# Day 05

2018年5月4日 8:38

课程回顾

1.多重if结构

if(条件){

}else if(条件){

}...

if嵌套的一个特例

2.switch 等值/定值判断

default的位置 ,是否添加break;

break;

switch(表达式的值) byte short int char enum String

3.switch和if区别

switch等值判断

if等值判断,范围判断

4.循环 ,一定注意循环次数尽量减少

while 适用有循环次数 ,也适用不确定循环次数 while(true)

do-while 适用先循环一次,再判断

for(;;) 适用有循环次数

for(;;) 等同 while(true)

5.读懂一个循环规律,推三次循环

6.break 退出switch和循环

continue 结束本次循环,继续下一次循环

7.debug调试 F5 F6 F8

恢复对应视图当中标签

window--perspective---reset perspective

数组:是一种数据结构

特点:只能存储相同数据类型的数据

且在内存中开辟连续内存空间

数组的定义:

考虑三件事:

1.是否声明

2.是否有空间,是否申请内存空间

3.如果有空间初值是否正确

声明了,是没有内存空间,但代码中是可以使用,但运行报错

申请内存空间了,一定声明了,在代码中和运行时可以使用不会报错

但是注意空间的数据初值是什么

- 数组的声明:

语法:

类型[] 数组名称;

说明:

类型可以是java中已知的类型,

可以是八种基本数据类型

也可以是程序员事先或之前已经定义好的其他的类类型

[]代表是一个数组,[]放在类型的后面,也可以放在数组名称的后面,但不建议

数组名称,符合java的标识符的规则

比如:

int[] scores; 声明准备在内存中开辟空间,但不知道开辟多少个int空间

-数组的开辟内存空间

语法:

类型[] 数组名 =new 类型[数组的大小/长度]

注意:一旦给内存申请指定个数的内存空间后,就不能改变数据的长度的

开辟内存空间前提一定是声明过了

可以先声明,后开辟内存空间,再后给初值

```
int[] scores;
scores=new int[5];
```

也可以在声明的同时,开辟内存空间并给赋值初值

```
int[] scores=new int[5]
```

整型 初值就是0

-数组赋值初值

不用默认值的情况下,给数组赋值初值

循环遍历数组的每一个元素,并给赋值初值

-声明的同时直接申请空间并赋值初值,不能分多行写

```
int[] scores=new int[]{100,0,60,80,90};
```

如果给了初值就new 类型[什么都不能写]{值1,值2,...}

int[] scores={100,0,60,80,90}

下面的写法是错的,

int[] scores=new int[];
scores={100,0,60,80,90};

int[] scores;
scores={100,0,60,80,90};

数组元素的使用:

语法

数组名称[下标]

说明:

下标是从0开始,不能超越数组的长度

最后一个数组元素的下标是数组长度-1

如果超范围报异常

java.lang.ArrayIndexOutOfBoundsException:

数组 下标索引超出范围 异常

数组在内存的表示:

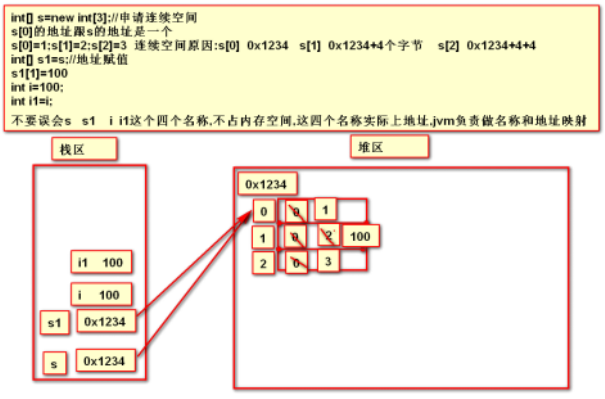
java中的内存会分为5个部分

方法区,堆区,栈区,本地方法栈,寄存器

方法区:存储的类的定义,静态的内容,常量池

堆区:存储类的对象,数组的空间等

栈区:对象或数组的首地址,存基本数据类型的数据




总结:

对于基本数据类型(八种) 只要 基本类型 变量名; 就是开辟空间 int i;//有空间无默认值

对于引用数据类型(数组,接口,类)

