

P169-171数组扩容与缩减

22:55

2021年5月6日

数组添加

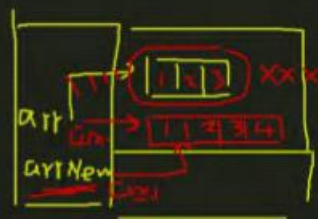
要求: 实现动态的给数组添加元素效果, 实现对数组扩容。ArrayAdd.java

- 1) 原始数组使用静态分配 `int[] arr = {1,2,3}`
- 2) 增加的元素, 直接放在数组的最后 `arr = {1,2,3,4}`
`arrNew = {1,2,3,4}`
- 3) 用户可以通过如下方法来决定是否继续添加, 添加成功, 是否继续? y/n

课后练习题: ArrayReduce.java

有一个数组 {1, 2, 3, 4, 5}, 可以将该数组进行缩减, 提示用户是否继续缩减, 每次缩减最后哪个元素。当只剩下最后一个元素, 提示, 不能再缩减。

```
int[] arr = {1,2,3};
int[] arrNew = new int[arr.length + 1];
//遍历 arr 数组, 依次将arr的元素拷贝到 arrNew数组
for(int i = 0; i < arr.length; i++) {
    arrNew[i] = arr[i];
}
//把4赋给arrNew最后一个元素
arrNew[arrNew.length - 1] = 4;
//让 arr 指向 arrNew,
arr = arrNew;
```

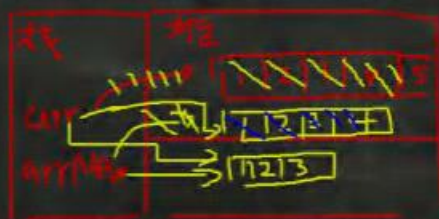


课后练习题: ArrayReduce.java

有一个数组 {1, 2, 3, 4, 5}, 可以将该数组进行缩减, 提示用户是否继续缩减, 每次缩减下最后一个元素, 提示, 不能再缩减。

ArrayAdd02.java

2min->试试 2min思考->试试



课后练习题: ArrayReduce.java

有一个数组 {1, 2, 3, 4, 5}, 可以将该数组进行缩减, 提示用户是否继续缩减, 每次缩减最后那个元素。当只剩下最后一个元素, 提示, 不能再缩减。

P172-174冒泡排序

2021年5月7日

9:17

排序的介绍

排序是将一群数据，依指定的顺序进行排列的过程。

排序的分类：

1. 内部排序：

指将需要处理的所有数据都加载到内部存储器中进行排序。包括(交换式排序法、选择式排序法和插入式排序法)；

2. 外部排序法：

数据量过大，无法全部加载到内存中，需要借助外部存储进行排序。包括(合并排序法和直接合并排序法)。

冒泡排序 (Bubble Sorting) 的基本思想是：通过对待排序序列从后向前（从下标较大的元素开始），依次比较相邻元素的值，若发现逆序则交换，使值较大的元素逐渐从前移向后部，就象水底下的气泡一样逐渐向上冒。



分析冒泡排序

数组 [24,69,80,57,13]

第1轮排序：目标把最大数放在最后

第1次比较[24,69,80,57,13]

第2次比较[24,69,80,57,13]

第3次比较[24,69,57,80,13]

第4次比较[24,69,57,13,80]

第2轮排序：目标把第二大数放在倒数第二位置

第1次比较[24,69,57,13,80]

第2次比较[24,57,69,13,80]

第3次比较[24,57,13,69,80]

第3轮排序：目标把第3大数放在倒数第3位置

第1次比较[24,57,13,69,80]

第2次比较[24,13,57,69,80]

第4轮排序：目标把第4大数放在倒数第4位置

第1次比较[13,24,57,69,80]

总结冒泡排序特点

1. 我们一共有5个元素
 2. 一共进行了 4轮排序, 可以看成是外层循环
 3. 每1轮排序可以确定一个数的位置, 比如第1轮排序确定最大数, 第2轮排序, 确定第2大的数位置, 依次类推
 4. 当进行比较时, 如果前面的数大于后面的数, 就交换
 5. 每轮比较在减少 4->3->2->1
- 分析思路->代码..

P175-176数组查找及二维数组初窥

2021年5月7日

16:54

查找

● 介绍:

在java中, 我们常用的查找有两种:

1. 顺序查找 **SeqSearch.java**
2. 二分查找【二分法, 我们放在算法讲解】

● 案例演示:

1. 有一个数列: 白眉鹰王、金毛狮王、紫衫龙王、青翼蝠王猜数游戏: 从键盘中任意输入一个名称, 判断数列中是否包含此名称【顺序查找】 要求: 如果找到了, 就提示找到, 并给出下标值。
2. 请对一个有序数组进行二分查找 {1,8, 10, 89, 1000, 1234}, 输入一个数看看该数组是否存在此数, 并且求出下标, 如果没有就提示"没有这个数"。

多维数组-二维数组

多维数组我们只介绍二维数组。

● 二维数组的应用场景

比如我们开发一个五子棋游戏, 棋盘就需要二维数组来表示。如图:



