

第一部分：IDEA开发工具

参见：IEDA的安装请参考文件夹PPT中的 04_IDEA.ppt

1.数组

1.1 数组介绍

数组就是存储数据长度固定的容器，存储多个数据的数据类型要一致。

1.2 数组的定义格式

1.2.1 第一种格式

数据类型[] 数组名

示例：

```
1 int[] arr;  
2 double[] arr;  
3 char[] arr;
```

1.2.2 第二种格式

数据类型 数组名[]

示例：

```
1 int arr[];  
2 double arr[];  
3 char arr[];
```

1.3 数组的动态初始化

1.3.1 什么是动态初始化

数组动态初始化就是只给定数组的长度，由系统给出默认初始化值

1.3.2 动态初始化格式

```
1 数据类型[] 数组名 = new 数据类型[数组长度];
```

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

1.3.3 动态初始化格式详解

- 等号左边:
 - int:数组的数据类型
 - []:代表这是一个数组
 - arr:代表数组的名称
- 等号右边:
 - new:为数组开辟内存空间
 - int:数组的数据类型
 - []:代表这是一个数组
 - 5:代表数组的长度

代码：

```

1 package com.itheima.array;
2
3 public class Demo2Array {
4     /*
5         数组的动态初始化:
6             在初始化的时候, 需要手动指定数组的长度, 系统会为数组容器分配初始值.
7
8         动态初始化格式:
9             数据类型[] 数组名 = new 数据类型[数组的长度];
10
11        注意:
12            打印数组变量的时候, 会打印出数组的内存地址
13
14        [I@10f87f48 :
15
16            @ : 分隔符
17            [ : 当前的空间是一个数组类型
18            I : 当前数组容器中所存储的数据类型
19            10f87f48 : 十六进制内存地址
20
21            0 1 2 3 4 5 6 7 8 9 a b c d e f
22        */
23        public static void main(String[] args) {
24            // 数据类型[] 数组名 = new 数据类型[数组的长度];
25            // 通过new关键字创建了一个int类型的数组容器, 该容器可以存储5个int类型的整数, 该容器被arr数
26            // 组变量所记录
27            int[] arr = new int[5];
28            // [I@10f87f48
29            System.out.println(arr);
30
31            byte[] bArr = new byte[3];
32            // [B@b4c966a
33            System.out.println(bArr);
34        }
35    }
36

```

1.4 数组元素访问

1.4.1 什么是索引

每一个存储到数组的元素，都会自动的拥有一个编号，从0开始。

这个自动编号称为数组索引(index)，可以通过数组的索引访问到数组中的元素。

1.4.2 访问数组元素格式

```
1 数组名[索引];
```

1.4.3 示例代码

```
1  package com.itheima.array;
2
3  public class Demo3ArrayIndex {
4      /*
5          数组动态初始化:
6              初始化的时候，手动指定数组长度，系统会为数组容器分配初始值。
7
8          数组的元素访问格式:
9              数组名[索引]
10
11             索引: 数组中数据的编号方式，编号从0开始
12             作用: 访问数组容器中的空间位置
13
14         注意:
15             数组在创建完毕后，即使没有赋值，也可以取出，但取出的元素都是默认初始值。
16
17         */
18     public static void main(String[] args) {
19         int[] arr = new int[3];           // 0 1 2
20         System.out.println(arr);        // 数组的内存地址 [I@10f87f48
21
22         // 数组名[索引] 访问数组容器中的空间位置
23         System.out.println(arr[0]);     // 0 系统自动分配的默认初始值
24         System.out.println(arr[1]);
25         System.out.println(arr[2]);
26
27         System.out.println("-----");
28
29         // 数组名[索引]
30         arr[0] = 11;
31         arr[1] = 22;
32         arr[2] = 33;
33
34         System.out.println(arr[0]);
35         System.out.println(arr[1]);
36         System.out.println(arr[2]);
37     }
38 }
```

