Day 06

2018年5月7日 8:34

课程回顾

数组:

为什么要有数组:单个变量只能一个值,用数组可以存储多个相同类型数据

数组就是一个数据结构

可以通过一个数组的名字和下标来确定一个具体数据

特点:存储同一个类型数据,在内存中开辟连续的内存空间

确定元素,数组的地址+下标*元素的类型占有的字节数

定义数组:

1.只有声明 int[] scores;没有内存空间,肯定没有值

2.有声明有空间 int[] scores=new int[5] 有内存有空间,有默认值

3.有声明有空间有指定的初值

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

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

绝对不能分行写

定义一个数组或者创建一个对象

1.声明 没有内存空间 ,写代码的时候是可以使用声明的数组,运行出错

2.new过 有内存空间,是否默认值

3.是否赋值指定的初值

使用数组 :数组名[下标]

下标:0-----数组长度-1

超范围报异常ArrayIndexOutofBoundException

遍历数组,一般情况用for循环(普通,增强)

数组的应用:

长度:数组名称.length;

数组的遍历

数组的排序,

数组的反转,

数组的复制

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

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

二维数组:

本质:就是由多个一维数组组成的

二维数组的定义:

语法:

数据类型[][] 数组名称;

数据类型[][] 数组名称=new 数据类型[大小][大小];

数据类型[][] 数组名称={{一维数组},{一维数组},{一维数组}};

说明:可以是八种基本数据类型,也可以系统定义好的类类型,程序员自定义类型

数组名称符合标识符的规则

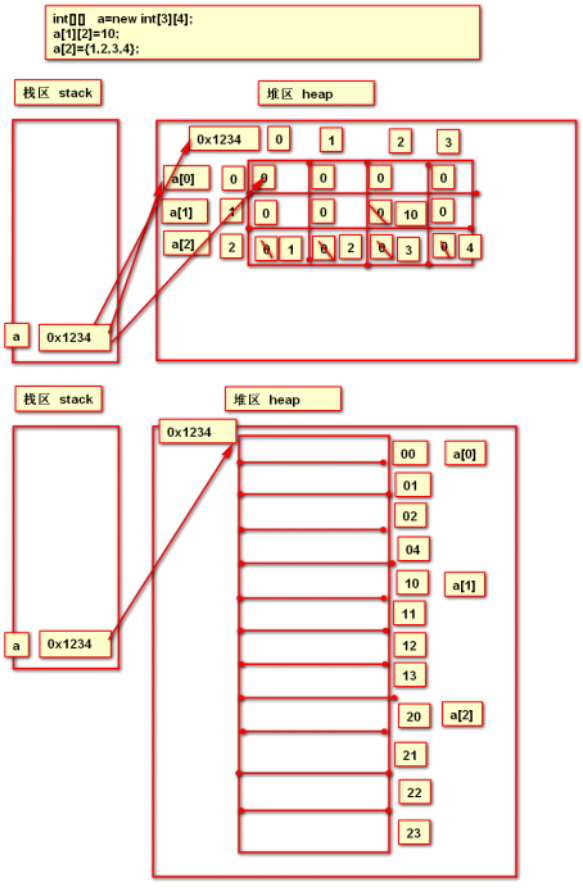
比如:

int[][] a=new int[3][4];

a的地址和a[0]的地址和a[0][0]一样

a[0] --->	a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1]--->	a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2]--->	a[2][0]	a[2][1]	a[2][2]	a[2][3]

二维数组在内存中的表示



使用一个二维数组

int[][] scores=new int[3][4];

scores.length; //3

scores[0].length;//4
scores[1].length;//4
scores[2].length;//4

参考案例

常规: int[][] scores;没有空间,没有数据

int[][] scores=new int[3][4];有空间,有数据,默认数据

int[][] scores={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

二维数组的几个特殊的用法:

```
int[][] scores=new int[3][];
public static void main(String[] args) {
    int[][] a=new int[3][];
    System.out.println(a);//数据的hash码,说明有地址
    System.out.println(a[0]);//null
    System.out.println(a.length);//3
    System.out.println(a[0].length);//NullPointerException
    System.out.println(a[0][0]);//NullPointerException
}
```

int[][] scores=new int[][];//语法错误的写法

int[][] scores=new int[][]{{1,2,3,4},{5,6,7,8},{9,10,11,12}};//正确的

int[][] scores=new int[][]{{1,2},{5,6,7,8},{9}};//占用了7个空间

```
int[][] scores=new int[][]{{1,2},{5,6,7,8},{9}};
for(int i=0;i<scores.length;i++){
    System.out.println("第"+(i+1)+"次外循环");
    for(int j=0;j<scores[i].length;j++){
        System.out.println(scores[i][j]);
    }
}
```

int[][] scores={{1,2},{5,6,7,8},{9}};//占用了7个空间

java中的方法:

