0.IDEA开发工具

参见：IDEA工具安装详解.pdf

1.数组

1.1什么是数组【理解】

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数组定义格式【记忆】

1.2.1第一种

数据类型[] 数组名

示例：

```
int[] arr;
double[] arr;
char[] arr;
```

1.2.2第二种

数据类型 数组名[]

示例：

```
int arr[];
double arr[];
char arr[];
```

1.3数组动态初始化【应用】

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

```
数据类型[] 数组名 = new 数据类型[数组长度];
```

```
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

1.4数组元素访问【应用】

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

```
数组名[索引];
```

1.4.3示例代码

```
public class ArrayDemo {
    public static void main(String[] args) {
        int[] arr = new int[3];

        //输出数组名
        System.out.println(arr); //[I@880ec60

        //输出数组中的元素
        System.out.println(arr[0]);
        System.out.println(arr[1]);
        System.out.println(arr[2]);
    }
}
```

1.5内存分配【理解】

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。

1.5.2java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单个数组的内存图【理解】

1.7多个数组的内存图【理解】

1.8多个数组指向相同内存图【理解】

1.9数组静态初始化【应用】

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

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

- 简化版格式

```
数据类型[] 数组名 = {元素1,元素2,...};
```

1.9.3示例代码

```
public class ArrayDemo {
    public static void main(String[] args) {
        //定义数组
        int[] arr = {1, 2, 3};

        //输出数组名
        System.out.println(arr);

        //输出数组中的元素
        System.out.println(arr[0]);
        System.out.println(arr[1]);
```

```
        System.out.println(arr[2]);
    }
}
```

1.10数组操作的两个常见小问题【应用】

1.10.1索引越界异常

- 出现原因

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

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，就必须要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

```
public class ArrayDemo {
    public static void main(String[] args) {
        int[] arr = new int[3];

        //把null赋值给数组
        arr = null;
        System.out.println(arr[0]);
    }
}
```

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，就必须要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【应用】

- 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。

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

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 lenght-1 ，可以作为循环的条件出现。

```
public class ArrayTest01 {
    public static void main(String[] args) {
        //定义数组
        int[] arr = {11, 22, 33, 44, 55};

        //使用通用的遍历格式
        for(int x=0; x<arr.length; x++) {
            System.out.println(arr[x]);
        }
    }
}
```

1.12数组最值【应用】

- 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：

```
public class ArrayTest02 {
    public static void main(String[] args) {
        //定义数组
        int[] arr = {12, 45, 98, 73, 60};

        //定义一个变量，用于保存最大值
        //取数组中第一个数据作为变量的初始值
        int max = arr[0];

        //与数组中剩余的数据逐个比对，每次比对将最大值保存到变量中
        for(int x=1; x<arr.length; x++) {
            if(arr[x] > max) {
                max = arr[x];
```

```
            }
        }

        //循环结束后打印变量的值
        System.out.println("max:" + max);

    }
}
```