数 组

> Author：zhangzhang
>
> Version：1.0.0

一、引言

1.1 为什么使用数组

> 如何存储100名学生的成绩?
>
> - 办法：使用变量存储，重复声明100个double类型变量即可。
> - 缺点：麻烦，重复操作过多。

> 如何让100名学生成绩全部+1?
>
> - 办法：100个变量重复相同操作，直至全部完毕。
> - 缺点：无法进行统一的操作。

二、数组的概念

2.1 数组的概念

> 概念：一组连续的存储空间，存储多个相同数据类型的值。

特点：

- 类型相同。
- 长度固定。

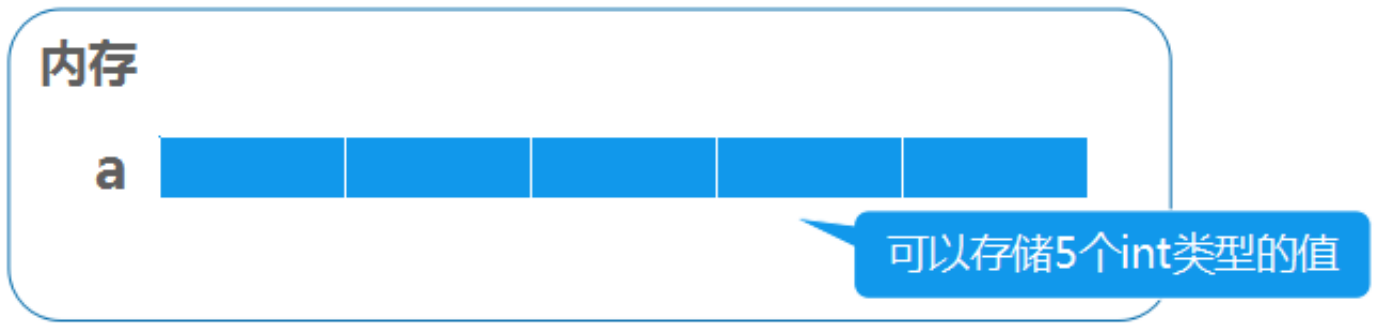
三、数组的声明与赋值

3.1 数组的创建

```
public class TestCreateArray {
    public static void main(String[] args) {
        int[] a = new int[5];
    }
}
```

声明int数组类型变量
定义变量名为a

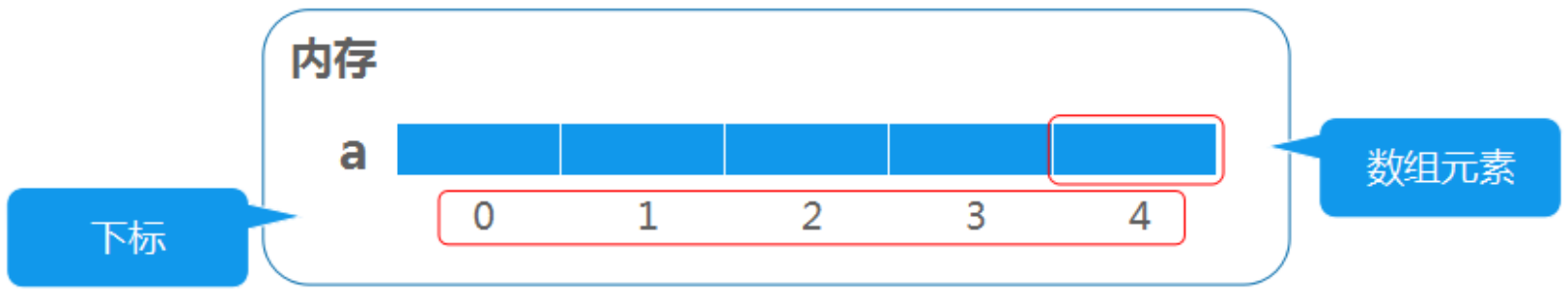
分配长度为5的连续空间



四、数组的组成

4.1 数组的组成

- 数组中的每个数据格被称为“数组元素”。
- 对每个元素进行赋值或取值的操作被称为“元素的访问”。
- 访问元素时，需要使用“下标”（从0开始，依次+1，自动生成）。
- 访问的语法：数组名[下标]; //例如 存：a[0]=10; 取：a[0];



4.2 数组的使用

```
public class TestCreateArray {
    public static void main(String[] args) {
        int[] a = new int[5];
        a[0]=5;
        a[1]=3;
        a[2]=4;
        a[3]=7;
        a[4]=10;
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
    }
}
```


运行结果：

```
5
3
4
7
10
```

4.3 下标的范围

```
public class TestCreateArray {
    public static void main(String[] args) {
        int[] a = new int[5];
        a[0]=5;
        a[1]=3;
        a[2]=4;
        a[3]=7;
        a[4]=10;
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
        System.out.println(a[5]);
    }
}
```

- 有效下标范围：0 ~ 数组长度-1
- 访问无效下标，会导致数组下标越界

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
```

五、数组的遍历【重点】

5.1 数组的遍历

> 遍历：从头至尾，逐一对数组的每个元素进行访问。

```
public class TestVisitArray {
    public static void main(String[] args) {
        int[] a = new int[5];
        a[0]=5;
        a[1]=3;
        a[2]=4;
        a[3]=7;
        a[4]=10;
        for (int i = 0; i < a.length; i++) {
            System.out.println(a[i]);
        }
    }
}
```

数组名.length 可动态获得数组长度。

使用循环变量“i”充当下标，
逐一访问数组中的每个元素。