二维数组的使用

● 快速入门案例:

TwoDimensionalArray01.java

请用二维数组输出如下图形

```
0 0 0 0 0
0 0 1 0 0
0 2 0 3 0
0 0 0 0 0
```

//老韩解读

//1. arr[i] 表示 二维数组的第 i+1 个元素 比如 arr[0]: 二维数组的第一个元素

//2. arr[i].length 得到 对应的 每个一维数组的长度

p177-180二维数组内存布局及使用

2021年5月7日

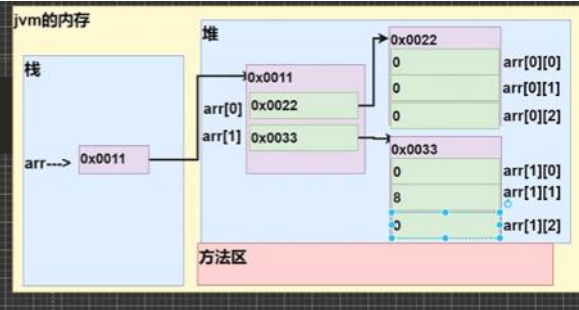
17:35

二维数组的使用

● 使用方式1: 动态初始化

TwoDimensionalArray02.java

- 1) 语法: 类型 `[][]` 数组名 = new 类型 [大小] [大小]
- 2) 比如: `int a[][] = new int[2][3]`
- 3) 使用演示
- 4) 二维数组在内存的存在形式



二维数组的使用

● 使用方式2: 动态初始化

- 1) 先声明: 类型 数组名 `[][]`; **TwoDimensionalArray02.java**
- 2) 再定义(开辟空间) 数组名 = new 类型 [大小] [大小]
- 3) 赋值(有默认值, 比如int 类型的就是0)
- 4) 使用演示

● 使用方式3: 动态初始化-列数不确定

- 1) 看一个需求: 动态创建下面二维数组, 并输出。

j \ i	j = 0	j = 1	j = 2
i = 0	1		
i = 1	2	2	
i = 2	3	3	3

- 2) 完成该案例
- 3) 画出执行分析示意图

二维数组的使用

● 使用方式4: 静态初始化

TwoDimensionalArray04.java

1. 定义 类型 数组名 `[][]` = {{值1,值2...},{值1,值2...},{值1,值2...}}
2. 使用即可 [固定方式访问]

比如:

```
int[][] arr = {{1,1,1}, {8,8,9}, {100}};
```

解读

1. 定义了一个二维数组 arr
2. arr有三个元素(每个元素都是一维数组)
3. 第一个一维数组有3个元素, 第二个一维数组有 3个元素, 第三个一维数组有1个元素

P181-184二维数组练习

2021年5月7日

23:20

二维数组的遍历

● 案例:

TwoDimensionalArray05.java

`int arr[][] = {{4,6},{1,4,5,7},{-2}};` 遍历该二维数组, 并得到和

二维数组的应用案例

使用二维数组打印一个 10 行杨辉三角 **YangHui.java**

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
....
```

【提示】

1. 第一行有 1 个元素, 第 n 行有 n 个元素
2. 每一行的第一个元素和最后一个元素都是 1
3. 从第三行开始, 对于非第一个元素和最后一个元素的元素值, `arr[i][j] = arr[i-1][j] + arr[i-1][j-1];` //必须找到这个规律

二维数组课堂练习

● 声明: `int[] x,y[];` 以下选项允许通过编译的是():

说明: x 是int类型一维数组, y是int类型的二维数组

- a) `x[0] = y;` //错误 `int[] -> int`
- b) `y[0] = x;` //正确 `int[] -> int[]`
- c) `y[0][0] = x;` //错误 `int[] -> int`
- d) `x[0][0] = y;` //错误 `x[0][0]` 是错误
- e) `y[0][0] = x[0];` //正确 `int -> int`
- f) `x = y;` //错误 `int[] -> int[]`

二维数组使用细节和注意事项

1. 一维数组的声明方式有:

`int[] x` 或者 `int x[]`

2. 二维数组的声明方式有:

`int[][] y` 或者 `int[] y[]` 或者 `int y[][]`

3. 二维数组实际上是由多个一维数组组成的, 它的各个一维数组的长度可以相同, 也可以不相同。比如: `map[][]` 是一个二维数组

`map [][] = {{1,2},{3,4,5}}`

由 `map[0]` 是一个含有两个元素的一维数组, `map[1]` 是一个含有三个元素的一维数组构成, 我们也称为列数不等的二维数组。

P185-190第六章作业+内容梳理

11:19

2021年5月8日

本章作业

1. 下面数组定义正确的有 **Homework01.java**
A. `String strs[] = { 'a', 'b', 'c' }; //error, char ->String`
B. `String[] strs = { "a", "b", "c" }; //ok`
C. `String[] strs = new String{"a" "b" "c"}; //error`
D. `String strs[] = new String[] {"a", "b", "c"}; //ok`
E. `String[] strs = new String[3] {"a", "b", "c"}; //error, 编译不通过`

2. 写出结果 **Homework02.java**