为什么要有方法,提高复用性,

复用率越高的方法,这个方法的功能就越单一

面想对象的设计原则之一,单一职责

一个软件功能要单一

一个项目功能要单一

一个模块功能要单一

一个package包功能要单一

一个类或接口功能要单一

一个方法功能要单一

一个变量功能要单一

一个循环功能要单一

一个分支功能要单一

极其困难的事,单一的度的问题,尽量单一

方法的语法:

语法:

带有方法体的方法,说明这方法有实现内容

访问修饰符 方法的返回类型 方法的名称(方法的参数列表){

//方法体

return 返回的类型

}

没有方法体的方法,这个方法就是仅仅是方法声明,没有功能

访问修饰符 方法的返回类型 方法的名称(方法的参数列表);

说明:

访问修饰符:

public,private,friendly(不写,默认),protected;

方法的返回类型:八种基本数据类型,系统的类类型,自定义的类类型

特殊的返回类型 void 解释成无返回类型

方法的名称:符合java的标识符规则,符合驼峰命名法 ,在一个类中名称唯一

方法的参数列表:可以有0个或多个参数,用逗号间隔

方法体:代表某个功能的若干代码的组成

return 返回的数据;

在方法中任何地方都可以出现return这个词语,只要出现这个词语的地方,终止当前的方法的执行,无论return后面是否有代码,继续方法后面代码,在方法可以多次出现return,根据业务,最终只能有一个return执行

return写法一

return 返回的数据;在终止当前方法的同时,返回一个数据

返回的数据只能有一个

1. return i,j;//错误写法,不能返回多个数据,编译错
2. int[] a=new int[10];
   return a;//正确
3. Student stu=new Student();

return stu;//正确

返回的数据的类型必须跟方法的返回类型一致

return 写法二

return; 方法的返回类型必须是void

如果方法的返回类型的是void,那么方法体中可以写return;(终止方法),可以放在任意位置

也可以不写return;就是正常执行完方法里的所有语句

演示定义方法,调用方法,方法中的变量的作用域

方法中的变量叫法:局部变量,本地变量,可以定义,不给初值就使用会报错

The local variable i may not have been initialized

本地 变量 ı 可能没有 被 初始化

DemoMethod1.java

```
1 package cn.tedu.day06;
2 
3 public class DemoMethod1 {
4     public int sum(int a,int b){
5         int result=a+b;
6         return result;
7     }
8 
9     public static void main(String[] args) {
10        DemoMethod1 dm1=new DemoMethod1();
11        int result=dm1.sum(2, 3);
12        System.out.println("result="+result);
13    }
14
15 }
```

有方法了,就要关注变量作用域

两个result变量是不同,因为隶属于不同的方法有不同作用域范围
result 5行开始 7行前
result 11行开始13行前
参数的作用域范围,从参数的位置方法的右大括号前

定义方法 调用方法,返回类型是系统默认的类类型

```
public class DemoMethod2 {
    public String sayHello(String name){

        return "hello "+name+"!";
    }
    public String hello(){
        return "大家好!";
    }
    public String hello1(){
        return "你好!";
    }
    public static void main(String[] args) {
        DemoMethod2 dm2=new DemoMethod2();
        String result=dm2.sayHello("zhangsan");
        System.out.println(result);
        System.out.println(dm2.hello());
        dm2.hello1();
    }
```

方法的返回值是系统的类型,有return 数据类型,数据类型跟方法方法的返回值类型一致
如果方法有返回值,程序员可以接收这个返回值,
也可以不接收这个返回值

没有接收返回值

方法中可以放置多个return

```
     * 如果传递参数是100以内就直接返回这个值
     * 否则,给这个值减去100,再返回
     * @param a
     * @return
     */
    public int method1(int a){
        if(a<100){
            return a;
        }else{
            return a-100;
        }
        //System.out.println("方法结束");//此代码永远不会执行
    }
    public static void main(String[] args) {
        DemoMethod3 dm3=new DemoMethod3();
        System.out.println(dm3.method1(10));
        System.out.println(dm3.method1(110));

    }
```