```
public class TestCreateArray{

  public static void main(String[] args){

    int[] a = new int[8]; //在内存中创建长度为5的整数数组

    a[0] = 11;
    a[1] = 22;
    a[2] = 33;
    a[3] = 44;
    a[4] = 55;
    a[5] = 66;
    a[6] = 77;
    a[7] = 88;

    /*
    System.out.println( a[0] );
    System.out.println( a[1] );
    System.out.println( a[2] );
    System.out.println( a[3] );
    System.out.println( a[4] );
    */

    //          i < 8
    for(int i = 0 ; i < a.length ; i++){// 1 <= 5    0 <= 4     0 < 5
      System.out.println( a[i] );
    }

  }
}
```

5.2 数组的默认值

```
public class TestDefaultValue {
    public static void main(String[] args) {
        int[] a = new int[5];

        for (int i = 0; i < 5; i++) {
            System.out.println(a[i]);
        }
    }
}
```

在没有为数组元素赋值的情况下，
依旧可以正确访问。

运行结果：

```
0
0
0
0
0
```

> 数组默认值：
> 整数：0
> 小数：0.0
> 字符：\u0000
> 布尔：false
> 其他：null

```
public class TestDefaultValue{

  public static void main(String[] args){

    int[] a = new int[5];

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

    double[] b = new double[5];

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

    String[] strs = new String[4];

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

  }
}
```

5.3 数组创建语法

> - 先声明、再分配空间：
>   - 数据类型[] 数组名;
>   - 数组名 = new 数据类型[长度];
> - 声明并分配空间：
>   - 数据类型[] 数组名 = new 数据类型[长度];
> - 声明并赋值（繁）：
>   - 数据类型[] 数组名 = new 数据类型[]{value1,value2,value3,...};
> - 声明并赋值（简）：
>   - 数据类型[] 数组名 = {value1,value2,value3,...};（显示初始化，注意：不可换行）。

```
public class TestCreates{

  public static void main(String[] args){

    //先声明、再分配空间
    int[] array1;

    array1 = new int[4];

    //System.out.println( array1[0] );

    //声明并分配空间
    int[] array2 = new int[4];


    //声明并赋值（繁）
    int[] array3;
    array3 = new int[]{ 11 , 22 , 33};

    for(int i = 0 ; i < array3.length ; i++){
      System.out.println( array3[i] );
    }

    //声明并赋值（简）
    int[] array4 = { 66,77,88,99 };//不支持换行书写

    for(int i = 0 ; i < array4.length ; i++){
      System.out.println( array4[i] );
    }
  }
}
```

5.4 课堂案例

> 给定一个整数数组，统计数组中所有元素的平均值。

```java
public class TestGetAvg{

  public static void main(String[] args){

    int[] numbers = new int[]{55,66,77,88,99};

    int sum = 0;

    for(int i = 0 ; i < numbers.length ; i++){

      sum += numbers[i];

    }

    double avg = sum / numbers.length;

    System.out.println(avg);

  }
}
```

> 给定一个整数数组，读入一个整数n，如果n在数组中存在，输出下标，不存在则输出-1。

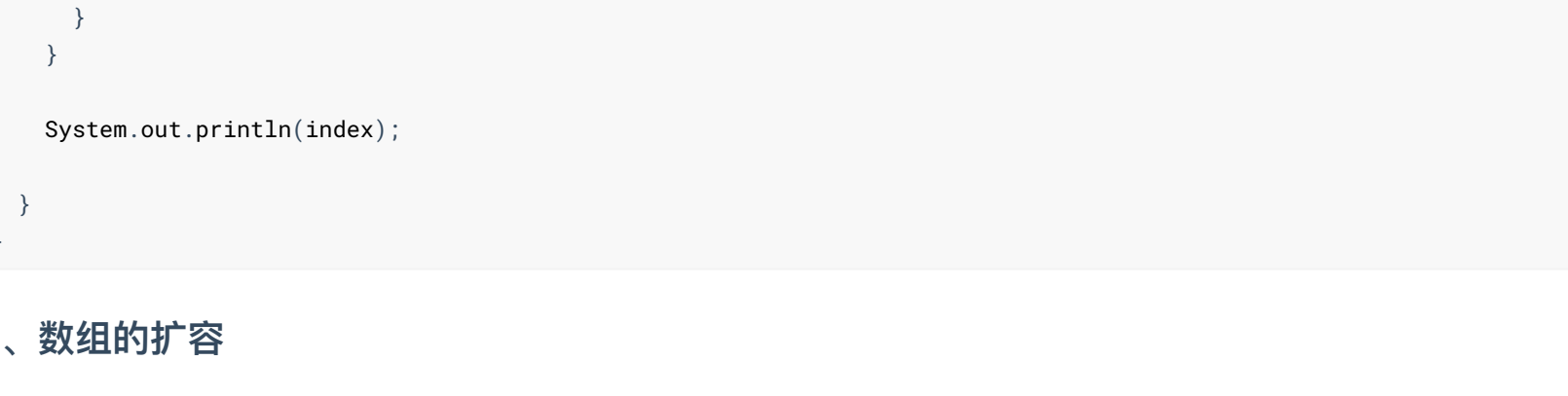
```java
import java.util.Scanner;

public class TestSearch{

  public static void main(String[] args){

    Scanner input = new Scanner(System.in);


    System.out.println("请输入一个整数: ");

    int n = input.nextInt();


    int[] numbers = new int[]{1,2,3,4,5,6,7};

    int index = -1;//代表n从未出现在数组中

    //循环查找的过程
    for(int i = 0 ; i < numbers.length ; i++){
      if(numbers[i] == n){
        //存在
        index = i;//改变index，代表n所出现的下标
        break;
      }
    }

    System.out.println(index);