```
String foo="blue";
boolean[] bar=new boolean[2];
if(bar[0]){
    foo="green";
}
System.out.println(foo);
```

本章作业

3. 以下Java代码的输出结果为 ()。 **Homework03.java**

```
int num=1;
while(num < 10) {
    System.out.println(num);
    if(num>5) {
        break;
    }
    num+=2;
}
```

4. 已知有个升序的数组，要求插入一个元素，该数组顺序依然是升序，比如：
[10, 12, 45, 90]，添加23后，数组为 [10, 12, 23, 45, 90]

Homework04.java

本章作业

5. 随机生成10个整数(1 100的范围)保存到数组，并倒序打印以及求平均值、求最大值和最大值的下标、并查找里面是否有 8 **Homework05.java**

6. 试写出以下代码的打印结果 **Homework06.java**

```
char[] arr1={'a','z','b','c'};
char[] arr2=arr1;
arr1[2]='韩';

for(int i=0;i<arr2.length;i++){
    System.out.println(arr1[i]+"," + arr2[i]);
}
```

7. 写出冒泡排序的代码 **Homework07.java**


P191-195类与对象、内存布局

2021年5月8日 11:25

类与对象

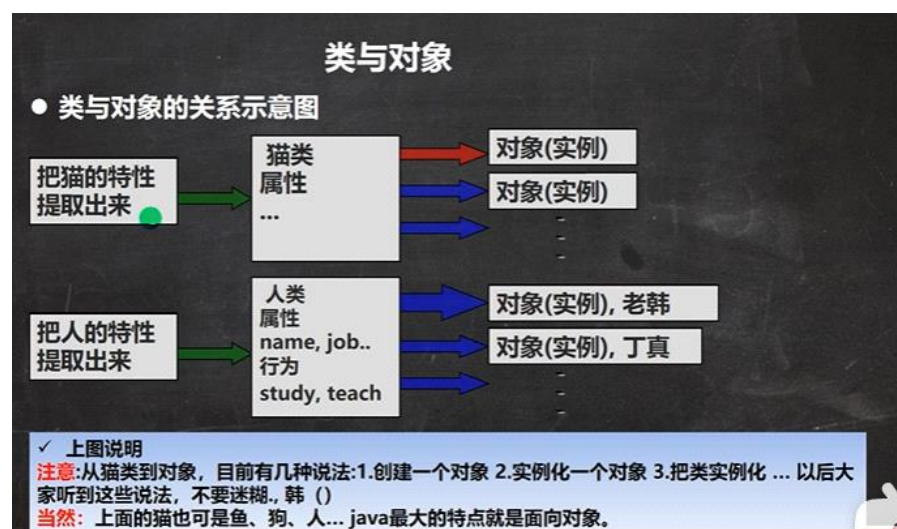
- 看一个养猫猫问题

张老太养了两只猫猫:一只名字叫小白,今年3岁,白色。还有一只叫小花,今年100岁,花色。请编写一个程序,当用户输入小猫的名字时,就显示该猫的名字,年龄,颜色。如果用户输入的小猫名错误,则显示 张老太没有这只猫猫。



类与对象

一个程序就是一个世界,有很多事物(对象[属性,行为])

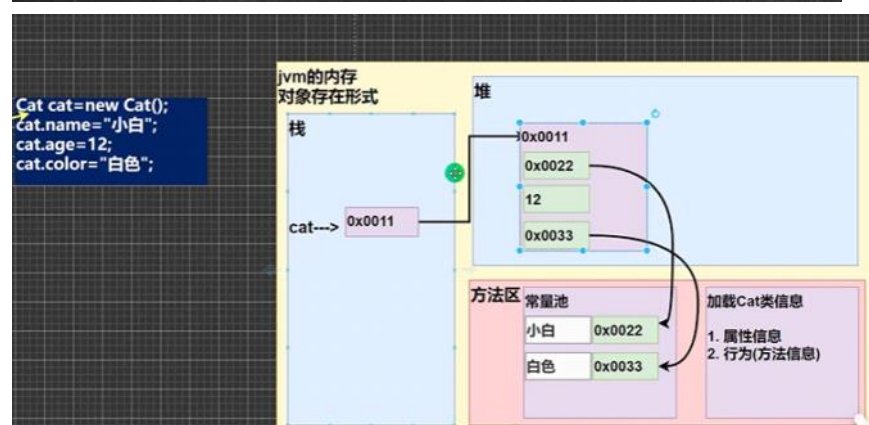


类与对象

- 类和对象的区别和联系

通过上面的案例和讲解我们可以看出:

- 1) 类是抽象的,概念的,代表一类事物,比如人类,猫类..., 即它是数据类型.
- 2) 对象是具体的,实际的,代表一个具体事物,即是实例.
- 3) 类是对象的模板,对象是类的一个个体,对应一个实例



P196-201属性细节与对象概念

2021年5月8日 11:26

类与对象

- 属性/成员变量
- ✓ 基本介绍

1. 从概念或叫法上看: 成员变量 = 属性 = field (即 成员变量是用来表示属性的, 授课中, 统一叫 属性)
案例演示: Car(name,price,color)

2. 属性是类的一个组成部分, 一般是基本数据类型, 也可能是引用类型(对象, 数组)。比如我们前面定义猫类 的 int age 就是属性

注意事项和细节说明

PropertiesDetail.java

