分数，可以用一个数组来保存。
        因为在评委没有录入分数之前，还不确定数组中应该存哪些数据。
        所以可以使用数组的动态初始化
    2.遍历数组中的每一个位置，并录入分数，将分数存入数组中
    3.遍历数组中的每一个元素，对元素求和
public static void main(String[] args) {
```

```
    // 某歌唱比赛，需要开发一个系统：可以录入6名评委的打分，录入完毕后立即输出平均分做选手得分
    double[] scores = new double[6];
    Scanner sc = new Scanner(System.in);
    for (int i = 1; i <= 6; i ++) {
        System.out.println("请输入第" + i + "个评委的成绩：");
        double score = sc.nextDouble();
        scores[i - 1] = score;
    }
    double sum = 0;
    double avg = 0;
    for (int i = 0; i < scores.length; i++) {
        System.out.println(scores[i]);
        sum += scores[i];
    }
    avg = sum / scores.length;
    System.out.println("选手的成绩是" + avg);
}
```

使用数组实现斐波拉切数列

```
public static void main(String[] args) {
    // 输入 n 输出前 n 个数 存储到数组 再输出
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入一个数字");
    int number = sc.nextInt();
    int[] feibo = new int[number];
    // int prev = 1; // 第一个值
    // int next = 1; // 第二个值
    // feibo[0] = prev;
    // feibo[1] = next;
    if (number == 1) {
        feibo[0] = 1;
    } else if (number == 2) {
        feibo[0] = 1;
        feibo[1] = 1;
    } else {
        // feibo : 1 1 2 3
        feibo[0] = 1;
        feibo[1] = 1;    // 1 1 x
        for (int i = 3; i <= number; i++) { // i = 3 i = 4
            feibo[i - 1] = feibo[i - 3] + feibo[i - 2];
            // int curr = prev + next;
            // feibo[i - 1] = curr;
            // prev = next;
            // next = curr;
```

```
        }
    }
    for (int i = 0; i < feibo.length; i++) {
        System.out.println(feibo[i]);
    }
}
```

数组使用过程可能出现的问题

```
public class ArrayDemo03{

    public static void main(String[] args){


        int[] scores = {32,45,45,76};

        System.out.println(scores[5]); // 下标越界

        int[] ages = {32,43,444,32,'a'};
        System.out.println(ages[4]);  // 97  报错

        int[] ages1 = {32,43,444,32L};
        ages1[2] = 100;
        ages[2] = 100L;
        System.out.println(ages1[3]); // 报错 32

    }
}
```

- 如果在数组中保存的元素可以自动提升(自动类型转化)为数组自己的类型，那是可以保存的
- 数组下标越界

数组在计算机中的执行原理

好的各位同学，在前面我们已经学习了数组的基本使用，也理解了数组的基本原理。由于数组是一个容器，变量也是一个容器，在理解他们执行原理的时候，有些同学就容易搞混，现在我把他们放在一起带着大家回顾一下他们的会执行原理，顺便带着大家详细理解一下Java程序的执行的内存原理。

数组的执行原理，Java程序的执行原理

我们以下面的代码，来讲解变量、数组的执原理。

```
public class ArrayDemo1 {
    public static void main(String[] args) {
        int a = 10;
        System.out.println(a);  // 10

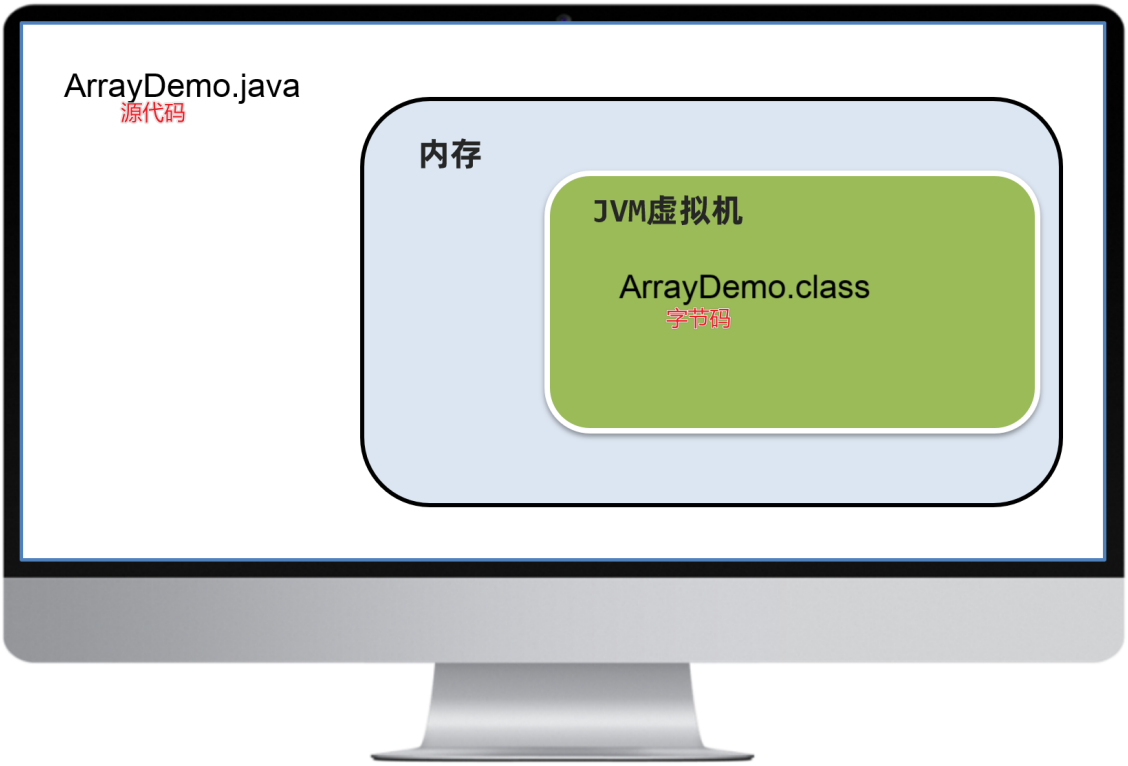
        int[] arr = new int[]{11, 22, 33}; // 静态初始化
        System.out.println(arr);  // 地址

        System.out.println(arr[1]); // 22

        arr[0] = 44;
        arr[1] = 55;
        arr[2] = 66;