  }
}
```

六、数组的扩容

6.1 数组的扩容

> 创建数组时，必须显示指定长度，并在创建之后不可更改长度。

> 扩容的思路：
>
> - 创建大于原数组长度的新数组。
> - 将原数组中的元素依次复制到新数组中。



6.2 复制的方式

> - 循环将原数组中所有元素逐一赋值给新数组。

```
public class TestCopyArray{

  public static void main(String[] args){

    int[] nums = new int[5];//数组创建之后，长度不可变

    nums[0] = 11;
    nums[1] = 22;
    nums[2] = 33;
    nums[3] = 44;
    nums[4] = 55;

    //1.创建比原数组大的新数组
    int[] newNums = new int[ nums.length * 2 ];

    //2.复制原数组中的所有数据到新数组
    for(int i = 0 ; i < nums.length ; i++){
      newNums[i] = nums[i];
    }



    //遍历原数组
    for(int i = 0 ; i < nums.length ; i++){
      System.out.print( nums[i] +"\t");
    }
    System.out.println();

    //遍历新数组
    for(int i = 0 ; i < newNums.length ; i++){
      System.out.print( newNums[i] +"\t");
    }

  }
}
```

- System.arraycopy(原数组,原数组起始,新数组,新数组起始,长度)。

```
public class TestCopyArray2{

  public static void main(String[] args){

    int[] nums = new int[5];//数组创建之后，长度不可变

    nums[0] = 11;
    nums[1] = 22;
    nums[2] = 33;
    nums[3] = 44;
    nums[4] = 55;


    //复制的赋值

    //1.创建新数组
    int[] newNums = new int[ nums.length * 2 ];


    //2.使用System.arraycopy(原数组，原数组起始下标，新数组，新数组的起始存储下标，需要赋值的个数或长度);
    System.arraycopy( nums , 0 , newNums , 0 , nums.length);


    //遍历新数组
    for(int i = 0 ; i < newNums.length ; i++){
      System.out.print( newNums[i] +"\t");
    }

  }
}
```

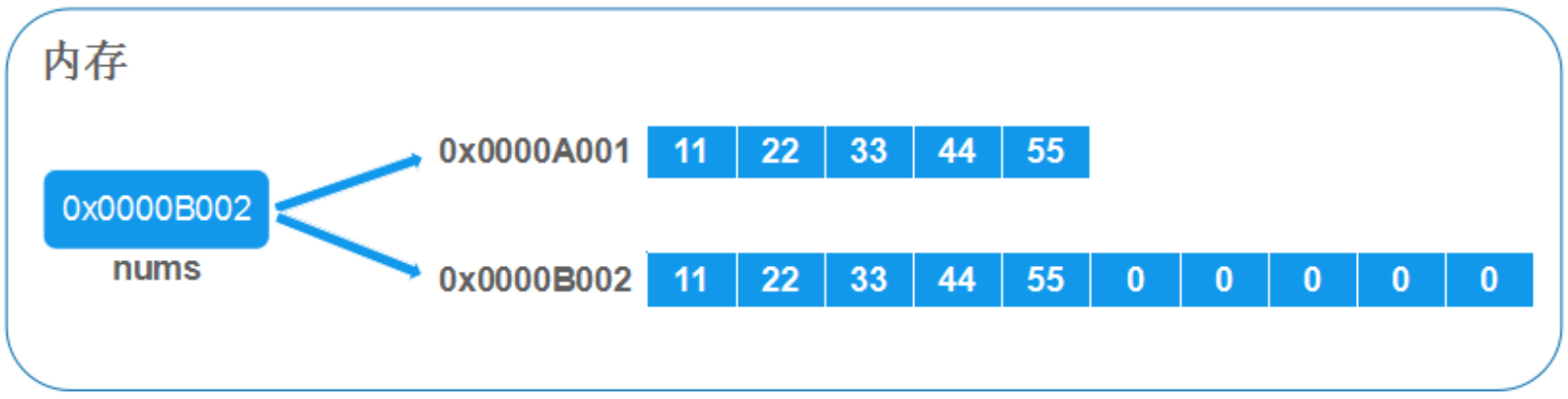
- java.util.Arrays.copyOf(原数组, 新长度)。
- 返回带有原值的新数组。

```
//import java.util.Arrays;

public class TestCopyArray3{

  public static void main(String[] args){

    int[] nums = new int[5];//数组创建之后，长度不可变

    nums[0] = 11;
```

```
    nums[1] = 22;
    nums[2] = 33;
    nums[3] = 44;
    nums[4] = 55;


    //1.创建新数组、复制元素
    int[] newNums = java.util.Arrays.copyOf(nums , nums.length * 2);//将带有原值的新数组返回给我们