- 1) 属性的定义语法同变量, 示例: 访问修饰符 属性类型 属性名;
这里老师简单的介绍访问修饰符: 控制属性的访问范围
- 有四种访问修饰符 public, protected, 默认, private ,后面我会详细介绍
- 2) 属性的定义类型可以为任意类型, 包含基本类型或引用类型
- 3) 属性如果不赋值, 有默认值, 规则和数组一致。具体说: int 0, short 0, byte 0, long 0, float 0.0, double 0.0, char \u0000, boolean false, String null
案例演示: [Person类]

类与对象

- 如何创建对象

1. 先声明再创建
`Cat cat ; //声明对象 cat`
`cat = new Cat(); //创建`
2. 直接创建
`Cat cat = new Cat();`

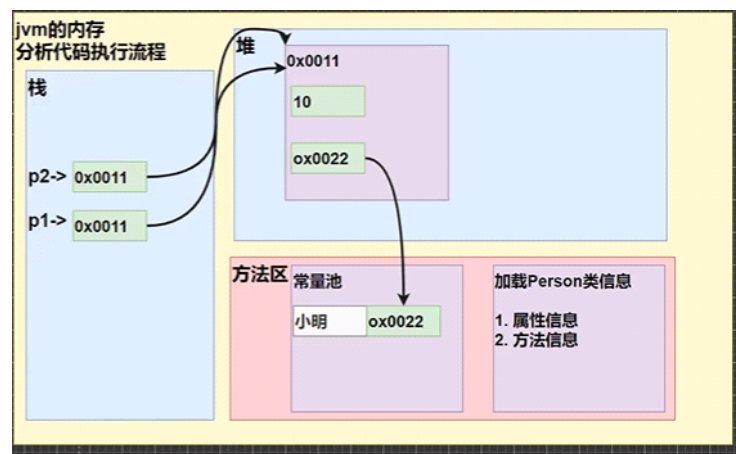


如何创建对象

1. 先声明再创建
`Cat cat ; //声明对象 cat`
`cat = new Cat(); //创建`
2. 直接创建
`Cat cat = new Cat();`

类与对象

- 类和对象的内存分配机制(重要)
- ✓ 看一个思考题
我们定义一个人类(Person)(包括 名字,年龄)。 (Object03.java)
我们看看下面一段代码:
`Person p1=new Person();`
`p1.age=10;`
`p1.name="小明";`
• `Person p2=p1; //把p1 赋给了 p2`
`System.out.println(p2.age);`
请问:p2.age究竟是多少? 并画出内存图:



成员方法

● 基本介绍

在某些情况下，我们需要定义成员方法(简称方法)。比如人类:除了有一些属性外(年龄, 姓名..), 我们人类还有一些行为比如:可以说话、跑步.., 通过学习, 还可以做算术题。这时就要用成员方法才能完成。现在要求对Person类完善。

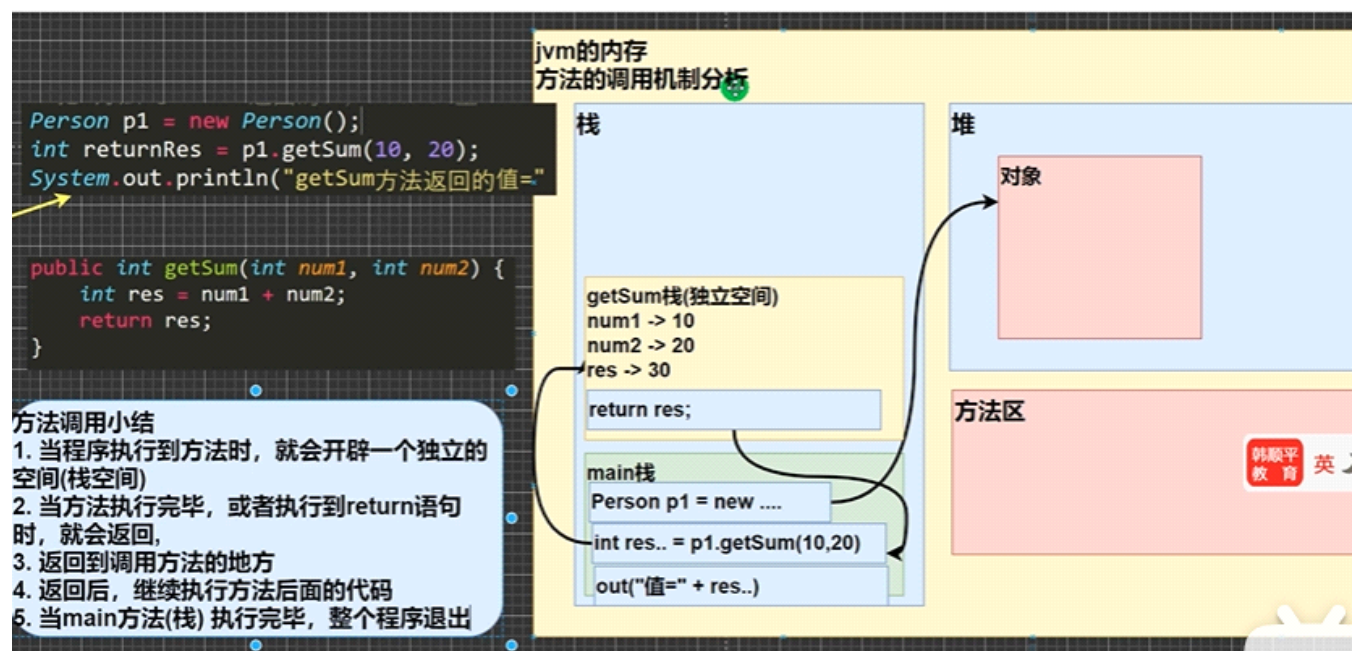
● 成员方法快速入门

Method01.java

- 1) 添加speak 成员方法, 输出 我是一只好人
- 2) 添加cal01 成员方法, 可以计算从 1+...+1000的结果
- 3) 添加cal02 成员方法, 该方法可以接收一个数n, 计算从 1+...+n 的结果
- 4) 添加getSum成员方法, 可以计算两个数的和

● 方法的调用机制原理: (重要!-示意图)

提示: 画出程序执行过程[cal02]+说明



P205-P209方法的定义与使用

2021年5月9日 16:37

成员方法

- 为什么需要成员方法

Method02.java

- 看一个需求:

请遍历一个数组, 输出数组的各个元素值。

- ✓ 解决思路1, 传统的方法, 就是使用单个for循环, 将数组输出, 大家看看问题是什么?
- ✓ 解决思路2: 定义一个类 MyTools ,然后写一个成员方法, 调用方法实现,看看效果又如何。

