一、定义数组存储数据

- 数组的概念

> 用来存储一批同种类型数据（可以理解成容器）

1.0 静态初始化数组

```
    //数据类型[] 变量名 = new 数据类型[]{数据.......}
    int[] arr =new int[]{1,2,3,4}
    //简化
    //数据类型[] 变量名 ={数据.......}
    int[] arr ={1,2,3,4,5}
    //c语言
    //数据类型 变量名[] =new 数据类型[]{数据.......}
    //类型 变量名[] = {数据.......}
    int a[] = new int[]{12, 3, 4, 5};
    int b[] = {12, 3, 4, 5};

    @Test
    public void testArray1() {
        final int[] arrays = {1, 2, 3, 4, 5};
        final double[] arrays2 = {1.0, 2.0, 3, 4, 5};
        final float[] arrays3 = {1.0f, 2.0f, 3f, 4f, 5f};
        System.out.println(arrays);
        System.out.println(arrays2);
        System.out.println(arrays3);
    }
==============================================
[I@ba8a1dc
[D@4f8e5cde
[F@504bae78
  [---->数组
  @之后的是十六进制地址
  数组是引用数据类型，句柄(变量名)放在栈中，引用放在堆中
```

2.0 动态初始化数组

```
1. 先确定元素的类型和数组的长度，之后再存入具体数据（会默认初始化数组中的数据）
2. 默认类型初始值：
        byte short int long 0
        char '/u0000'转为int是0
        boolean false
        double float 0.0
        String Object null
3. 格式：
    数据类型[] 变量名 =new 数据类型[数组长度];
4.使用场景：
    当前已经知道存入的元素值，用静态初始化。
    当前还不清楚要存入哪些数据，用动态初始化
5. 注意：
    两种类型的写法是独立的，不能两边都用
```

```
int[] arrs new int[3]{1,2,3}----------->是错误的写法
```

元素默认值规则

数据类型	明细	默认值
基本类型	byte、short、char、int、long	0
	float、double	0.0
	boolean	false
引用类型	类、接口、数组、String	null

二、操作数组

1.0 数组的访问

```
数组名[索引]
通过这种方式寻找数组中的元素的时间复杂度是O(1)
因为可以通过简单的计算得到所找元素的地址【目的地址=首元素地址+数据类型大小*索引值】
```

2.0 数组的长度

```
数组名.length    数组属性length
当元素个数大于0；数组的最大索引length - 1
```

3.0 遍历

```
访问数组中的每一个元素
```

```
    @Test
    public void testArray3() {
        //存储的是二进制数字，'a'存的是97的二进制数，'4'存的是52的二进制，65存的是65的二进制
        final char[] ints = new char[]{'a', '4', 65, 99, 48};
        //这种方法操作比较灵活
        for (int i = 0; i < ints.length; i++) {
            System.out.print(i == ints.length - 1 ? ints[i] : ints[i] + ",");
        }
        System.out.println();
        System.out.println("-----------------------------------------------");
        //增强for循环也称为foreach 底层使用的是iterator
        for (char anInt : ints) {
            System.out.print(anInt + " ");
        }
        System.out.println();
        System.out.println("-----------------------------------------------");
    }
==========================================
```

```
a,4,A,c,0
-------------------------------------------
a 4 A c 0
```

4.0 去掉做高最低的平均分

```
    @Test
    public void testArray4() {
        //简化静态初始化
        final int[] arrs = {32, 3, 55, 1, 39, 54, 0};
        int max, min, sum = 0;
        max = min = arrs[0];
        for (int arr : arrs) {
            if (arr > max) {
                max = arr;
            }
            if (arr < min) {
                min = arr;
            }
            sum += arr;
        }
        System.out.println("去掉最高最低的平均分为：" + sum / (arrs.length - 2));
    }
```