        System.out.println(arr[0]); // 44
        System.out.println(arr[1]); // 55
        System.out.println(arr[2]); // 66
    }
}
```

前面我们给大家讲过，程序是在内存中执行的。实际上Java程序是把编译后的字节码加载到Java虚拟机中执行的。

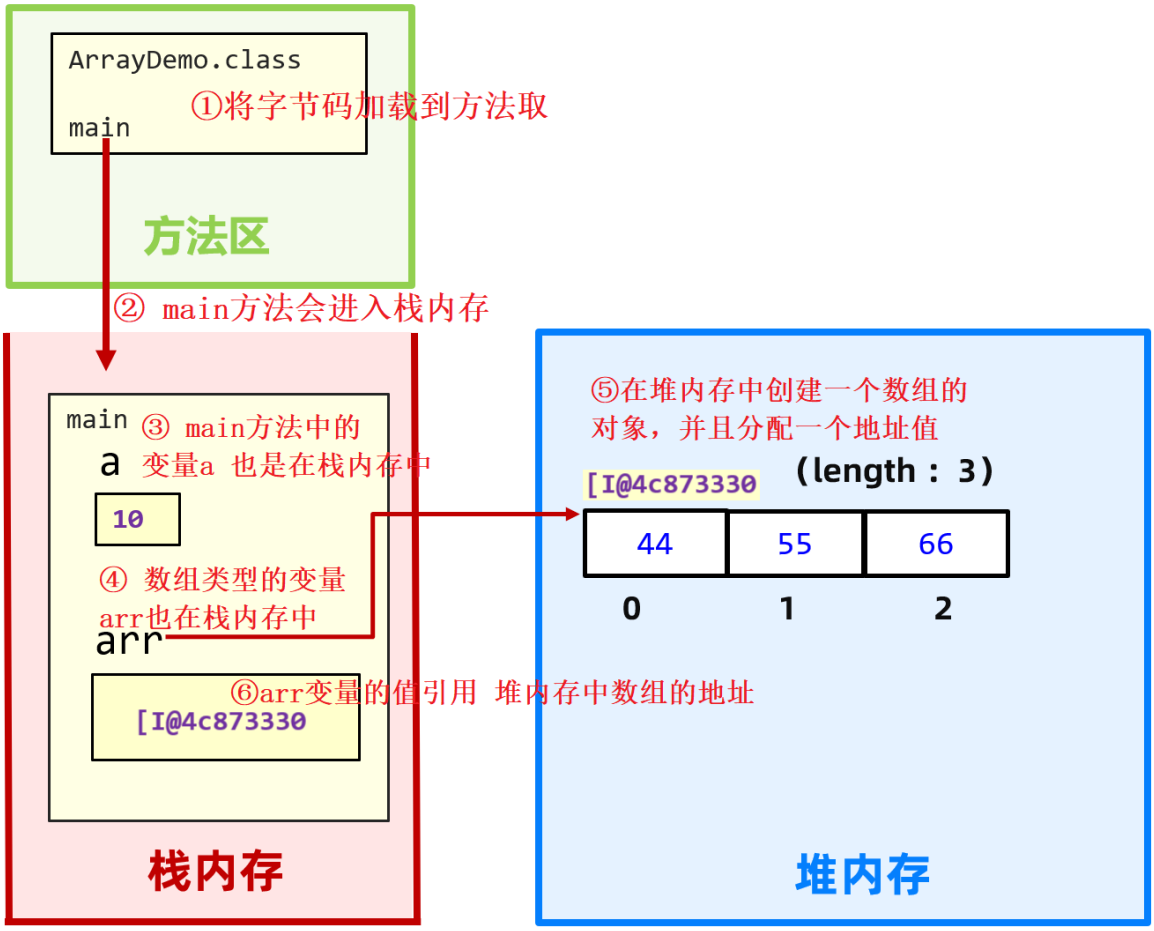


Java为了便于虚拟机执行Java程序，将虚拟机的内存划分为 方法区、栈、堆、本地方法栈、寄存器 这5块区域。同学们需要重点关注的是 **方法区、栈、堆**。

下面把每一个块内存区域作用介绍一下，我们大致只需要知道每一部分存储什么内容就行。

- **方法区**：字节码文件先加载到这里
- ```
  Random rand = new Random();
  Scanner sc = new Scanner(System.in);
  ```
-
- **栈**：方法运行时所进入的内存区域，由于变量在方法中，所以变量也在这一块区域中
- **堆**：存储new出来的东西，并分配地址。由于数组是new 出来的，所以数组也在这块区域。

下面是上面案例执行的内存原理如下图所示，按照① ② ③ ④ ⑤ ⑥ 的标记的顺序来看
  ```



总结一下`int a = 10`与`int[] arr = new int[]{11,22,33}`**的区别**

- **a**是一个变量，在栈内存中，**a**变量中存储的数据就是**10**这个值。
- **arr**也是一个变量，在栈中，存储的是数组对象在堆内存中的地址值

```
// 这里的int a是一个基本类型变量，存储的是一个数值
int a = 10 ;
//这里的int[] arr是一个引用类型的变量，存储的是一个地址值
int[] arr = new int[]{44,55,66};
```

多个变量指向同一个数组的问题

各位同学，我们了解了数组在内存中的执行原理。我们知道数组类型的变量，指向的是堆内存中数组对象的地址。但是在实际开发中可能存在一种特殊情况，就是多个变量指向同一个数组对象的形式。