1.5 内存分配

1.5.1 内存概述

内存是计算机中的重要原件，临时存储区域，作用是运行程序。

我们编写的程序是存放在硬盘中的，在硬盘中的程序是不会运行的。

必须放进内存中才能运行，运行完毕后会清空内存。

Java虚拟机要运行程序，必须要对内存进行空间的分配和管理。

1.5.2 java中的内存分配

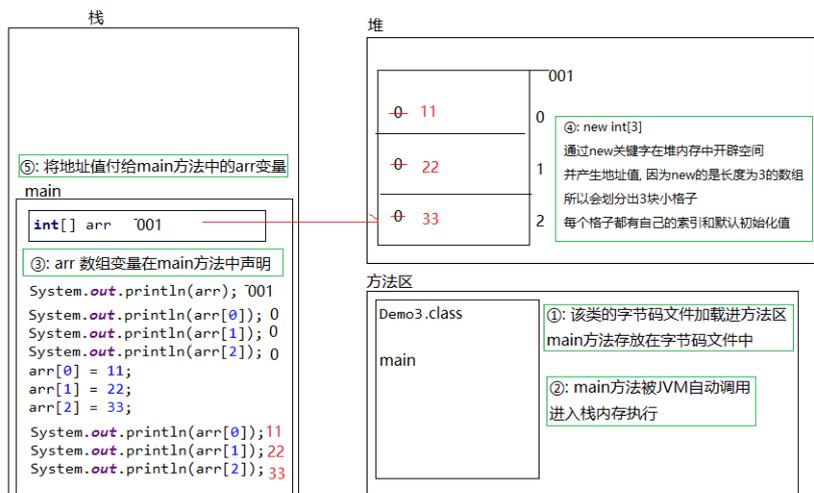
- 目前我们只需要记住两个内存，分别是：栈内存和堆内存

区域名称	作用
寄存器	给CPU使用，和我们开发无关。
本地方法栈	JVM在使用操作系统功能的时候使用，和我们开发无关。
方法区	存储可以运行的class文件。
堆内存	存储对象或者数组，new来创建的，都存储在堆内存。
方法栈	方法运行时使用的内存，比如main方法运行，进入方法栈中执行。

1.6 Java内存分配-一个数组内存图

```
public class Demo3 {  
    public static void main(String[] args) {  
        int[] arr = new int[3];  
        System.out.println(arr);  
        System.out.println(arr[0]);  
        System.out.println(arr[1]);  
        System.out.println(arr[2]);  
        arr[0] = 11;  
        arr[1] = 22;  
        arr[2] = 33;  
        System.out.println(arr[0]);  
        System.out.println(arr[1]);  
        System.out.println(arr[2]);  
    }  
}
```

栈内存：方法运行时，进入的内存，局部变量都存放于这块内存当中
堆内存：new出来的内容都会进入堆内存，并且会存在地址值
方法区：字节码文件(.class文件)加载时进入的内存



1.7 两个数组内存图

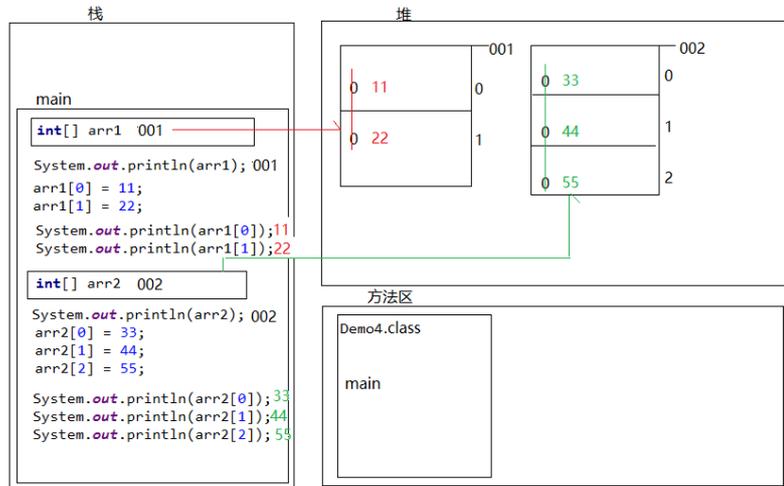
```

public class Demo4 {
    public static void main(String[] args) {
        int[] arr1 = new int[2];
        System.out.println(arr1);
        arr1[0] = 11;
        arr1[1] = 22;
        System.out.println(arr1[0]);
        System.out.println(arr1[1]);

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

        int[] arr2 = new int[3];
        System.out.println(arr2);
        arr2[0] = 33;
        arr2[1] = 44;
        arr2[2] = 55;
        System.out.println(arr2[0]);
        System.out.println(arr2[1]);
        System.out.println(arr2[2]);
    }
}