    //2.遍历
    for(int i = 0 ; i < newNums.length ; i++){
      System.out.print( newNums[i] +"\t");
    }
  }
}
```

6.3 地址替换

> - 数组作为引用类型之一，其变量中存储的是数组的地址。
> - 完成元素复制后，需将新数组地址，赋值给原变量进行替换。

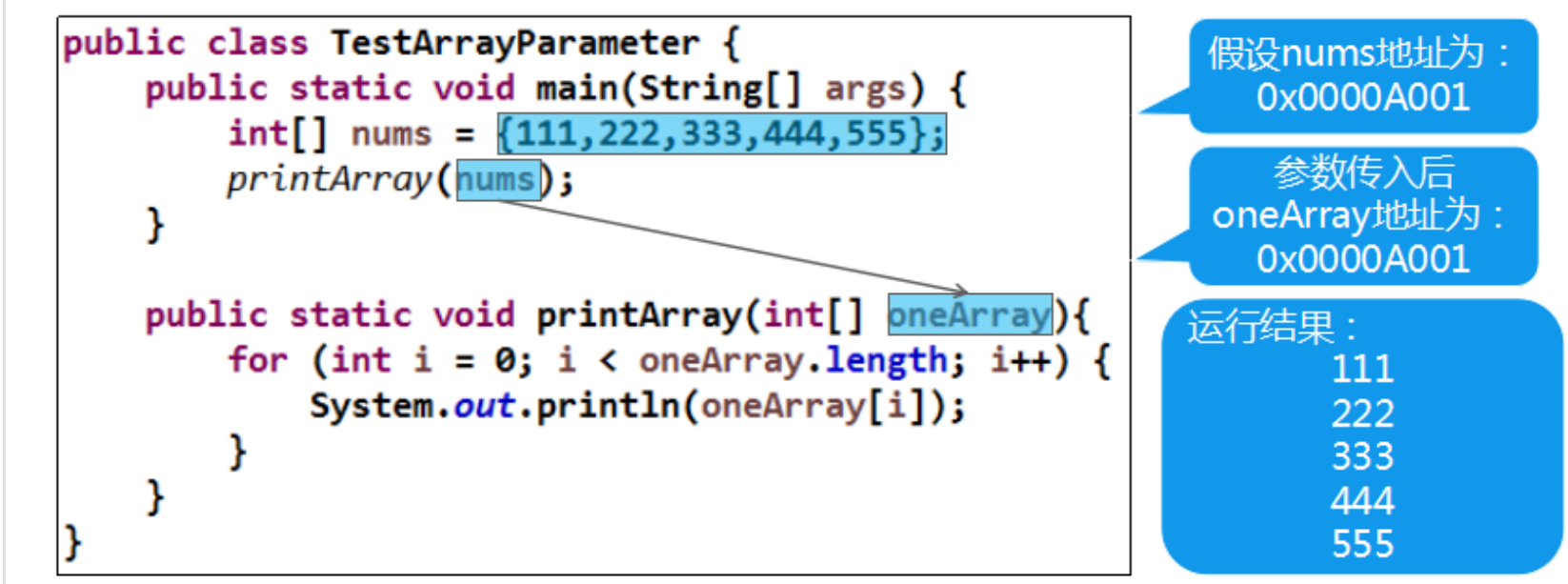



```
public class TestAddress{

  public static void main(String[] args){

    //Java的两大类数据类型


    //1.基本数据类型，存储在栈中（基本类型的变量中，直接存储“值”）
    int a = 10; //



    //2.引用数据类型，存储在堆中（引用类型的变量中，存储的是堆中的“地址”）
    String str = "abc";


    //int[] nums = new int[4];

    //nums[0] = 10;

    //System.out.println(nums);


    double[] scores = new double[4];

    System.out.println(scores);



    /*
    String home = " 吉林省吉林市。。。。。1-1-201"; //0x00001111


    home = "北京市 昌平区。。。。。2-2-305" //0x00002222


    买了台电脑，放到家里
    home ---> ?
    */


    int[] nums = new int[5]{55,11,22,33,88};//0x0000A001
```

```
    int[] newNums = new int[10]{55,11,22,33,88,0,0,0,0,0};//0x0000B002


    //地址的替换
    nums = newNums;


    ////插入一个新值，为了可以存储更多的值
    nums[5] = 99;



  }
}
```

七、数组类型的参数

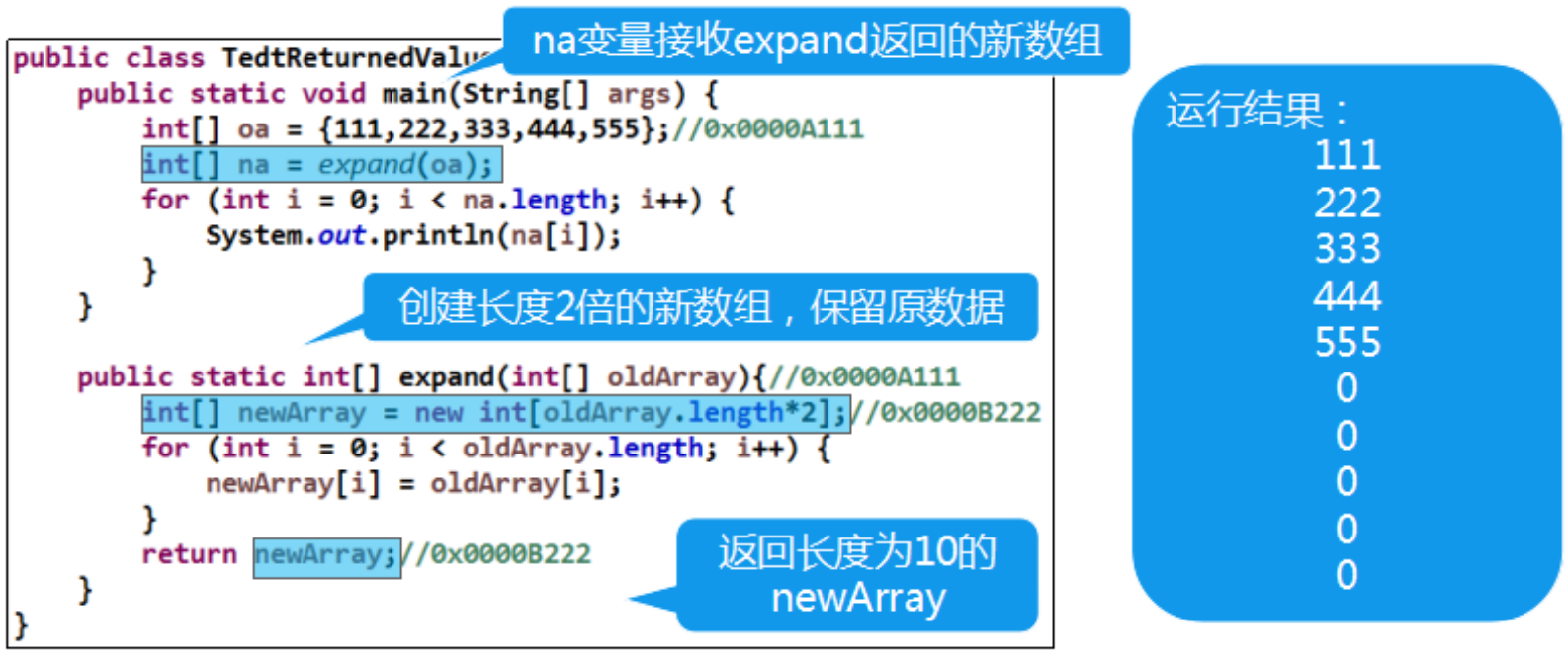
7.1 数组类型的参数



- 方法调用时，将nums中的地址赋值给oneArray，此时二者指向同一个数组。
- 传递参数时：基本类型传递的是变量中的值；引用类型传递的是变量中的地址。