```
int [][] map =  
{{0,0,1},{1,1,1},{1,1,3}};
```

- 成员方法的好处

- ✓ 提高代码的复用性
- ✓ 可以将实现的细节封装起来, 然后供其他用户来调用即可。

成员方法

- 成员方法的定义

```
public 返回数据类型 方法名 (参数列表..) { //方法体  
    语句;  
    return 返回值;  
}
```

1. 参数列表: 表示成员方法输入 cal(int n)
2. 数据类型 (返回类型): 表示成员方法输出, void 表示没有返回值
3. 方法主体: 表示为了实现某一功能代码块
4. return 语句不是必须的。
5. 老韩提示: 结合前面的题示意图, 来理解

成员方法

- 注意事项和使用细节

MethodDetail.java

- ✓ 访问修饰符 (作用是控制 方法使用的范围)
如果不写默认访问, [有四种: public, protected, 默认, private], 具体在后面说
- ✓ 返回数据类型
 1. 一个方法最多有一个返回值 [思考, 如何返回多个结果 返回数组]
 2. 返回类型可以为任意类型, 包含基本类型或引用类型(数组, 对象)
 3. 如果方法要求有**返回数据类型**, 则方法体中最后的执行语句必须为 **return 值**; 而且要求返回值类型必须和return的值类型一致或兼容
 4. 如果方法是**void**, 则方法体中可以没有return语句, 或者 只写 return ;
- ✓ 方法名
遵循驼峰命名法, 最好见名知义, 表达出该功能的意思即可, 比如 得到两个数的和 getSum, 开发中按照规范

成员方法

- 注意事项和使用细节

- 形参列表

1. 一个方法可以有0个参数, 也可以有多个参数, 中间用逗号隔开, 比如 getSum(int n1,int n2)
2. 参数类型可以为任意类型, 包含基本类型或引用类型, 比如 printArr(int[][] map)
3. 调用带参数的方法时, 一定对应着参数列表传入相同类型或兼容类型 的参数! [getSum]
4. 方法定义时的参数称为形式参数, 简称形参; 方法调用时的传入参数称为实际参数, 简称实参, 实参和形参的类型要一致或兼容、个数、顺序必须一致! [演示]

- 方法体

里面写完成功能的具体的语句, 可以为输入、输出、变量、运算、分支、循环、方法调用, 但里面不能再定义方法! 即: 方法不能嵌套定义。[演示]

P210-211方法练习题与传参机制

2021年5月9日 20:26

成员方法

- 注意事项和使用细节

MethodDetail02.java

✓ 方法调用细节说明

1. 同一个类中的方法调用：直接调用即可。比如 `print(参数);`
案例演示：A类 `sayOk` 调用 `print()`
2. 跨类中的方法A类调用B类方法：需要通过对象名调用。比如 `对象名.方法名(参数);` 案例演示：B类 `sayHello` 调用 `print()`
3. 特别说明一下：跨类的方法调用和方法的访问修饰符相关，先暂时这么提一下，后面我们讲到访问修饰符时，还要再细说。

成员方法

- 课堂练习题

MethodExercise01.java

1. 编写类AA，有一个方法：判断一个数是奇数odd还是偶数，返回boolean
2. 根据行、列、字符打印 对应行数和列数的字符，比如：行：4，列：4，字符：#，则打印相应的效果

老师建议：一定要自己写一遍，不要嫌太简单。

P212-214方法传参机制, 克隆对象

2021年5月9日 20:58

成员方法传参机制

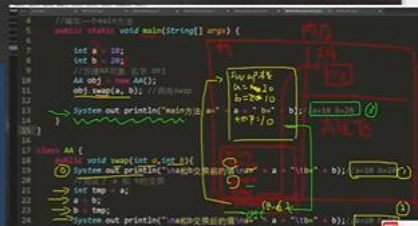
方法的传参机制对我们今后的编程**非常重要**，一定要搞的清清楚楚明明白白。我们通过案例来学习

● 基本数据类型的传参机制

MethodParameter01.java

1) 看一个案例, 分析结果是什么?

