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

计算机生成了可选文字:
int00 a-new int131141; 
a [ 1 ] [ 2 ] 0 ； 
a [ 2 ] 粼 1 之 3 } ； 
栈 区 stack 
0x1234 
堆 区 heap 
回 国 国 国 
0x1234 
彩 硐 
a 
a 
堆 区 heap 
0x1234 
00 
02 
20 
22 
a101 
alll 
a121 
栈 区 stack 
0x1234 

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 I 可能没有 被 初始化

计算机生成了可选文字:
国 DemoMeth0d1java 
b)f 
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 1 1 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 
1 
2 
3 
4 彐 
5 
6 
7 
8 
9 彐 
19 
11 
12 
13 
14 
15 
acka e c n ． ted u ． d a 96 ． 
有 方 法 了 ， 就 要 关 注 变 量 作 用 域 
public int sum(int a,int 
int result—a+b 
return 
public static void main(StringtJ 
args) { 
DemoMethod1 dml:new DemoMethod1( ） ； 
int result—dm1.sum(), 3 ） ； 
System.out · printIn( " result:' +result) ； 

定义方法 调用方法,返回类型是系统默认的类类型

计算机生成了可选文字:
public class DemoMet hod2 { 
public String sayHe110(String 
return " he110 ' " ． 
public String he1100 { 
return " 大 家 好 ！ " ． 
public String he11010 { 
return 你 好 ！ " ． 
name)f 
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 
public static void main(StringtJ args) { 
DemoMethod2 dm2:new DemoMethod2( ） ， 
String 
· sayHe110( " zhangsan " ） ； 
System.out.println(result), 
System.out.print1n(dm2 
· hello()), 
dm2 ． he1101 （ ） ， 
没 有 接 收 返 回 值 

方法中可以放置多个return

计算机生成了可选文字:
如 果 传 递 参 数 是 199 以 内 就 直 接 返 回 这 个 值 
否 则 ， 给 这 个 值 减 去 199 ， 再 返 回 
@param a 
@returnl 
方 法 中 可 以 有 多 个 return, 但 最 终 
根 据 条 件 ， 只 有 一 个 执 行 
此 代 码 永 远 不 会 执 行 
public int methodl(int a)f 
if(aKIO)f 
return a 
)elsef 
return a 
一 199 ； 
System.out.println 方 法 结 束 
public static void main(StringLJ args) { Unreachable code 
无 法 触 及 的 代 码 
DemoMethod3 dm3:new DemoMethod3( ） ； 
System.out.print1n(dm3.method1(1Ø)), 
System.out.print1n(dm3.method1(11Ø)); 

有关于无法触及代码的问题

计算机生成了可选文字:
package cn.tedu.dayØ6; 
public class DemoMet hod4 { 
public int methodl( ） { 
int i:e; 
while(true)f 
因 为 while(true), 不 是 因 
为 return, 也 不 是 因 为 if 
System.out · println(i++), 
if(i::10)f 
return i; 
〖 WY*,$,*å);//l-Jnreachable code 
public void method2( ） { 
int i:e; 
while(true)f 
System.out · println(i++); 
if(i::10)f 
break, 
System.out.println( " 程 序 结 束 " ） ； 
public int method3()f 
int i:e; 
while(true)f 
System.out · println(i++); 
while(true) 检 测 到 有 break; 
可 以 终 止 死 循 环 
因 为 while(true) 
、 〖 WY*,$,*å);//l-Jnreachable code 
public static void main(StringtJ args) { 
DemoMet hod4 d m4:new DemoMethod4( ） ； 

计算机生成了可选文字:
package cn.tedu.dayØ6; 
public class DemoMet hod 5 { 
public void methodl(int a)f 
int S u m 
for(int i:e;i«a,i++)f 
if(i::2Ø)f 
return; 
sum+:l; 
System.out .println( "sum:' + sum ） ； / / 不 会 执 行 这 句 话 
public void method2(int a)f 
int S u m 
for(int i:e;i«a,i++)f 
if(i::2Ø)f 
break, 
sum+:l; 
System.out · println( " sum: 
+SUm); 
public static void main(StringtJ args) { 
DemoMethod5 dm5:new DemoMethod5( ） ； 
dm5.method1(10) ， / / 没 有 结 果 
dm5.method2(10); 

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

计算机生成了可选文字:
值 传 
值 传 
public void transferValue(int 司 { 
a 00 彩 把 传 过 来 的 10 给 皿 了 改 成 100 ， 
System.out.println("tr 
ferValue a ' + 司 ； / / 100 
public static void ain(StringO args) { 
DemoMethod m7:new DemoMethod7(); 
栈 
transferValu 
e 方 法 中 
Int 
作 方 法 
复 制 
main 法 中 
Int 
90 100 1 10 
值 传 测 ． 
•nt a: 1 
dm7.transferValue(a); 
System.out.println("main 
引 用 传 
引 用 传 
a: + 司 ； / / 10 
'+a121); 
public void tranferReference(intO 司 { 
a [ 0 ] 0 ； 
a [ 1 ] 00 ； 
a [ 2 ] 10 ； 
System.out.println(alOl + 
a 的 引 用 就 丢 失 
public static void main(Strin 
' alll+' 
args) { 
2 
10 
0 
3 
0 
DemoMethod7 dm7:new moMethod7(); 
引 用 传 测 试 
into aa:new intO(1,2,3); 
dm7.tranferReference( ） ； 
System.out.println(aalOl + 
aa 的 引 用 也 丢 失 了 
0x1234 
aa 作 用 域 是 main 方 法 
+aalll+ 
堆 
'+aa121) ； 
0x1234 
1 90 
栈 
aa 
a 作 用 域 是 tranferReference 方 法 

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)

计算机生成了可选文字:
package cn.tedu.dayØ6; 
public class DemoMet hod6 { 
/ / 给 参 数 值 更 灵 活 和 方 便 
public int sum(int...a)f 
int s:e; 
for(int i 
a)f 
return S ； 
public static void main(StringtJ args) { 
DemoMethod6 dm6:new DemoMethod6( ） ； 
System.out 
System.out 
System.out 
System.out 
System.out 
· print1n(dm6 
· print1n(dm6 
· print1n(dm6 
· print1n(dm6 
.print1n(dm6 
． sum （ 1 ， 2 ， 3)) ； 
.sum(new i 砒 [ ] { 1 ， 2 ， 3 ， 4 } )) ； 