讲解这个知识点的目的，是让同学们注意多个变量指向同一个数组对象存在什么问题？

我们先看一段代码

```
public class ArrayDemo2 {
    public static void main(String[] args) {
```

```
        // 目标：认识多个变量指向同一个数组对象的形式，并掌握其注意事项。
        int[] arr1 = new int[]{11, 22, 33};

        // 把int类型的数组变量arr1赋值给int类型的数组变量arr2
        int[] arr2 = arr1;
        int[] arr3 = new int[]{11, 22, 33};

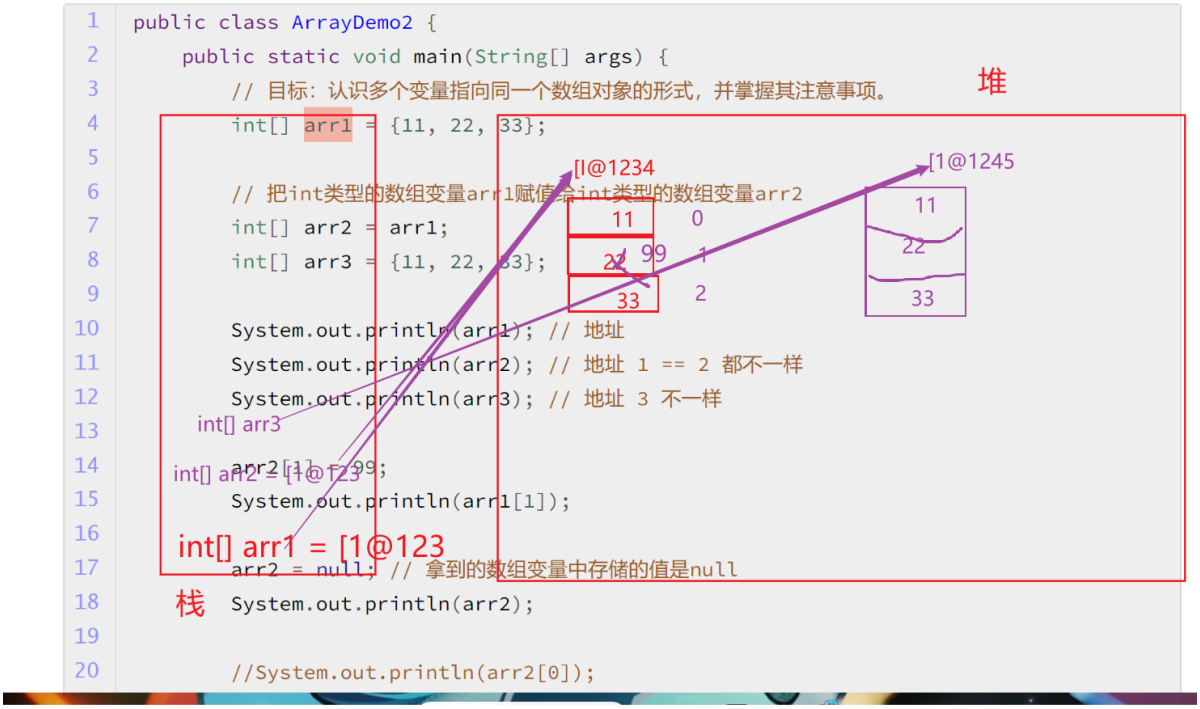
        System.out.println(arr1); // 地址
        System.out.println(arr2); // 地址 1 == 2 都不一样