方法中可以有多个return,但最终根据条件,只有一个执行

Unreachable code
无法触及的代码

有关于无法触及代码的问题

```
package cn.tedu.day06;

public class DemoMethod4 {
    public int method1(){
        int i=0;
        while(true){
            System.out.println(i++);
            if(i==100){
                return i;
            }
        }
        System.out.println("程序结束");//Unreachable code
    }
    public void method2(){
        int i=0;
        while(true){
            System.out.println(i++);
            if(i==100){
                break;
            }
        }
        System.out.println("程序结束");
    }
    public int method3(){
        int i=0;
        while(true){
            System.out.println(i++);
        }
        System.out.println("程序结束");//Unreachable code
    }
    public static void main(String[] args) {
        DemoMethod4 dm4=new DemoMethod4();

    }

}
```

因为while(true),不是因为return,也不是因为if

while(true)检测到有break;可以终止死循环

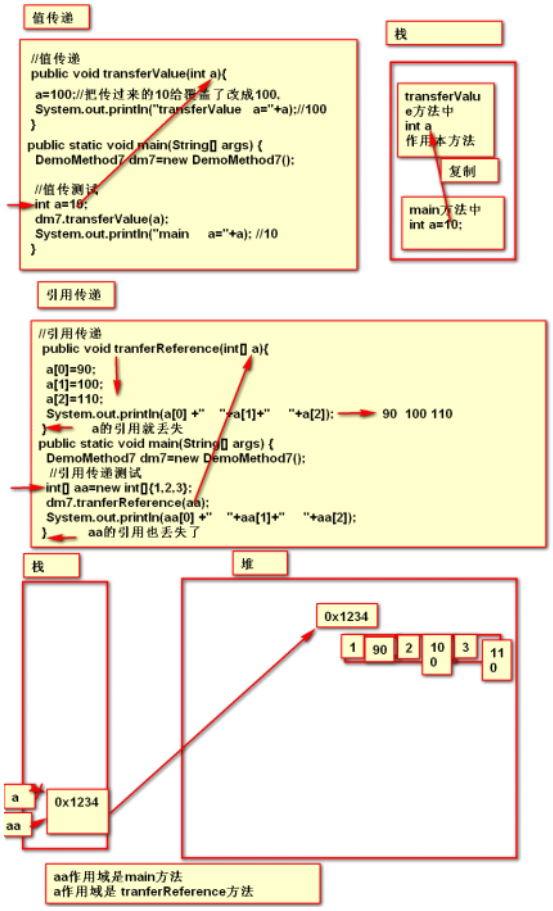
因为while(true)

```java
package cn.tedu.day06;

public class DemoMethod5 {
    public void method1(int a){
        int sum=0;
        for(int i=0;i<a;i++){
            if(i==20){
                return;
            }
            sum+=i;
        }
        System.out.println("sum="+sum);//不会执行这句话
    }
    public void method2(int a){
        int sum=0;
        for(int i=0;i<a;i++){
            if(i==20){
                break;
            }
            sum+=i;
        }
        System.out.println("sum="+sum);
    }
    public static void main(String[] args) {
        DemoMethod5 dm5=new DemoMethod5();
        dm5.method1(100);//没有结果
        dm5.method2(100);
    }

}
```

做方法的目的,提高复用率

参数的传递分类:只要给方法传递参数,要么是值传递,要么引用传递
值传递:方法的参数传递的基本数据类型,传递的一个副本数值

引用传递:方法的参数是引用数据类型(数组,接口的对象,类的对象),传递的时自己本身



break;continue;return;的区别:

Break;终止当前switch和循环

Continue:终止本次循环继续下一次循环

Return:终止当前方法

补充内容:

方法的可变参数

可变参数只能放在方法的参数列表中的最后的位置,只能有一个

可变参数就是特殊的数组

public void method(类型 ... 参数名称)

```
package cn.tedu.day06;

public class DemoMethod6 {
    //给参数赋值更灵活和方便
    public int sum(int...a){
        int s=0;
        for(int i : a){
            s+=i;
        }
        return s;
    }
    public static void main(String[] args) {
        DemoMethod6 dm6=new DemoMethod6();
        System.out.println(dm6.sum());
        System.out.println(dm6.sum(1));
        System.out.println(dm6.sum(1,2));
        System.out.println(dm6.sum(1,2,3));
        System.out.println(dm6.sum(new int[]{1,2,3,4}));
    }

}
```