```
public void swap(int a,int b){
    int tmp = a;
    a = b;
    b = tmp;
    System.out.println("a="+a+"\tb="+b);
}
```



2) 结论及示意图

基本数据类型，传递的是值（值拷贝），形参的任何改变不影响实参！

猿平教育

成员方法传参机制

● 引用数据类型的传参机制

1. 看一个案例 MethodParameter02.java

B类中编写一个方法test100，可以接收一个数组，在方法中修改该数组，看看原来的数组是否变化？

B类中编写一个方法test200，可以接收一个Person(age,sal)对象，在方法中修改该对象属性，看看原来的对象是否变化？



2. 结论及示意图

引用类型传递的是地址（传递也是值，但是值是地址），可以通过形参影响实参！

3. 在看一个案例，下面的方法会对原来的对象有影响吗？

p=null 和 p = new Person(); 对应示意图

成员方法传参机制

- 成员方法返回类型是引用类型应用实例

MethodExercise02.java

- 1) 编写类MyTools类，编写一个方法可以打印二维数组的数据。
- 2) 编写一个方法copyPerson，可以复制一个Person对象，返回复制的对象。克隆对象，注意要求得到新对象和原来的对象是两个独立的对象，只是他们的属性相同

```
//注意要求得到新对象和原来的对象是两个独立的对象，只是他们的属性相同
//
//编写方法的思路
//1. 方法的返回类型 Person
//2. 方法的名字 copyPerson
//3. 方法的形参 (Person p)
//4. 方法体，创建一个新的对象，并复制属性，返回即可
public Person copyPerson(Person p){
    //创建一个新的对象
    Person p2 = new Person();
    p2.name = p.name; //把原来对象的姓名赋给p2.name
    p2.age = p.age; //把原来对象的年龄赋给p2.age
    return p2;
}
```


P215-218递归执行机制

2021年5月10日 14:02

方法递归调用

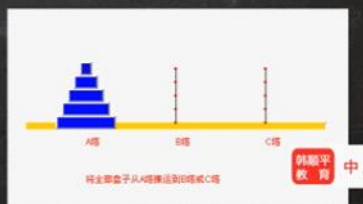
● 基本介绍

简单的说: 递归就是方法自己调用自己,每次调用时传入不同的变量.递归有助于编程者解决复杂问题,同时可以让代码变得简洁

方法递归调用

● 递归能解决什么问题?

1. 各种数学问题如: 8皇后问题, 汉诺塔, 阶乘问题, 迷宫问题, 球和篮子的问题(google编程大赛)[简单演示]
2. 各种算法中也会使用到递归, 比如快排, 归并排序, 二分查找, 分治算法等.
3. 将用栈解决的问题-->递归代码比较简洁



方法递归调用

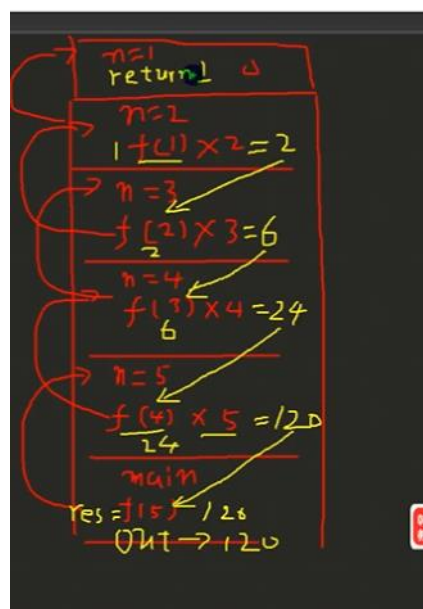
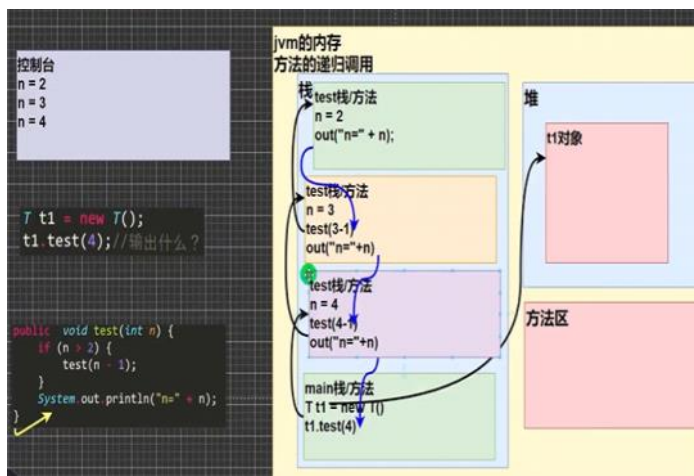
● 递归举例

列举两个小案例,来帮助大家理解递归调用机制

1. 打印问题
2. 阶乘问题