5.0 猜数游戏

```
    public static void main(String[] args) {
        final int[] ints = new int[4];
        final Random r = new Random();
        for (int i = 0; i < 4; i++) {
            ints[i] = r.nextInt(6) + 45;//45----50
        }
        final Scanner sc = new Scanner(System.in);
        //外层循环标志
        OUT:
        //死循环
        while (true) {
            System.out.println("请输入45-----50之间的整数进行猜数");
            final int i = sc.nextInt();
            for (int a = 0; a < ints.length; a++) {
                if (ints[a] == i) {
                    System.out.println("恭喜猜对了，位置位于:" + a);
                    System.out.println(Arrays.toString(ints));
                    break OUT;//用于结束外层循环（break循环和switch） continue循环
                }
            }
            System.out.println("猜错了！");
        }
    }
```

```
请输入45-----50之间的整数进行猜数
45
恭喜猜对了，位置位于:2
[48, 50, 45, 45]
```

6.0 数组打乱

```
    @Test
    public void testArray5() {
        final int[] arrs = {32, 3, 55, 1, 39, 54, 0};
        System.out.println("原数组: " + Arrays.toString(arrs));
        final Random random = new Random();
        //遍历数组中的每一个元素
        for (int i = 0; i < arrs.length; i++) {
            //随机一个数组范围内的数字
            final int r = random.nextInt(arrs.length);
            //将遍历到的元素与随机索引的元素交换位置
            int tempAr = arrs[i];
            arrs[i] = arrs[r];
            arrs[r] = tempAr;
        }
        System.out.println("打乱后数组: " + Arrays.toString(arrs));
    }
============================================================
原数组:[32, 3, 55, 1, 39, 54, 0]
打乱后数组:[3, 55, 54, 32, 0, 1, 39]
```

7.0 冒泡排序

- 1

```
    /**
     * 没有优化
     */
    @Test
    public void testArrays2() {
        int[] arrs = {12, 33, 4, -3, 1, 90};
        for (int i = 0; i < arrs.length - 1; i++) {
            for (int j = 0; j < arrs.length - 1; j++) {
                if (arrs[j] > arrs[j + 1]) {
                    swap(arrs, j, j + 1);
                }
            }
            System.out.println("第" + (i + 1) + "轮" + Arrays.toString(arrs));
        }
        System.out.println("长度为:" + arrs.length);
        System.out.println("结果:" + Arrays.toString(arrs));
    }
========================================
第1轮[12, 4, -3, 1, 33, 90]
第2轮[4, -3, 1, 12, 33, 90]
第3轮[-3, 1, 4, 12, 33, 90]
第4轮[-3, 1, 4, 12, 33, 90]
第5轮[-3, 1, 4, 12, 33, 90]
长度为:6
结果:[-3, 1, 4, 12, 33, 90]
```