```



1.8 多个数组指向相同内存图

```

public class Demo5Array {
    public static void main(String[] args) {
        int[] arr1 = new int[2];
        arr1[0] = 11;
        arr1[1] = 22;

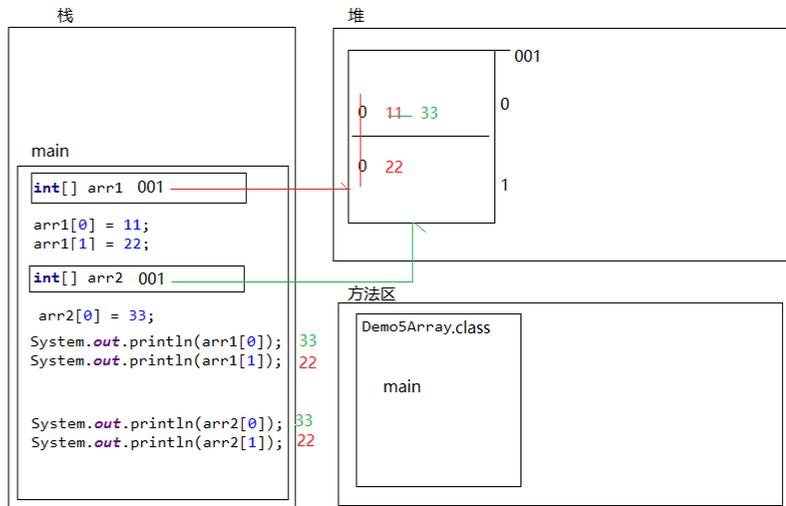
        int[] arr2 = arr1;
        arr2[0] = 33;

        System.out.println(arr1[0]);
        System.out.println(arr1[1]);

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

        System.out.println(arr2[0]);
        System.out.println(arr2[1]);
    }
}

```



1.9 数组的静态初始化

1.9.1 什么是静态初始化

在创建数组时，直接将元素确定

1.9.2 静态初始化格式

- 完整版格式

```

1 数据类型[] 数组名 = new 数据类型[] {元素1, 元素2, ...};

```

- 简化版格式

```

1 数据类型[] 数组名 = {元素1, 元素2, ...};

```

1.9.3 示例代码

```
1 package com.itheima.array2;
2
3 public class Demo1Array {
4     /*
5         数组静态初始化：初始化时指定每个数组元素的初始值，由系统决定数组长度
6
7         完整格式：
8             数据类型[] 数组名 = new 数据类型[]{数据1,数据2,数据3...};
9
10        简化格式：
11            数据类型[] 数组名 = {数据1,数据2,数据3...};
12    */
13    public static void main(String[] args) {
14        // 数据类型[] 数组名 = new 数据类型[]{数据1,数据2,数据3...};
15        int[] arr = new int[]{11,22,33};
16        System.out.println(arr[0]);
17        System.out.println(arr[1]);
18        System.out.println(arr[2]);
19
20        // 数据类型[] 数组名 = {数据1,数据2,数据3...};
21        int[] arr2 = {44,55,66};
22        System.out.println(arr2);
23        System.out.println(arr2[0]);
24        System.out.println(arr2[1]);
25        System.out.println(arr2[2]);
26    }
27 }
```

1.10 数组操作的两个常见问题

1.10.1 索引越界异常

- 出现原因

```
1 public class ArrayDemo {
2     public static void main(String[] args) {
3         int[] arr = new int[3];
4         System.out.println(arr[3]);
5     }
6 }
```

数组长度为3，索引范围是0~2，但是我们却访问了一个3的索引。

程序运行后，将会抛出`ArrayIndexOutOfBoundsException` 数组越界异常。在开发中，数组的越界异常是不能出现的，一旦出现了，就必须修改我们编写的代码。

- 解决方案

将错误的索引修改为正确的索引范围即可！

1.10.2 空指针异常

- 出现原因

```
1 public class ArrayDemo {
2     public static void main(String[] args) {
3         int[] arr = new int[3];
4
5         //把null赋值给数组
6         arr = null;
7         System.out.println(arr[0]);
8     }
9 }
```

arr = null 这行代码，意味着变量arr将不会在保存数组的内存地址，也就不允许再操作数组了，因此运行的时候会抛出 NullPointerException 空指针异常。在开发中，数组的越界异常是不能出现的，一旦出现了，就必须修改我们编写的代码。