```
//输出什么?  
public void test(int n) {  
    if (n > 2) {  
        test(n - 1);  
    }  
    System.out.println("n=" + n);  
}
```

```
//阶乘  
public int factorial(int n) {  
    if (n == 1) {  
        return 1;  
    } else {  
        return factorial(n - 1) * n;  
    }  
}
```



方法递归调用

递归重要规则

1. 执行一个方法时，就创建一个新的受保护的独立空间(栈空间)
2. 方法的局部变量是独立的，不会相互影响，比如n变量
3. 如果方法中使用的是引用类型变量(比如数组)，就会共享该引用类型的数据。
4. 递归必须向退出递归的条件逼近，否则就是无限递归,出现StackOverflowError，死龟了:)
5. 当一个方法执行完毕，或者遇到return，就会返回，遵守谁调用，就将结果返回给谁，同时当方法执行完毕或者返回时，该方法也就执行完毕。

方法递归调用

课堂练习

RecursionExercise01.java

1. 请使用递归的方式求出斐波那契数1,1,2,3,5,8,13...给你一个整数n，求出它的值是多
2. 猴子吃桃子问题：有一堆桃子，猴子第一天吃了其中的一半，并再多吃了一个！以后每天猴子都吃其中的一半，然后再多吃一个。当到第10天时，想再吃时（即还没吃），发现只有1个桃子了。问题：最初共多少个桃子？

韩耀平
教育 英

```
class M2{
    public int peach(int day){
        if(day == 10){
            return 1;
        }else if (day >= 1 && day <= 9){
            return ((peach( day: day + 1) +1) * 2);
        }else {
            System.out.println("day应该在1-10");
            return -1;
        }
    }
}
```