- 2

```
    /**
```

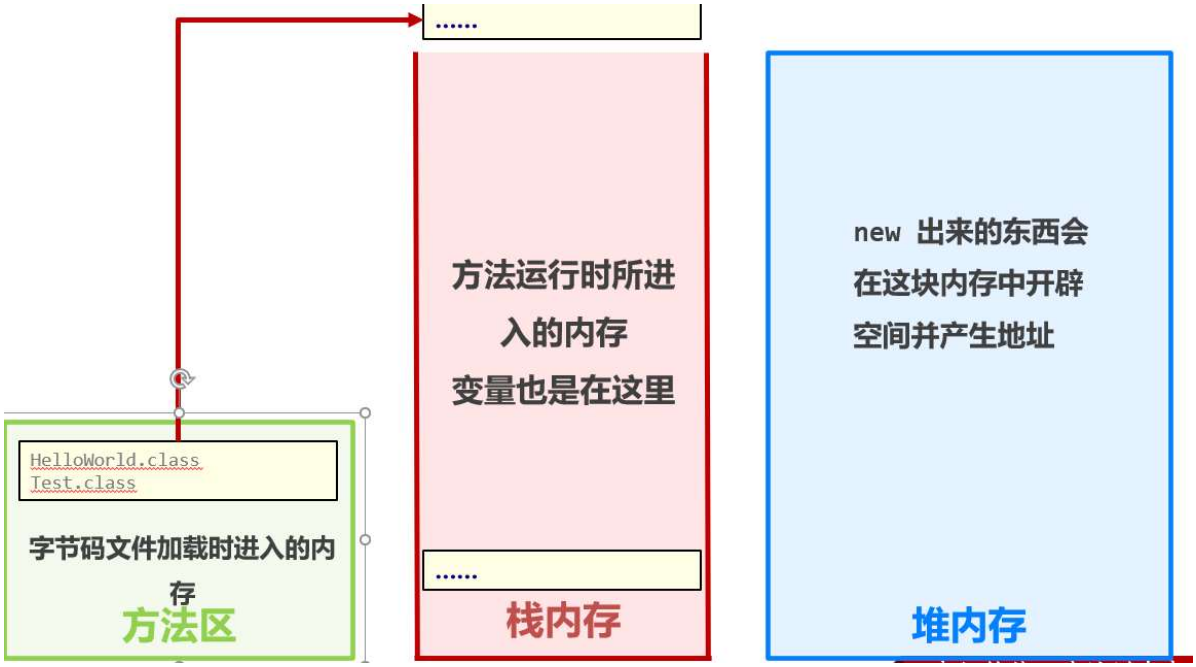
```
     * 优化一
     */
    @Test
    public void bubbleSortedArray() {
        int[] array = {12, 33, 4, -3, 1, 90};
        for (int i = 0; i < array.length - 1; i++) {
            //没有进行过交换
            boolean flag = false;
            for (int j = 0; j < array.length - 1 - i; j++) {
                if (array[j] > array[j + 1]) {
                    swap(array, j, j + 1);
                    //有进行过交换
                    flag = true;
                }
            }
            System.out.println("第" + (i + 1) + "轮" + Arrays.toString(array));
            //没有尽心过交换就退出
            if (!flag) break;
        }
        System.out.println("长度为:" + array.length);
        System.out.println("结果:" + Arrays.toString(array));
    }
==============================================
第1轮[12, 4, -3, 1, 33, 90]
第2轮[4, -3, 1, 12, 33, 90]
第3轮[-3, 1, 4, 12, 33, 90]
第4轮[-3, 1, 4, 12, 33, 90]
长度为:6
结果:[-3, 1, 4, 12, 33, 90]
```

- 3

```
    /**
     * 优化二
     */
    @Test
    public void bubbleSortedArray2() {
        int[] array = {12, 33, 4, -3, 1, 90};
        int count = 0;
        int n = array.length - 1;
        while (true) {
            //当数组已经有序，不出现交换时，让其退出
            int last = 0;
            for (int i = 0; i < n; i++) {
                if (array[i] > array[i + 1]) {
                    swap(array, i, i + 1);
                    //记录每次的交换位置
                    last = i;
                }
            }
            //记录最后的交换位置，并以此为次数的条件
            n = last;
            System.out.println("第" + (++count) + "轮" + Arrays.toString(array));
            if (n == 0) break;
        }
        System.out.println("长度为:" + array.length);
        System.out.println("结果:" + Arrays.toString(array));
```

```
    }
==========================================
第1轮[12, 4, -3, 1, 33, 90]
第2轮[4, -3, 1, 12, 33, 90]
第3轮[-3, 1, 4, 12, 33, 90]
第4轮[-3, 1, 4, 12, 33, 90]
长度为:6
结果:[-3, 1, 4, 12, 33, 90]
```

三、数组内存原理



- 说明

```
1. java源代码通过javac.exe编译成字节码文件再通过java.exe将字节码文件加载进jvm中
2. 字节码文件会加载进方法区，字节码文件会暴露静态方法的访问地址
3. 栈内存用来运行方法的(方法中又包含局部变量)，局部变量的引用在栈中
4. 基本数据类型的变量直接存储值
5. 引用数据类型的变量存储地址
6. 对象存在于堆内存中
```