        System.out.println(arr3); // 地址 3 不一样

        arr2[1] = 99;
        System.out.println(arr1[1]);

        arr2 = null; // 拿到的数组变量中存储的值是null
        System.out.println(arr2);

        //System.out.println(arr2[0]);
        //System.out.println(arr2.length);
    }
}
```

我们先看一段代码



我们重点关注这一段代码

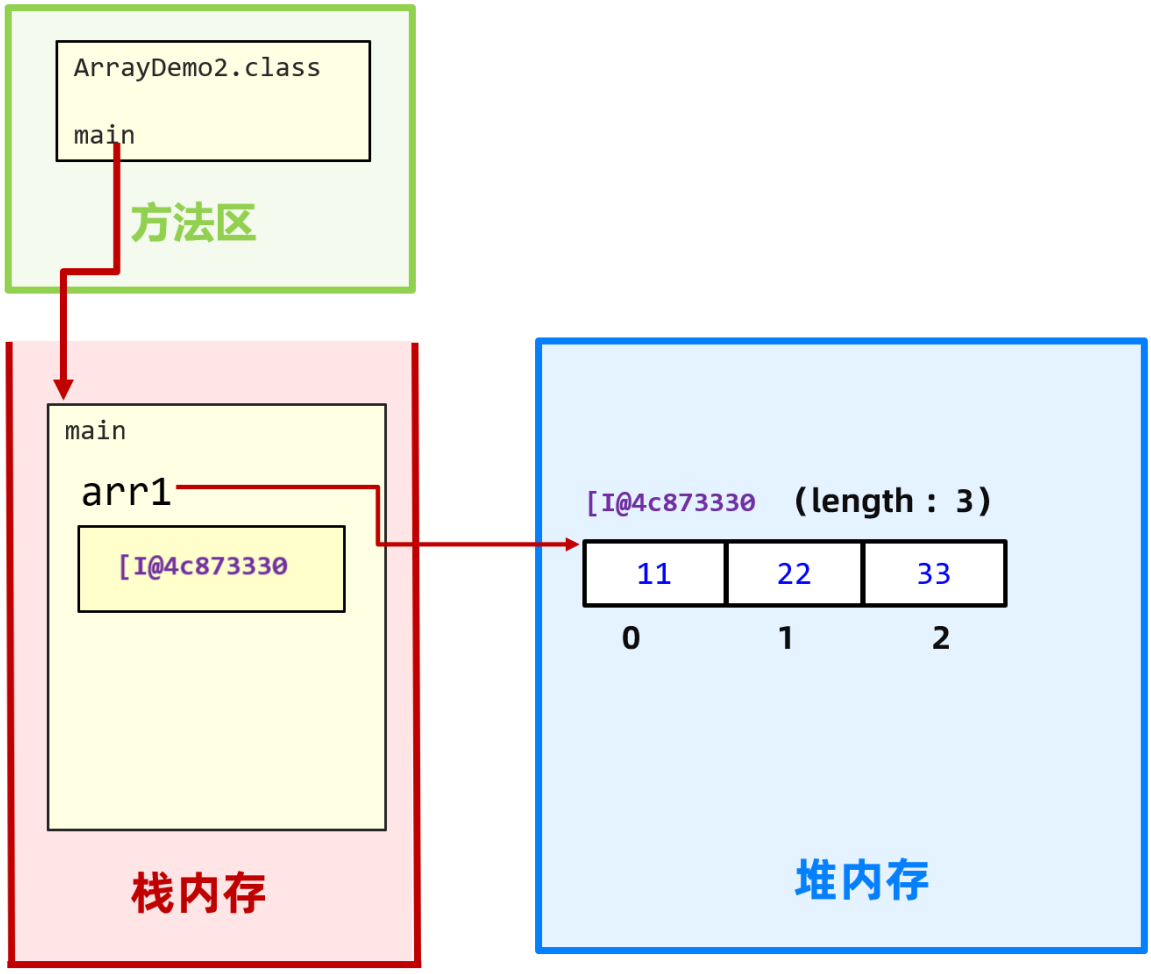
这里arr1记录的是数组的地址值

```
int[] arr1 = {11, 22, 33};

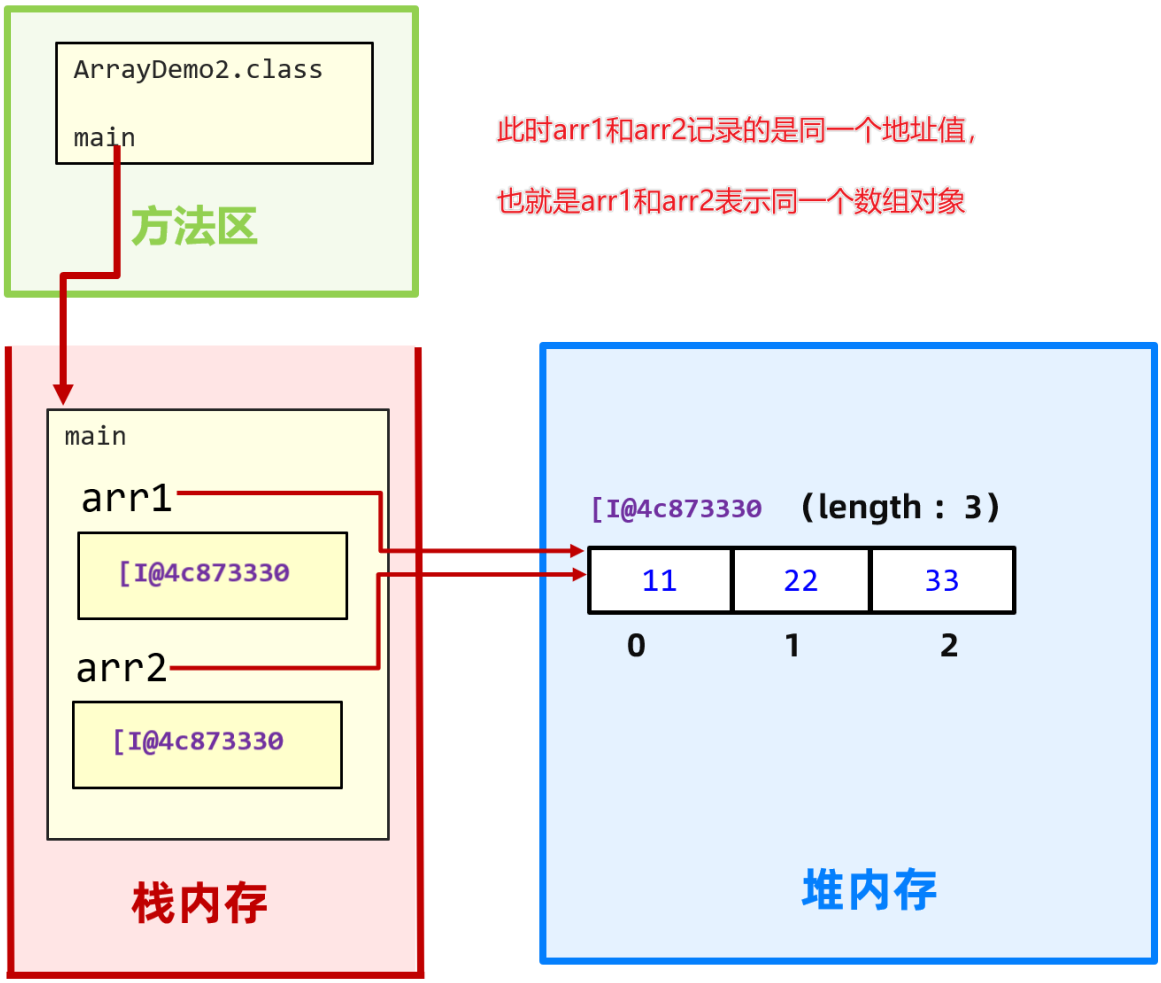
// 把int类型的数组变量arr1赋值给int类型的数组变量arr2
int[] arr2 = arr1; 把arr1记录的地址，再赋值给arr2
```