- 解决方案

给数组一个真正的堆内存空间引用即可！

1.11 数组遍历

- 数组遍历：就是将数组中的每个元素分别获取出来，就是遍历。遍历也是数组操作中的基石。

```
1 public class ArrayTest01 {
2     public static void main(String[] args) {
3         int[] arr = { 1, 2, 3, 4, 5 };
4         System.out.println(arr[0]);
5         System.out.println(arr[1]);
6         System.out.println(arr[2]);
7         System.out.println(arr[3]);
8         System.out.println(arr[4]);
9     }
10 }
```

以上代码是可以将数组中每个元素全部遍历出来，但是如果数组元素非常多，这种写法肯定不行，因此我们需要改造成循环的写法。数组的索引是 0 到 length-1，可以作为循环的条件出现。

```
1 public class ArrayTest01 {
2     public static void main(String[] args) {
3         //定义数组
4         int[] arr = {11, 22, 33, 44, 55};
5
6         //使用通用的遍历格式
7         for(int x=0; x<arr.length; x++) {
8             System.out.println(arr[x]);
9         }
10    }
11 }
```

1.12 数组获取最大值

- 最大值获取：从数组的所有元素中找出最大值。
- 实现思路：
 - 定义变量，保存数组0索引上的元素
 - 遍历数组，获取出数组中的每个元素
 - 将遍历到的元素和保存数组0索引上值的变量进行比较
 - 如果数组元素的值大于了变量的值，变量记录住新的值
 - 数组循环遍历结束，变量保存的就是数组中的最大值
- 代码实现：

```
1 package com.itheima.test;
2
3 import java.util.Scanner;
4
5 public class Test2Array {
6     /*
7         需求：从数组中查找最大值
8
9         int[] arr = {12,45,98,73,60};
10
11        实现步骤：
12            1. 假设数组中的第一个元素为最大值
13            2. 遍历数组，获取每一个元素，准备进行比较
14            3. 如果比较的过程中，出现了比max更大的，让max记录更大的值
15            4. 循环结束后，打印最大值.
16        */
17     public static void main(String[] args) {
18         int[] arr = {12,45,98,73,60};
19         // 1. 假设数组中的第一个元素为最大值
20         int max = arr[0];
21         // 2. 遍历数组，获取每一个元素，准备进行比较
22         for(int i = 1; i < arr.length; i++){
23             // 3. 如果比较的过程中，出现了比max更大的，让max记录更大的值
24             if(arr[i] > max){
25                 max = arr[i];
26             }
27         }
28         // 4. 循环结束后，打印最大值.
29         System.out.println("max:" + max);
30     }
31 }
32
```

1.13 数组元素求和

- 需求：键盘录入5个整数，存储到数组中，并对数组求和

- 思路：1.创建键盘录入对象，准备键盘录入 2.定义一个求和变量，准备记录累加后的结果 3.动态初始化一个长度为5的int数组，准备存储键盘录入的数值 4.将键盘录入的数值存储到数组中 5.遍历数组，取出每一个元素，并求和 6.输出总和
- 代码实现：

```

1  package com.itheima.test;
2
3  import java.util.Scanner;
4
5  public class Test3Array {
6      /*
7          需求：键盘录入5个整数，存储到数组中，并对数组求和
8
9          思路：
10         1.创建键盘录入对象，准备键盘录入
11         2.定义一个求和变量，准备记录累加后的结果
12         3.动态初始化一个长度为5的int数组，准备存储键盘录入的数值
13         4.将键盘录入的数值存储到数组中
14         5.遍历数组，取出每一个元素，并求和
15         6.输出总和
16     */
17     public static void main(String[] args) {
18         // 1.创建键盘录入对象，准备键盘录入
19         Scanner sc = new Scanner(System.in);
20         // 2.定义一个求和变量，准备记录累加后的结果
21         int sum = 0;
22         // 3.动态初始化一个长度为5的int数组，准备存储键盘录入的数值
23         int[] arr = new int[5];
24         // 4.将键盘录入的数值存储到数组中
25         for(int i = 0; i < arr.length; i++){
26             System.out.println("请输入第" + (i+1) + "个整数:");
27             //arr[i] = 10;
28             arr[i] = sc.nextInt();
29         }
30
31         // 5.遍历数组，取出每一个元素，并求和
32         for (int i = 0; i < arr.length; i++) {
33             sum += arr[i];
34         }
35
36         // 6.输出总和
37         System.out.println("sum:" + sum);
38
39     }
40 }
41