以数组为例子

int[] s;//没内存空间　堆里和栈里都没有

int[] s=new int[5];//有内存空间,堆里和栈里都有,有默认值

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

//有内存空间,堆里和栈里都有,有指定的初始数据值

数组的应用:

获取数组的长度:数组实际的开辟空间元素的个数

数组的名称.length

遍历数组,一定循环,而且for循环居多

获取数组中的最大值

数组的排序

```
//排序
for (int i = 1; i < nums.length; i++) { // 控制轮数和选择的下标
    for (int j = 1 + i; j <= nums.length; j++) { // 控制次数和比 较的下标
        if (nums[i - 1] > nums[j - 1]) {
            int temp = nums[i - 1];
            nums[i - 1] = nums[j - 1];
            nums[j - 1] = temp;
        }
    }
}
```

第一次外循环:

i=1　i<7　1<7　　i++　　i=2

内第一次 j=1+i　2　2<=7　　nums[0]>nums[1]　j++　j=3

内第二次 j=3　3<=7　nums[0]>nums[2]　　j++　　j=4

内第三次 j=4　4<=7　nums[0]>nums[3]　　j++　　j=5

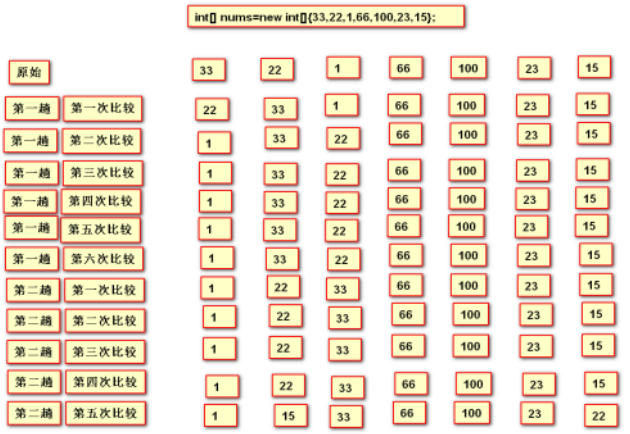
第二次外循环:

i=2　i<7　2<7

内第一次 j=1+i　3　3<=7　　nums[1]>nums[2]　j++　j=4

内第二次 j=3　3<=7　nums[1]>nums[3]　　j++　　j=5

内第三次 j=4　4<=7　nums[1]>nums[4]　　j++　　j=6




数组的反转

数组的查找,折半查找 前提必须有序

数组的复制

System.arraycopy(src, srcPos, dest, destPos, length);

说明:

src:原数组

srcPos:原数组的位置

dest:目标数组

destPos:目标数组中的位置

length:长度

跟数组的长和位置有关,否则报数组下标越界

数组的扩/缩容量

类型[] 新数组的名称= java.util.Arrays.copyOf(original, newLength)

original:原始的数组

newLength:新数组的长度

新数组的长度大于原数组的长度,多余的元素是默认值

新数组的长度小于原数组的长度,

补充:

输出数据对象结果 [I@2a139a55

[ I @ 2a139a55

数据　数组元素类型　　间隔符　十六进制数据是内存地址的哈希码

增强for循环

语法:

```
for(类型 变量名 : 数组或集合){
    //循环体代码块
 }
```

说明:

特性:只有遍历完了才推出循环

从数组或集合中取出第一数据 赋值给冒号前面的变量,

然后进入循环体使用变量的值,

然后回到for,从数组或集合中取出下一个元素赋值给冒号前面的变量

然后进入循环体使用变量的值

以此,循环往复,直到数组或集合中的数据全部遍历完毕

类型 变量名　每次循环都是重新申请变量的空间　for(int i:　scores){}

不会保留上一次变量空间和值

数组或集合:一批数据,且数组或集合中的数的类型必须跟前面类型一致

在方法中声明一个变量,不使用是可以的,这个变量没有值,有内存空间

声明一个变量但没给初值,就使用,会报

The local variable i may not have been initialized

本地 变量　I 可能没有　　被　　初始化

结论:方法中任何变量的定义,必须给初值

后面会学到,在类中定义变量可以不给初值就使用,因为给默认值

在同一个方法中不可以定义同名变量

Duplicate local variable I

重复　　本地变量　　i