此时：arr1和arr2就是同一个地址值

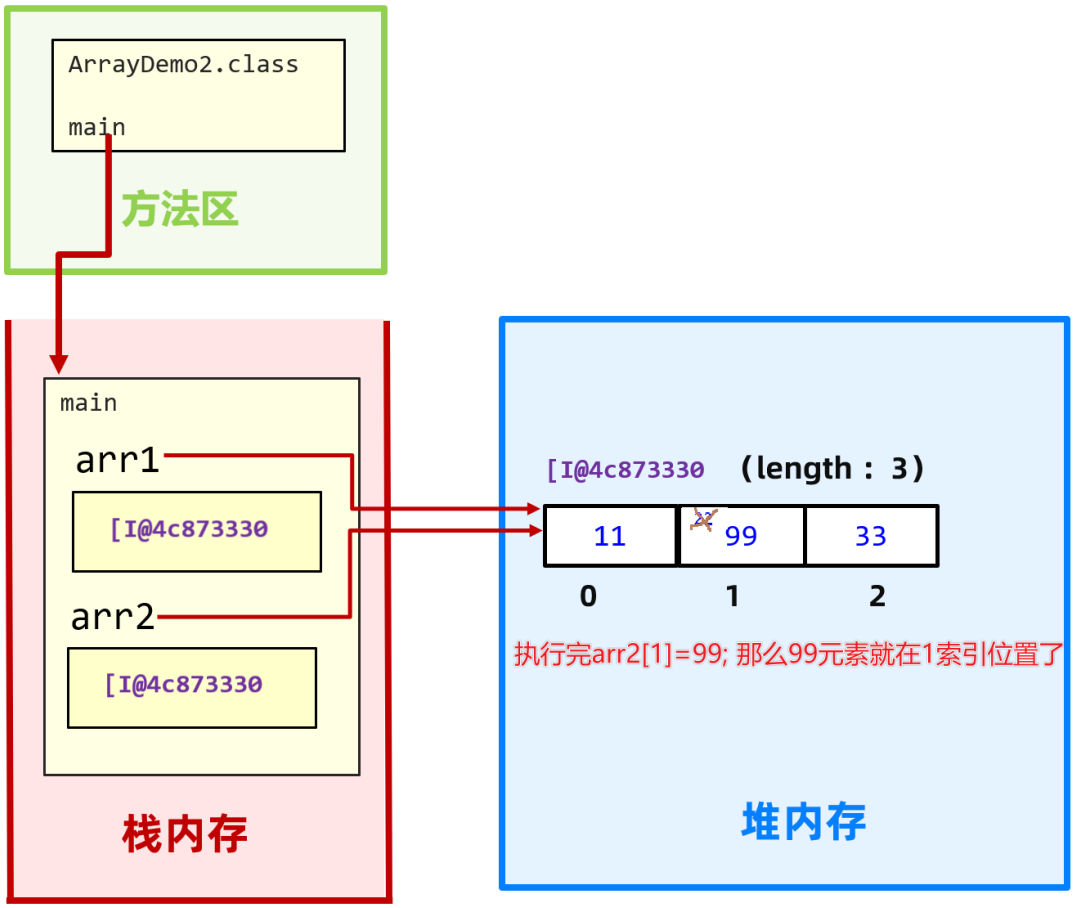
刚执行完`int[] arr1 = {11,22,33};`时，内存原理如下



当执行完`int[] arr2 = arr1;`后，内存原理如下



当执行到`arr2[1]=99;时`，内存原理如下



总结一下：

- 两个变量指向同一个数组时，两个变量记录的是同一个地址值。
- 当一个变量修改数组中的元素时，另一个变量去访问数组中的元素，元素已经被修改过了。

到这里有关数组的基本操作，和内存原理我们就全部学习完了。

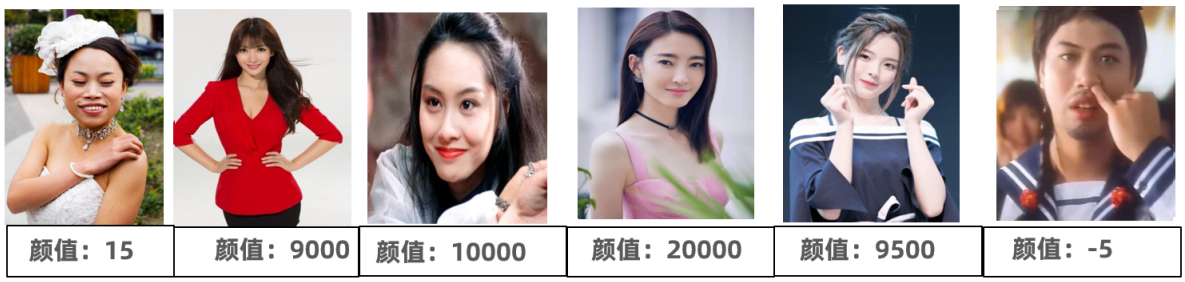
数组专项练习

接下来我们做一些专项练习题，把数组的常见操作练习一下。在学习这个案例时，重点掌握数组求最值的思路，代码只是用来表达你的思路的。

数组求最值

```
需求：定义一个int类型数组，求数组中元素的最大值，并打印最大值
```

我们先看一下选美比赛，是怎么选出颜值最高的人的。然后再以此思路，来写代码找出数组中元素的最大值。



① 首先，准备一个擂台，凤姐带着15的颜值先上台
② 接着，后面的每一个人，依次上台和擂台上的主角进行比较
③ 然后，每次比较，将颜值胜出的人留在擂台上。
④ 等每一个元素都比较完了，留在擂台上的就是颜值最高的

```
数组求最大值思路：
    1)先找出数组中0索引的元素，假设为最大值，用max表示【擂主】
    2)遍历后面的每一个元素和max比较，把较大的元素值重新赋值给max(擂主换人)
    3)最后max就是所有元素的最大值(最后站在台上的擂主)
```

总结一下：

通过这个案例，我们主要掌握求最值的思路，以后不管遇到求最大值还是最小值，编程思路都是一样的，不同的可能是数据不同。

课堂练习：

```
// 求最小值
```

使用增强for循环遍历数组

JDK1.5及其之后的版本中提供了增强for循环语句，实现了Iterable接口的类都可以使用增强for循环进行元素的迭代。增强for循环的语法规则如下：

```
for (元素类型 变量名 : 要迭代的对象) {
    System.out.println(变量名);
}

int[] arr = {15, 9000, 10000, 20000, 9500, -5};
for (int e : arr) {
    System.out.printn(e);
}

for (int i = 0; i < arr.length; i ++) {
    System.out.printn(arr[i]);
}

for (;;) {

}
```

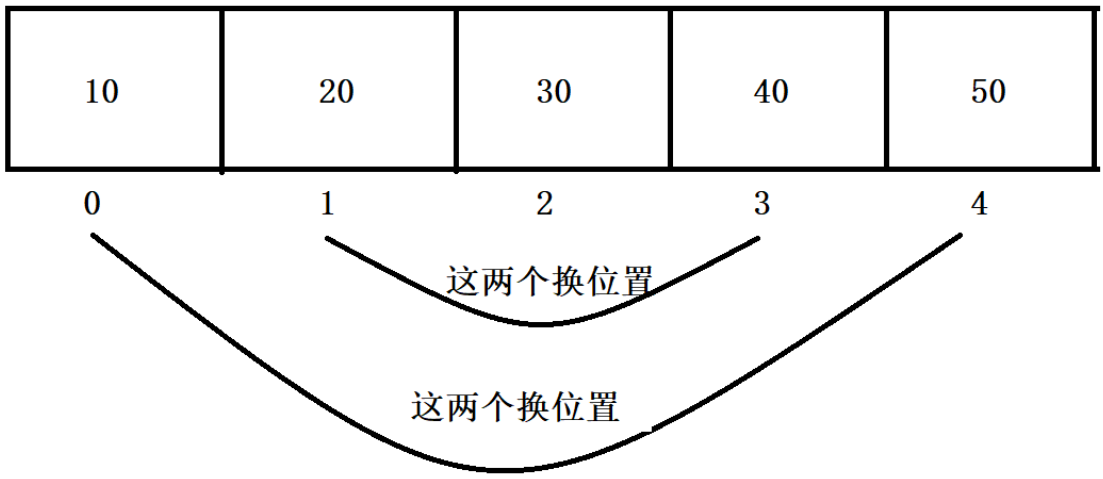
语法解析：

- 元素类型是指数组或集合中的元素的类型。
- 变量名在循环时用来保存每个元素的值。
- 冒号后面是要遍历的数组或集合的名称。

数组元素反转

```
需求：某个数组有5个数据：10,20,30,40,50，请将这个数组中的数据进行反转。
      [10, 20, 30, 40, 50]  反转后 [50, 40, 30, 20, 10]
```

怎么样，才能达到元素反转的效果呢？我们只需将第一个和最后一个元素互换、第二个和倒数第二个互换、依次内推.... 如下图所示



怎么样写代码，才能达到上面的效果呢？我们继续分析

```
1.每次交换，需要有左右两边的两个索引，我们可以用i和j表示
    刚开始i=0，j=数组长度-1;
2.每次让i和j索引位置的两个元素互换位置
    arr[i]和arr[j]互换位置
3.每次还完位置之后，让i往右移动一位，让j往前移动一位
```

具体代码如下

```
public static void main(String[] args) {
    //  现有一个 int 数组，数组中有十个元素。将数组反转后输出。
    int[] arr = new int[]{9, 1, 3, 4, 54, 56, 23, 22, 20, 43};
    //  for (int i = arr.length - 1; i >= 0; i--) {
    //      System.out.println(arr[i]);
    //  }
    for (int i = 0, j = arr.length - 1; i < arr.length / 2; i++, j--) {
        // int temp = arr[i];
        // arr[i] = arr[j];
        // arr[j] = temp;
        arr[i] = arr[i] ^ arr[j]; // 1100 0000 1100   1100  0000
        arr[j] = arr[i] ^ arr[j]; // arr[i] ^ arr[j] ^ arr[j] = arr[i] ^ 0 = arr[i]
        arr[i] = arr[i] ^ arr[j]; // arr[i] ^ arr[j] ^ arr[i] = 0 ^ arr[j] = arr[j]
```

```
14      }
15      for (int a : arr) {
16          System.out.println(a);
17      }
18
19  }
```