```
public class TestParams{

  public static void main(String[] args){

    int[] nums = new int[]{11,22,33,44,55}; //0x00000001

    print(nums);

    int[] array = new int[]{66,77,88,99}; //0x00000002

    print(array);

    //引用数据类型，赋值也好、传参也好，操作的都是地址
  }

  //将一段需要重复使用逻辑代码，放在一个函数中，在需要使用的为进行调用
  //可以遍历所有的int数组
  public static void print(int[] arr){//arr = 0x00000002

    for(int i = 0 ; i < arr.length ; i++){
      System.out.print( arr[i] +"\t");
    }
    System.out.println();

  }

}
```

7.2 数组类型的返回值



> - 调用数组类型返回值的方法时，方法执行后，返回的是数组的地址。

```java
public class TestCompare{

  public static void main(String[] args){

    //1.基本数据类型的传递，是“值”的传递，一方改变，不会影响另一方
    //2.引用数据类型的传递，是“地址”的传递，一方改变，会影响另一方

    int i = 10;
    m1(i);
    System.out.println(i);

    //----------------------

    byte b = 123;

    b = m2(b);

    System.out.println(b);

    //----------------------

    int[] nums = {1,2,3,4}; // nums = 0x11223344(长度为4的数组，存储了888,2,3,4)

    m3(nums);

    System.out.println( nums[0] +"\t"+ nums[1] +"\t"+ nums[2]+"\t"+ nums[3]);

    //----------------------

    int[] numbers = {5,4,3,2,1}; //0x11112222

    m4(numbers);

    System.out.println( numbers.length );

    //--------------------------------

    int[] arrays = {111,222,333};//0x77778888

    expand(arrays);//0x12345678

    System.out.println(arrays.length);
  }

  public static void m1(int n){
    n = n * 2;
    System.out.println(n);
  }

  public static byte m2(byte c){
    c++;
    return c;
  }

  public static void m3(int[] arr){ // 0x11223344
    arr[0] = 888;
  }

  public static void m4(int[] arr){ //0x11112222
    int[] numbers = java.util.Arrays.copyOf(arr , arr.length * 2); //0x45456666 new int[arr.length * 2]
  }


  public static int[] expand(int[] arr){//0x77778888
```

```
    int[] newNums = new int[arr.length * 2];//0x12345678

    System.arraycopy(arr , 0 , newNums , 0 , arr.length);

    return newNums;//0x12345678
  }


}
```

八、可变长参数

8.1 可变长参数

> 概念：可接收多个同类型实参，个数不限，使用方式与数组相同。

> 语法：数据类型... 形参名 //必须定义在形参列表的最后，且只能有一个。

```
public class TestArrayParameter {
    public static void main(String[] args) {
        printArray(111,222,333,444,555);
    }

    public static void printArray(int... oneArray){
        for (int i = 0; i < oneArray.length; i++) {
            System.out.println(oneArray[i]);
        }
    }
}
```

可为可变长参数赋予
0 ~ N 个实际参数

定义int型可变长参数

运行结果：
111
222
333
444
555

```
public class TestChangeLength{

  public static void main(String[] args){

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

    //1.支持传递数组类型的实参
    //method(numbers);

    //2.支持传递零散数据的实参
    method(1,2,3,4,5); //也支持0个参数

    //---------------------------


    int[] newArray = expand(20,11,22,33,44,55,66);

    method(newArray);
  }

  //函数的参数是“可变长参数”
  public static void method(int... arr){// int[] arr
    System.out.println("---method executed---");
    for(int i = 0 ; i < arr.length ; i++){
      System.out.print( arr[i] +"\t");
    }
    System.out.println();
  }


  public static int[] expand(int length , int... arr){//规则：必须在形参列表的最后，且只有一个

    int[] newNums = new int[length];

    System.arraycopy(arr , 0 , newNums , 0 , arr.length);

    return newNums;
  }

}
```

九、数组的排序【重点】

9.1 数组的排序

> - 冒泡排序：相邻的两个数值比较大小，互换位置。

```
public class TestBubble{

  public static void main(String[] args){

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

    //冒泡排序：相邻的两个数值比较大小，互换位置


    //轮次：数组长度-1
    //单轮次数：(数组长度-1)基础上再做逐级递减


    for(int i = 0 ; i < nums.length - 1 ; i++){// i=3  外层循环：控制比较轮次（num.length-1）比较4轮
      for(int j = 0 ; j < nums.length - 1 - i ; j++){ // 内层循环：单轮当中比较次数
        if(nums[j] < nums[j+1]){
          int temp = nums[j];
          nums[j] = nums[j+1];
          nums[j+1] = temp;
        }
      }