四、java参数的传递方式【值传递】

```java
    /**
     * java参数传递方式都是值传递（数组地址值）
     */
    @Test
    public void test1() {
        int[] arrs = {1, 255, 3, 23, 0, -43};
        System.out.println("原来的值：" + Arrays.toString(arrs));
        System.out.println("原来的地址：" + arrs);
        set(arrs);
        System.out.println("变化后的值：" + Arrays.toString(arrs));

    }
    private void set(int[] arrs) {
        System.out.println("方法获得的地址：" + arrs);
        for (int i = 0; i < arrs.length; i++) {
```

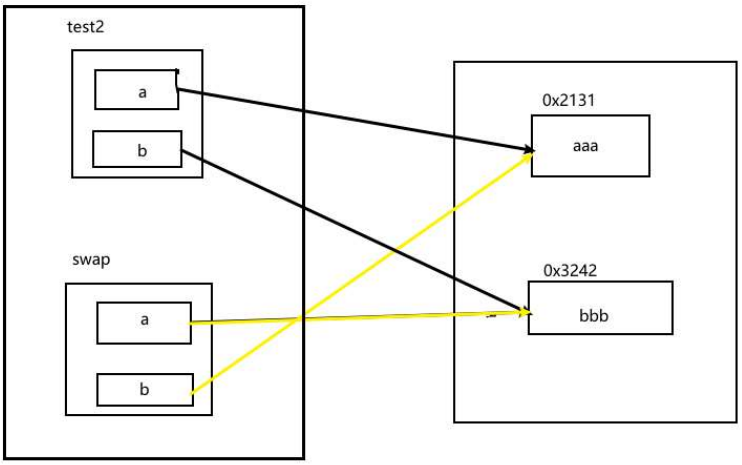
```
            arrs[i] = 8;
        }
        System.out.println("方法内部的值：" + Arrays.toString(arrs));
    }
================================================================
原来的值：[1, 255, 3, 23, 0, -43]
原来的地址：[I@61e717c2
方法获得的地址：[I@61e717c2
方法内部的值：[8, 8, 8, 8, 8, 8]
变化后的值：[8, 8, 8, 8, 8, 8]
```

1.0 地址交换

```
/**
     * java参数传递方式都是值传递（字符串地址值）
     两个局部变量交换地址，对成员变量没有半毛钱影响
     */
    @Test
    public void test2() {
        String a = "aaa";
        String b = "bbb";
        System.out.println("原值：a=" + a + " b=" + b);
        swap(a, b);
        System.out.println("更改后：a=" + a + " b=" + b);

    }

    private void swap(String a, String b) {
        String temp = a;
        a = b;
        b = temp;
        System.out.println("方法内：a=" + a + " b=" + b);
    }
================================================================
原值：a=aaa b=bbb
方法内：a=bbb b=aaa
更改后：a=aaa b=bbb
```



五、数组使用注意点

1. 为了吸引c语言的使用者，数组也可以定义为 数据类型 变量名[] =new 数据类型[x]
2. 什么类型的数组就存放什么类型的数据
3. 数组一但定义出来，其长度，类型就固定了（无论动静）
4. 动态初始化时，数组中各个元素都有默认值
5. 数组有两种遍历的方法:for、foreach(增强for)
6. 动静初始化数组不可以混着用
7. java.lang.ArrayIndexOutOfBoundsException

- 数组下标越界异常

```
    @Test
    public void test3() {
        final int[] ints = new int[4];
        System.out.println(ints[3]);
        System.out.println(ints[4]);//java.lang.ArrayIndexOutOfBoundsException:
4
        System.out.println("程序结束！");//(如果不处理异常)以下代码不会执行
    }
```

- 空指针异常

```
    @Test
    public void test4() {
        int[] ints = new int[4];
        System.out.println(ints[1]);
        ints = null;
        System.out.println(ints[1]);//java.lang.NullPointerException
        System.out.println("程序结束！");//如果不处理异常，将无法执行此代码
    }
```

问题1：如果访问的元素位置超过最大索引，执行时会出现ArrayIndexOutOfBoundsException(数组索引越界异常)

问题2：如果数组变量中没有存储数组的地址，而是null，在访问数组信息时会出现NullPointerException(空指针异常)

六、IDEA DEBUG