总结一下：

通过上面的案例，需要我们掌握元素互换位置的编程思路；以后遇到数据互换问题，都这样做。

排序

排序算法有多种，常用的排序算法有冒泡排序、插入排序、选择排序、快速排序、堆排序、归并排序、希尔排序、二叉树排序、计数排序等。

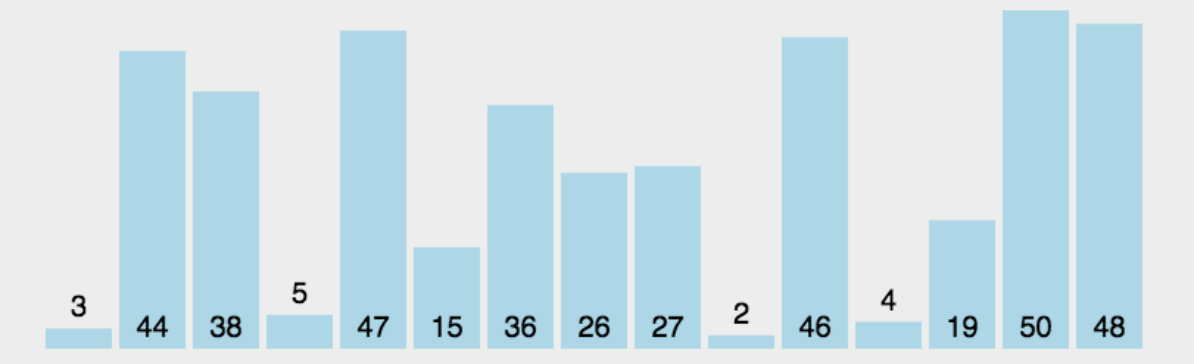
选择排序

表现**最稳定的排序算法之一**，因为**无论什么数据进去都是O(n2)的时间复杂度**，所以用到它的时候，数据规模越小越好。唯一的好处可能就是不占用额外的内存空间了吧。理论上讲，选择排序可能也是平时排序一般人想到的最多的排序方法了吧。

选择排序(Selection-sort)是一种简单直观的排序算法。它的工作原理：首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。

n个记录的直接选择排序可经过n-1趟直接选择排序得到有序结果。具体算法描述如下：

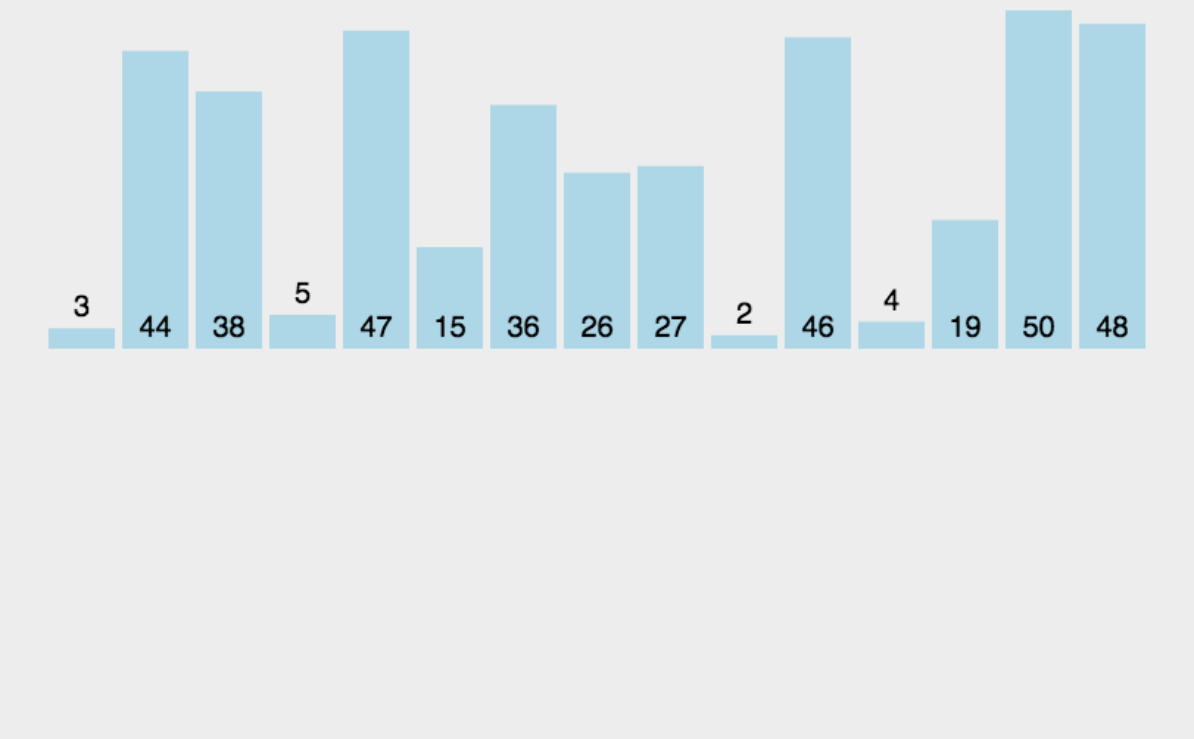
- 初始状态：无序区为R[1..n]，有序区为空；
- 第i趟排序(i=1,2,3…n-1)开始时，当前有序区和无序区分别为R[1..i-1]和R(i..n）。该趟排序从当前无序区中-选出关键字最小的记录 R[k]，将它与无序区的第1个记录R交换，使R[1..i]和R[i+1..n)分别变为记录个数增加1个的新有序区和记录个数减少1个的新无序区；
- n-1趟结束，数组有序化了。



插入排序

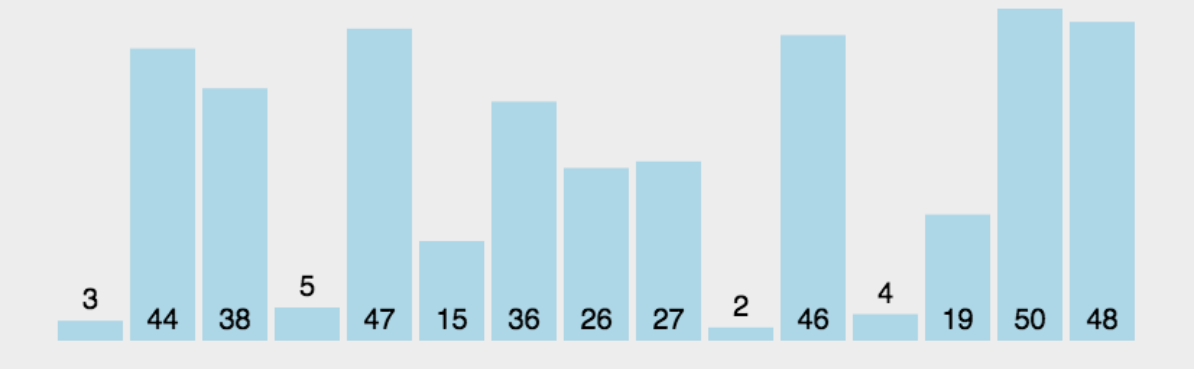
插入排序的算法描述是一种简单直观的排序算法。它的工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。插入排序在实现上，需要反复把已排序元素逐步向后挪位，为最新元素提供插入空间。算法描述:

- 从第一个元素开始，该元素可以认为已经被排序；
- 取出下一个元素，在已经排序的元素序列中从后向前扫描；
- 如果该元素（已排序）大于新元素，将该元素移到下一位置；
- 重复步骤3，直到找到已排序的元素小于或者等于新元素的位置；
- 将新元素插入到该位置后；
- 重复步骤2~5。



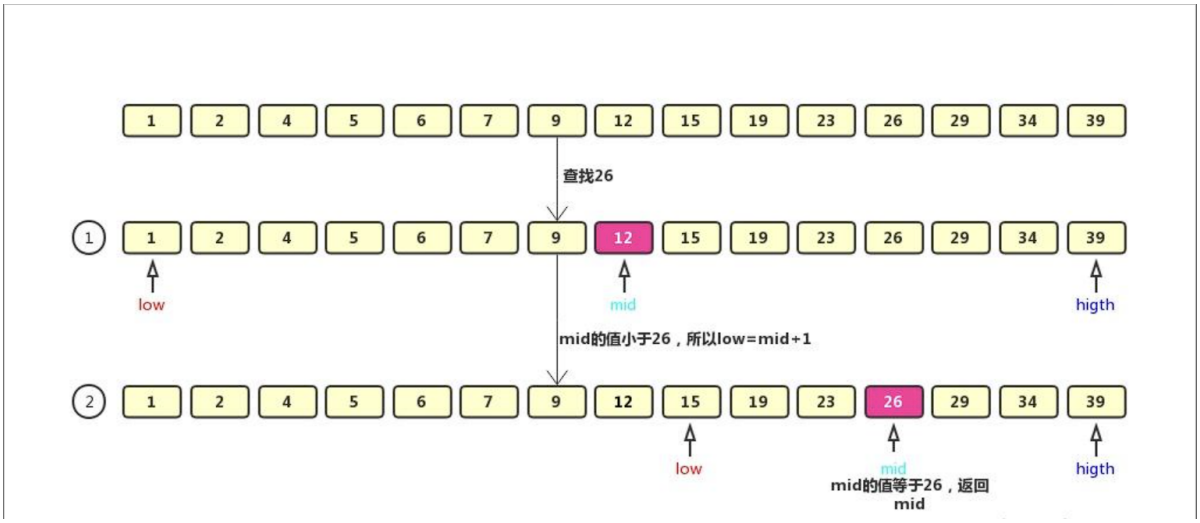
冒泡排序

它重复地走访过要排序的数组，一次比较两个元素，如果它们的顺序错误就把它们交换过来。走访数组的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。



二分查找

二分查找（Binary Search）算法，也叫折半查找算法。二分查找的思想非常简单，有点类似分治的思想。二分查找针对的是一个有序的数据集合，每次都通过跟区间的中间元素对比，将待查找的区间缩小为之前的一半，直到找到要查找的元素，或者区间被缩小为 0。



随机排名

各位同学，通过数组元素反转的案例，我们学会了如何对两个数据进行交换。接下来，我们再学习随机排名案例，将数据交换的思路再巩固一下。

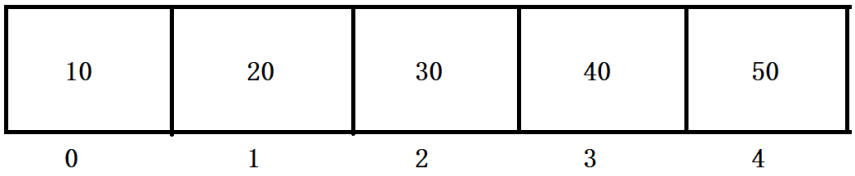
先来看一下需求

```
需求：某公司开发部5名开发人员，要进行项目进展汇报演讲，现在采取随机排名后进行汇报。请先依次录入5名员工的工号，然后展示出一组随机的排名顺序。
```

分析一下随机排名的思路

```
1.在程序中录入5名员工的工号存储起来 ---> 使用动态初始化数组的方式。
2.依次遍历数组中的每个数据。
3.每遍历到一个数据，都随机一个索引值出来，让当前数据与该索引位置处的数据进行交换。
```

如下图所示，每次遍历到一个元素，随机将当前位置元素和随机索引元素换位置。



核心思路：遍历过程中，每获取一个元素，同时随机产生一个索引，让当前索引位置的元素和随机索引位置的元素互换

当前索引：i = 0;
随机索引：index = 3;
arr[i]和arr[index]换位置

当前索引：i = 1;
随机索引：index = 2
arr[i]和arr[index]换位置

代码如下

```java
public class Test3 {
    public static void main(String[] args) {
        // 目标：完成随机排名
        // 1、定义一个动态初始化的数组用于存储5名员工的工号
        int[] codes = new int[5];

        // 2、提示用户录入5名员工的工号。
        Scanner sc = new Scanner(System.in);
        for (int i = 0; i < codes.length; i++) {
            // i = 0 1 2 3 4
            System.out.println("请您输入第" + (i + 1) +"个员工的工号：");
            int code = sc.nextInt();
            codes[i] = code;
        }

        // 3、打乱数组中的元素顺序。
        // [12, 33, 54, 26, 8]
        //  i       index
```

```
        Random r =  new Random();
        for (int i = 0; i < codes.length; i++) {
            // codes[i]
            // 每遍历到一个数据，都随机一个数组索引范围内的值。
            //然后让当前遍历的数据与该索引位置处的值交换。
            int index = r.nextInt(codes.length); // 0 - 4
            // 定义一个临时变量记住index位置处的值
            int temp = codes[index];
            // 把i位置处的值赋值给index位置处
            codes[index] = codes[i];
            // 把index位置原来的值赋值给i位置处
            codes[i] = temp;
        }

        // 4、遍历数组中的工号输出即可
        for (int i = 0; i < codes.length; i++) {
            System.out.print(codes[i] + " ");
        }
    }
}
```

多维数组

在 Java 中，可以使用多维数组来存储多个数据值，以便更好地组织和访问这些数据。Java 中的多维数组是一种数组的数组，即一个数组的元素也是一个数组。Java 中的多维数组可以包含任意数量的维度。在处理多维数组时，需要注意数组下标的范围，以避免数组越界异常。同时，还可以使用循环嵌套来遍历多维数组中的所有元素。