    }

    //外层4次
    //内层4次
    //第一轮4次，第二轮3次，第三轮2次，第四轮1次

    /*

    //第一轮
    for(int j = 0 ; j < nums.length - 1 ; j++){// j = 4;
      if(nums[j] > nums[j+1]){
        int temp = nums[j];
        nums[j] = nums[j+1];
        nums[j+1] = temp;
      }
    }


    //第二轮
    for(int j = 0 ; j < nums.length - 1 - 1 ; j++){// j = 2;
      if(nums[j] > nums[j+1]){
        int temp = nums[j];
        nums[j] = nums[j+1];
        nums[j+1] = temp;
      }
    }


    //第三轮
    for(int j = 0 ; j < nums.length - 1 - 2 ; j++){// j = 1;
      if(nums[j] > nums[j+1]){
        int temp = nums[j];
        nums[j] = nums[j+1];
        nums[j+1] = temp;
      }
    }


    //第四轮
    for(int j = 0 ; j < nums.length - 1 - 3 ; j++){// j = 0;
      if(nums[j] > nums[j+1]){
        int temp = nums[j];
        nums[j] = nums[j+1];
        nums[j+1] = temp;
      }
    }
    */
    for(int i = 0 ; i < nums.length ; i++){
      System.out.print(nums[i]+"\t");
    }
    System.out.println();

    //两值交换，借助第三变量
    int a = 10;
    int b = 20;
    int temp = a;// temp = 10
    a = b; // a = 20;
    b = temp; // b = temp;
    //System.out.println(a +"\t"+ b);
  }
```

```
}
```

> - 选择排序：固定值与其他值依次比较大小，互换位置。

```
public class TestSelect{

  public static void main(String[] args){

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

    //固定值与其他值依次比较，互换位置


    for(int i = 0 ; i < nums.length - 1 ; i++){ // i = 2 //外层控制轮次（4轮）

      for(int j = i + 1 ; j < nums.length; j++){// j = 3
        if(nums[i] > nums[j]){
          int temp = nums[i];
          nums[i] = nums[j];
          nums[j] = temp;
      }
    }

  }

  /*

  //第一轮
  int i = 0;

  for(int j = i + 1 ; j < nums.length; j++){// j = 4

    if(nums[i] > nums[j]){
      int temp = nums[i];
      nums[i] = nums[j];
      nums[j] = temp;
    }

  }


  //第二轮
  i++; // i = 1

  for(int j = i + 1 ; j < nums.length ; j++){ // j = 4

    if(nums[i] > nums[j]){
      int temp = nums[i];
      nums[i] = nums[j];
      nums[j] = temp;
    }

  }

  //第三轮
  i++; // i = 2

  for(int j = i + 1 ; j < nums.length ; j++){ // j = 4

    if(nums[i] > nums[j]){
      int temp = nums[i];
      nums[i] = nums[j];
      nums[j] = temp;
    }

  }


  //第四轮
  i++; // i = 3

  for(int j = i + 1 ; j < nums.length ; j++){ // j = 4

    if(nums[i] > nums[j]){
      int temp = nums[i];
      nums[i] = nums[j];
      nums[j] = temp;
    }

  }


  */
  for(int k = 0 ; k < nums.length ; k++){
    System.out.print( nums[k] + "\t");
  }
```

```
    System.out.println();
  }
}
```

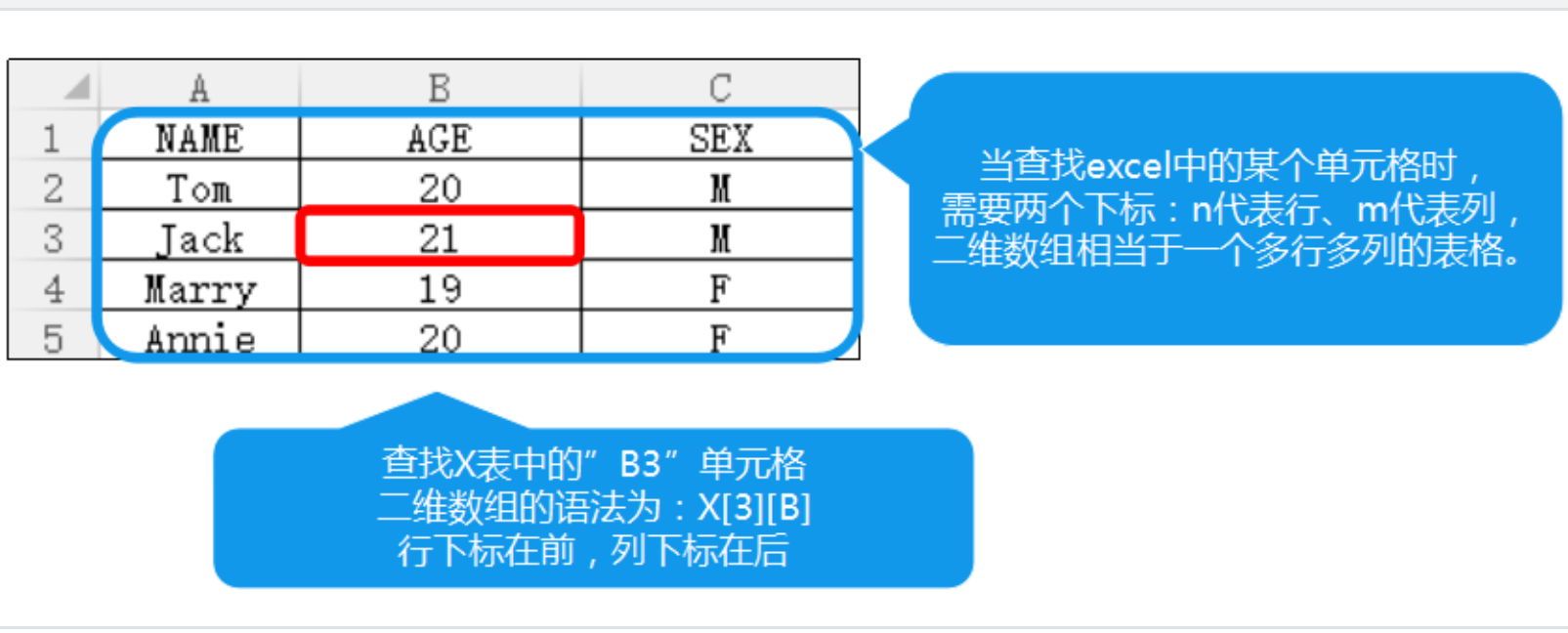
> - JDK排序：java.util.Arrays.sort(数组名)。