```

1.14 数组基本查找【应用】

- 需求：已知一个数组 arr = {19, 28, 37, 46, 50}; 键盘录入一个数据，查找该数据在数组中的索引，并在控制台输出找到的索引值。

- 思路：1.定义一个数组，用静态初始化完成数组元素的初始化 2.键盘录入要查找的数据，用一个变量接收 3.定义一个索引变量，初始值为-1 4.遍历数组，获取到数组中的每一个元素 5.拿键盘录入的数据和数组中的每一个元素进行比较，如果值相同，就把该值对应的索引赋值给索引变量，并结束循环 6.输出索引变量
- 代码实现：

```

1 public static void main(String[] args) {
2     // 1.定义一个数组，用静态初始化完成数组元素的初始化
3     int[] arr = {19, 28, 37, 46, 50};
4     // 2.键盘录入要查找的数据，用一个变量接收
5     Scanner sc = new Scanner(System.in);
6     System.out.println("请输入您要查找的元素:");
7     int num = sc.nextInt();
8     // 3.定义一个索引变量，初始值为-1
9     // 假设要查找的数据，在数组中就是不存在的
10    int index = -1;
11    // 4.遍历数组，获取到数组中的每一个元素
12    for (int i = 0; i < arr.length; i++) {
13        // 5.拿键盘录入的数据和数组中的每一个元素进行比较，如果值相同，就把该值对应的索引赋
        值给索引变量，并结束循环
14        if(num == arr[i]){
15            // 如果值相同，就把该值对应的索引赋值给索引变量，并结束循环
16            index = i;
17            break;
18        }
19    }
20    // 6.输出索引变量
21    System.out.println(index);
22 }
23 }
24

```

1.15 评委打分【应用】

- 需求：在编程竞赛中，有6个评委为参赛的选手打分，分数为0-100的整数分。选手的最后得分为：去掉一个最高分和一个最低分后的4个评委平均值(不考虑小数部分)。
- 思路：1.定义一个数组，用动态初始化完成数组元素的初始化，长度为6 2.键盘录入评委分数 3.由于是6个评委打分，所以，接收评委分数的操作，用循环 4.求出数组最大值 5.求出数组最小值 6.求出数组总和 7.按照计算规则进行计算得到平均分 8.输出平均分
- 代码实现：

```

1 public static void main(String[] args) {
2     // 1.定义一个数组，用动态初始化完成数组元素的初始化，长度为6
3     int[] arr = new int[6];
4     // 2.键盘录入评委分数
5     Scanner sc = new Scanner(System.in);
6     // 3.由于是6个评委打分，所以，接收评委分数的操作，用循环
7     for (int i = 0; i < arr.length; i++) {
8         System.out.println("请输入第" + (i+1) + "个评委的打分:");
9         int score = sc.nextInt();
10        if(score >= 0 && score <= 100){

```

```
11         // 合法的分值
12         arr[i] = score;
13     }else{
14         // 非法的分值
15         System.out.println("您的打分输入有误, 请检查是否是0-100之间的");
16         i--;
17     }
18 }
19
20 // 4. 求出数组最大值
21 int max = arr[0];
22 for (int i = 1; i < arr.length; i++) {
23     if(max < arr[i]){
24         max = arr[i];
25     }
26 }
27
28 // 5. 求出数组最小值
29 int min = arr[0];
30 for (int i = 1; i < arr.length; i++) {
31     if(min > arr[i]){
32         min = arr[i];
33     }
34 }
35
36 // 6. 求出数组总和
37 int sum = 0;
38 for (int i = 0; i < arr.length; i++) {
39     sum += arr[i];
40 }
41
42 // 7. 按照计算规则进行计算得到平均分
43 int avg = (sum - max - min ) / 4;
44
45 // 8. 输出平均分
46 System.out.println(avg);
47 }
48 }
```