二维数组

Java中定义和操作多维数组的语法与一维数组类似。在实际应用中，三维及其以上的数组使用很少，主要使用二维数组。使用二维数组同一维数组的步骤，（1）定义数组、（2）为数组元素分配内存、（3）数组元素初始化、（4）使用数组。下面主要以二维数组为例进行讲解。

H5 定义二维数组

定义二维数组的语法规则如下：

`数据类型[ ][ ] 数组名;`

或者

`数据类型 数组名[ ][ ];`

语法解析：

`[][]`表示二维数组，前面的[ ]表示第一维，后面的[ ]表示第二维。

`[][]`放在数组名的前面或后面都是正确的。

H5 分配内存

```
int[][] arr = new int[3][4];
```

H5 数组元素初始化

```
// 二维数组初始化
int[][] arr = new int[3][4]; // 动态初始化
arr[0][0] = 1;
int[][] arr1 = new int[][]{  // 静态初始化
    {1, 2, 3},
    {2, 3},
    {3, 4, 5, 4}
};
```

H5 二维数组的迭代

```
for (int i = 0; i < arr1.length; i++) {
    System.out.println(arr1[i]);
    for (int i1 = 0; i1 < arr1[i].length; i1++) {
        System.out.println(arr1[i][i1]);
    }
}

for (int[] t : arr1) {
    for (int a : t) {
        System.out.println(a);
    }
}
```

练习

```
// 需求：打印杨辉三角形(行数可以键盘录入)

//  1             0
//  1 1           1
```

```
//  1 2 1          2
//  1 3 3 1        3
//  1 4 6 4 1      4
//  1 5 10 10 5 1  5
// 分析：看这种图像的规律
//       A:任何一行的第一列和最后一列都是1
//       B:从第三行开始，每一个数据是它上一行的前一列和它上一行的本列之和。

//  步骤：
//       A：首先定义一个二维数组。行数如果是n，我们把列数也先定义为n。
//          这个n的数据来自于键盘录入。
//       B：给这个二维数组任何一行的第一列和最后一列赋值为1
//       C：按照规律给其他元素赋值
//            从第三行开始，每一个数据是它上一行的前一列和它上一行的本列之和。
//       D：遍历这个二维数组。

```

数组工具类

Arrays类

JDK中提供了一个专门用于操作数组的工具类，即Arrays类，位于java.util包中。该类提供了一些列方法来操作数组，如排序、复制、比较、填充等，用户直接调用这些方法即可，不需要自己编码实现，降低了开发难度。Arrays类的常用方法

方法	返回类型	说明
equals(array1,array2)	boolean	比较两个数组是否相等
sort(array)	void	对数组array的元素进行排序
toString(array)	String	将一个数组array转换成一个字符串
fill(array,val)	void	把数组array的所有元素都赋值成val
copyOf(array,length)	与array数据类型一致	把数组array复制成一个长度为length的新数组
binarySearch(array,val)	int	查询元素值val在数组array中的下标
compare(array1,array2)	int	按字典顺序比较数组，前面的数组大，返回大于0的值，反之返回小于0的值
copyOfRange(arr,start,end)	与array数据类型一致	将指定数组的指定范围复制到新数组中。
fill(arr,start,end, val)	void	将指定的值分配给指定数组的指定范围的每个元素。
mismatch(array1,array2)	int	查找并返回两个数组之间第一个不匹配的索引，如果未找到不匹配，则返回 -1。
mismatch(array1,start1, end1,array2,start2,end2)	int	查找并返回指定范围内两个数组之间第一个不匹配的相对索引，如果未找到不匹配，则返回 -1。

Arrays类的应用

比较两个数组是否相等

Arrays类的equals()方法用于比较两个数组是否相等。只有当两个数组长度相等，对应位置的元素也一一相等时，该方法返回true，否则返回false。

对数组元素进行升序排序

Arrays类的sort()方法对数组的元素进行升序排序。

将数组转换成字符串

Arrays类中提供了专门输出数组内容的toString()方法。该方法用于将一个数组转换成一个字符串。它按顺序把多个数组元素连在一起，多个数组元素之间使用英文逗号和空格隔开。利用这种方法可以很清楚地观察到各个数组元素的值。

将数组所有元素赋值为相同的值

Arrays类的fill(array,val)方法用于把数组array的所有元素都赋值为val。

将数组赋值成一个长度为设定值的新数组

Arrays类的copyOf()方法把数组复制成一个长度为设定值的新数组。

> Arrays类的copyOf(array,length)方法可以进行数组复制，把原数据复制成一个新数组，其中length是新数组的长度。如果length小于原数组的长度，则新数组就是原数组的前面length个元素；如果length大于原数组的长度，则新数组前面的元素就是原数组的所有元素，后面的元素是按照数组类型补充的默认值，如整数补充0，浮点数补充0.0等。

System.arraycopy() 方法从指定的源数组复制一个数组，从指定位置开始，到目标数组的指定位置。该方法声明如下：

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

参数解析：

- src：这是源数组。
- srcPos：这是源数组中的起始位置。
- dest：这是目标数组。
- destPos：这是目标数据中的起始位置。
- length: 这是要复制的数组元素的数量。

数组组件的子序列从 src 引用的源数组复制到 dest 引用的目标数组。复制的组件数等于 length 参数。源数组中位置srcPos到srcPos + length - 1的元素被复制到目标数组的destPos 到 destPos + length - 1的位置。

查询元素在数组中的下标

Arrays类的binarySearch(Object[],Object key)方法用于查询数组元素在数组中的下标。调用该方法时要求数组中的元素已经按升序排列。如果key在数组中，则返回搜索值的索引；如果key不在数组中，返回值-1或“-”（插入点）。插入点的值有如下四种情况。

[1] 搜索值是数组元素，从0开始计数，得搜索值的索引值；

[2] 搜索值不是数组元素，且在数组范围内，从1开始计数，得“ - 插入点索引值”；

[3] 搜索值不是数组元素，且大于数组内元素，索引值为 – (length + 1);

[4] 搜索值不是数组元素，且小于数组内元素，索引值为 – 1。

练习

```
// 定义一个5*5的二维数组并初始化，找出数组中的最大值的坐标。
int[][] arr = {
    {56, 6, 3, 7, 5},
    {34, 10, 5, 10, 4},
    {354, 435, 34, 23, 54},
    {325, 56, 98, 454, 3},
    {3, 5, 7, 6, 4},
};
// 定义一个5*5的二维数组并初始化，找出最小值的坐标，计算出该坐标周边的所有元素之和。
```