```
import java.util.Arrays;

public class TestSort{

  public static void main(String[] args){

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

    //借助JDK提供的数组工具，进行排序
    Arrays.sort(nums);

    //第一次遍历（升序）
    for(int i = 0 ; i < nums.length ; i++){
      System.out.println(nums[i]);
    }


    //降序：需要手工的方式完成元素的倒置  5 2 3 4 1

    for(int i = 0 ; i < nums.length / 2 ; i++){// i = 0

      int temp = nums[i];// int temp = 1;

      nums[i] = nums[ nums.length - 1 - i];

      nums[ nums.length - 1 - i] = temp;

    }

    //第二次遍历（降序）
    for(int i = 0 ; i < nums.length ; i++){
      System.out.println(nums[i]);
    }


    //两值交换，借助第三变量
    /*
    int a = 10;
    int b = 20;
    int c = a;//将a中的值保存在c中
    a = b;//将b中的值保存在a中
    b = c;//将c中的值保存在b中
    */
  }
}
```

十、二维数组

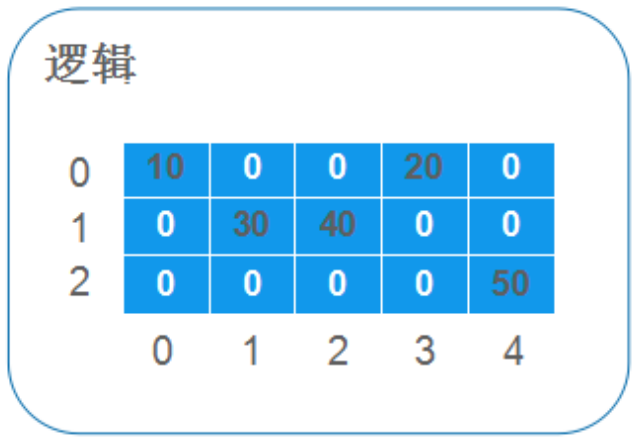
10.1 二维数组的概念

> 概念：一维数组中的一维数组；数组中的元素，还是数组。



10.2 二维数组的赋值

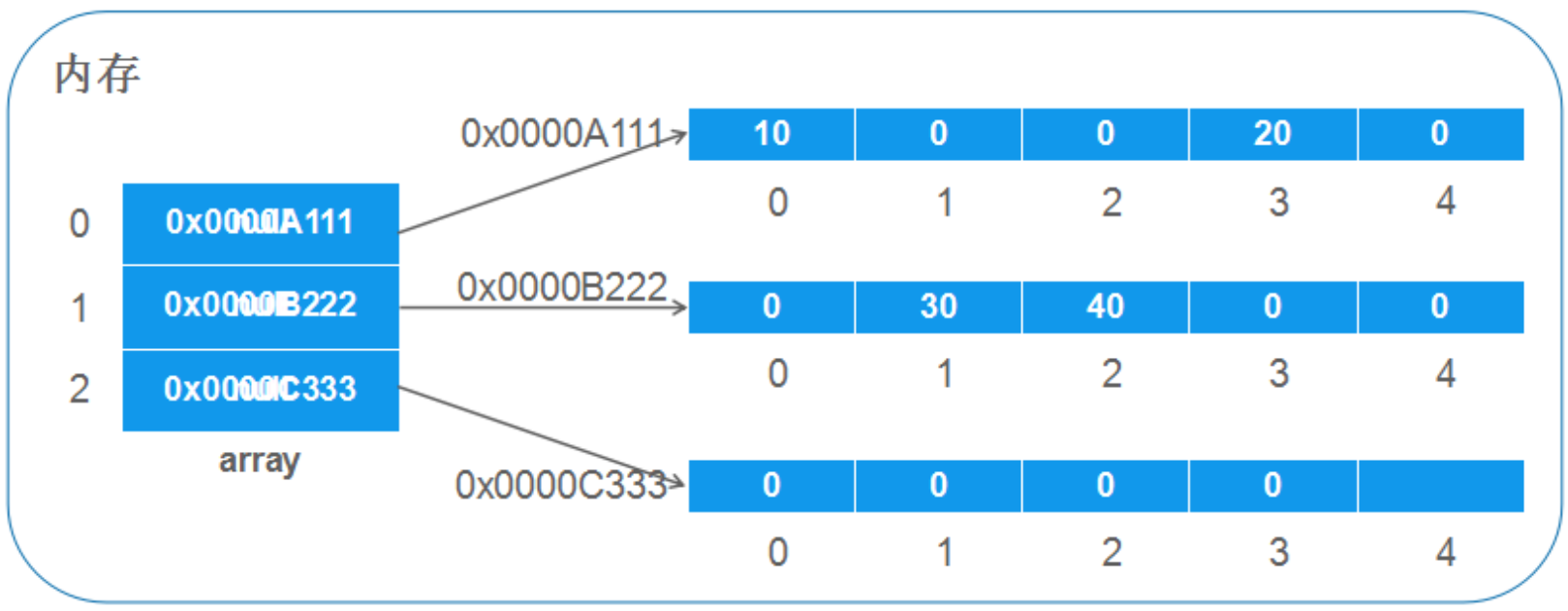
```java
public class Test2DArray {
    public static void main(String[] args) {
        int[][] array = new int[3][5];
        array[0][0] = 10;
        array[0][3] = 20;
        array[1][1] = 30;
        array[1][2] = 40;
        array[2][4] = 50;
    }
}
```



使用双下标访问二维数组中的元素：

- 第一个下标代表：行号（高维下标）。
- 第二个下标代表：列号（低维下标）。

10.3 二维数组的内存分配



高维数组中的每一个元素，保存了低维数组的地址。访问array[0]等价于在访问0x0000A111。

10.4 二维数组的访问

```java
public class Test2DArray {
    public static void main(String[] args) {
        int[][] array = new int[3][5];
        array[0][0] = 10;
        array[0][3] = 20;
        array[1][1] = 30;
        array[1][2] = 40;
        array[2][4] = 50;
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print(array[i][j]);
            }
            System.out.println();
        }
    }
}
```

访问低维长度：
array[0].length
首个低维数组的长度

访问低维数组元素：
array[0][0]
首个低维数组的首个元素

```java
public class Test2DArray{

  public static void main(String[] args){

    int[][] nums = new int[3][5];

    nums[0][0] = 10;//第一行，第一列
    nums[0][3] = 20;//第一行，第四列
    nums[1][0] = 30;//第二行，第一列
    nums[1][1] = 40;//第二行，第二列
    nums[2][2] = 50;//第三行，第三列
    nums[2][4] = 60;//第三行，第五列


    for(int i = 0 ; i < nums.length ; i++){ //外层控制行数

      for(int j = 0 ; j < nums[i].length ; j++){ // j = 1 内层控制列数
```

```
        System.out.print( nums[i][j] +"\t");

      }
      System.out.println();

    }

    System.out.println( nums.length );//nums变量中，所持有的地址对应的数组长度是多少 = 3

    System.out.println( nums[0].length );// nums[0]，高位数组中的第一个元素所指向的低维数组的长度


    System.out.println( nums[0] ); // 长度5    先创建了长度为5的低维数组，将地址赋值给高维空间
    System.out.println( nums[1] );
    System.out.println( nums[2] );


  }
}
```

10.5 二维数组创建语法

- 先声明、再分配空间：
  - 数据类型[][] 数组名;
  - 数组名 = new 数据类型[高维长度][低维长度];
- 声明并分配空间：
  - 数据类型[][] 数组名 = new 数据类型[高维长度][低维长度];
- 声明并赋值（繁）：
  - 数据类型[][] 数组名 = new 数据类型[高维长度][]; （不规则数组，自行new低维数组）。
- 声明并赋值（简）：
  - 数据类型[][] 数组名 = { {v1,v2,v3},{v4,v5},{v6,v7,v8,v9} }; （显示初始化）。

```
public class Test2DArray2{

  public static void main(String[] args){


    int[][] array = new int[3][]; //只有高维空间，没有低维空间

    array[0] = new int[5];
    array[1] = new int[3];
    array[2] = new int[7];

    System.out.println(array[0]);
    System.out.println(array[1]);
    System.out.println(array[2]);


    System.out.println(array[0].length);
    System.out.println(array[1].length);
    System.out.println(array[2].length);


    for(int i = 0 ; i < array.length ; i++){

      for(int j = 0 ; j < array[i].length ; j++){
        System.out.print(array[i][j] +"\t");
      }
      System.out.println();

    }

    //显示初始化
    int[][] numbers = { {1,2,3} , {4,5,6,7} , {8,9} };

    System.out.println(numbers.length);

    System.out.println("---------------");

    System.out.println(numbers[0].length);
    System.out.println(numbers[1].length);
    System.out.println(numbers[2].length);


  }
}
```

10.6 课堂案例

> 杨辉三角

```
public class TestYH{

  public static void main(String[] args){

    //####1#
    //###1#1
    //##1#0#1
    //#1#0#0#1
    //1#0#0#0#1

    int rows = 7;

    int[][] yh = new int[rows][];

    //创建多个不同长度的二维数组
    for(int i = 0 ; i < rows ; i++){
      yh[i] = new int[i+1];
    }

    //完成初始值的赋值（每行的首位都是1）
    for(int i = 0 ; i < yh.length ; i++){ // i = 2
      yh[i][0] = 1;
      yh[i][i] = 1;
    }

    //计算
    for(int i = 2 ; i < yh.length ; i++){
      for(int j = 1 ; j < i ; j++){
        //当前位置的值 = 上一行的同列 + 上一行的前一个列
        yh[i][j] = yh[i-1][j] + yh[i-1][j-1];
      }
    }

    for(int i = 0 ; i < yh.length ; i++){

      for(int j = rows - 1 ; j > i ; j--){//满足4次
        System.out.print("\t");
      }


      for(int j = 0 ; j < yh[i].length ; j++){
        System.out.print( "\t"+ yh[i][j] +"\t");
      }
      System.out.println();
    }

  }

}
```