```
class M{
    public int fibonacci(int n){
        if(n >= 1){
            if(n == 1 || n == 2){
                return 1;
            }else {
                return fibonacci( n: n -1) + fibonacci( n: n -2);
            }
        }else{
            System.out.println("请输入大于1的数字");
            return -1;
        }
    }
}
```

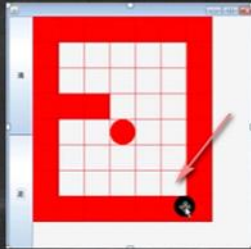

P222-227 老鼠出迷宫、汉诺塔、八皇后

2021年5月10日 23:12

方法递归调用

- 递归调用应用实例-迷宫问题

1. 小球得到的路径, 和程序员设置的找路策略有关即: 找路的上下左右的顺序相关
2. 再得到小球路径时, 可以先使用(下右上左), 再改成(上右下左), 看看路径是不是有变化
3. 测试回溯现象
4. 扩展思考: 如何求出最短路径?




方法递归调用

- 递归调用应用实例-汉诺塔

✓ 汉诺塔传说

汉诺塔: 汉诺塔 (又称河内塔) 问题是源于印度一个古老传说的益智玩具。大梵天创造世界的时候做了三根金刚石柱, 在一根柱子上从下往上按照大小顺序摞着64片圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定, 在小圆盘上不能放大圆盘, 在三根柱子之间一次只能移动一个圆盘。

假如每秒钟移动一次, 共需多长时间呢? 移完这些金片需要5845.54亿年以上, 太阳系的预期寿命据说也就是数百亿年。真的过了5845.54亿年, 地球上的一切生命, 连同梵塔、庙宇等, 都早已经灰飞烟灭



方法递归调用

- 递归调用应用实例-八皇后

先尝试做, 后面老师讲解]

✓ 八皇后问题说明

八皇后问题, 是一个古老而著名的问题, 是回溯算法的典型实例。该问题是国际西洋棋棋手马克斯·贝瑟尔于1848年提出: 在8×8格的国际象棋上摆放八个皇后, 使其不能互相攻击, 即: 任意两个皇后都不能处于同一行、同一列或同一斜线上, 问有多少种摆法。



http://360.6822.com/www19/play_76277



方法递归调用

- 递归调用应用实例-八皇后问题

✓ 八皇后思路分析, 老韩说下

- 1) 第一个皇后先放第一行第一列
- 2) 第二个皇后放在第二行第一列, 然后判断是否OK, 如果不OK, 继续放在第二列、第三列、依次把所有列都放完, 找到一个合适
- 3) 继续第三个皇后, 还是第一列、第二列.....直到第8个皇后也能放在一个不冲突的位置, 算是找到了一个正确解
- 4) 当得到一个正确解时, 在栈回溯到上一个栈时, 就会开始回溯, 即将第一个皇后, 放到第一列的所有正确解, 全部得到
- 5) 然后回头继续第一个皇后放第二列, 后面继续循环执行 1,2,3,4 的步骤 【示意图】

说明: 理论上应该创建一个二维数组来表示棋盘, 但是实际上可以通过算法, 用一个一维数组即可解决问题。arr[8] = {0, 4, 7, 5, 2, 6, 1, 3} //对应arr 下标 表示第几行, 即第几个皇后, arr[i] = val, val 表示第i+1个皇后, 放在第i+1行的第val+1列



P228-233方法重载讲解及练习

2021年5月11日

1:23

方法重载(OverLoad)

- 基本介绍
java中允许同一个类中, 多个同名方法的存在, 但要求 形参列表不一致!
比如: `System.out.println();` `out`是`PrintStream`类型
- 重载的好处
 - 1) 减轻了起名的麻烦
 - 2) 减轻了记名的麻烦

方法重载(OverLoad)

- 注意事项和使用细节
 - 1) 方法名: 必须相同
 - 2) 形参列表: 必须不同 (参数类型或个数或顺序, 至少有一样不同, 参数名无要求)
 - 3) 返回类型: 无要求

方法的重载(OverLoad)

- 课堂练习题
 1. 判断题:
与`void show(int a, char b, double c){}`构成重载的有: [☐ b ☐ c ☐ d ☐ e ☐]
 - a) `void show(int x, char y, double z){}` //不是
 - b) `int show(int a, double c, char b){}` //是
 - c) `void show(int a, double c, char b){}` //是
 - d) `boolean show(int c, char b){}` //是
 - e) `void show(double c){}` //是
 - f) `double show(int x, char y, double z){}` //不是
 - g) `void shows(){}` //不是

方法的重载(OverLoad)

1. 编写程序, 类`Methods`中定义三个重载方法并调用。方法名为`m`。三个方法分别接收一个`int`参数、两个`int`参数、一个字符串参数。分别执行平方运算并输出结果, 相乘并输出结果, 输出字符串信息。在主类的`main()`方法中分别用参数区别调用三个方法。 `OverLoadExercise.java`
2. 在`Methods`类, 定义三个重载方法`max()`, 第一个方法, 返回两个`int`值中的最大值, 第二个方法, 返回两个`double`值中的最大值, 第三个方法, 返回三个`double`值中的最大值, 并分别调用三个方法。

P234-236 可变参数

2021年5月11日 12:20

可变参数

- 基本概念

java允许将同一个类中**多个同名同功能**但**参数个数不同**的方法，封装成一个方法。就可以通过可变参数实现

- 基本语法

访问修饰符 返回类型 方法名(数据类型... 形参名) {
}

- 快速入门案例(VarParameter01.java)

看一个案例 类 HspMethod, 方法 sum 【可以计算 2个数的和, 3个数的和, 4. 5, ...】

可变参数

- 注意事项和使用细节

VarParameterDetail.java

1) 可变参数的实参可以为0个或任意多个。

2) 可变参数的实参可以为数组。

3) 可变参数的本质就是数组。

4) 可变参数可以和普通类型的参数一起放在形参列表，但必须保证可变参数在最后

5) 一个形参列表中只能出现一个可变参数

```
public int sum(String str,int... a,String... s){//
```

可变参数

- 课堂练习

VarParameterExercise.java

有三个方法，分别实现返回姓名和两门课成绩(总分)，返回姓名和三门课成绩(总分)，返回姓名和五门课成绩(总分)。封装成一个可变参数的方法

类名 HspMethod 方法名 showScore

P237-239作用域基本使用

2021年5月11日 12:44

作用域

● 基本使用

面向对象中，变量作用域是**非常重要**知识点，相对来说不是特别好理解，请大家注意听，认真思考，要求深刻掌握变量作用域。Scope01.java

1. 在java编程中，主要的变量就是属性(成员变量)和局部变量。
2. 我们说的局部变量一般是指在成员方法中定义的变量。【举例 Cat类: cry】
3. java中作用域的分类
全局变量：也就是属性，作用域为整个类体 Cat类: cry eat 等方法使用属性
【举例】
局部变量：也就是除了属性之外的其他变量，作用域为定义它的代码块中！
4. 全局变量可以不赋值，直接使用，因为有默认值，局部变量必须赋值后，才能使用，因为没有默认值。【举例】

```
public class VarScope {  
    //编写一个main方法  
    public static void main(String[] args) {  
    }  
}  
class Cat {  
    //全局变量：也就是属性，作用域为整个类体 Cat类: cry eat 等方法使用属性  
    //属性在定义时，可以直接赋值  
    int age = 10; //指定的值是 10  
  
    public void cry() {  
        //1. 局部变量一般是指在成员方法中定义的变量  
        //2. n 和 name 就是局部变量  
        //3. n 和 name的作用域在 cry方法中  
        int n = 10;  
        String name = "jack";  
        System.out.println("在cry中使用属性 age=" + age);  
    }  
}
```

作用域

● 注意事项和细节使用

VarScopeDetail.java

1. 属性和局部变量可以重名，访问时遵循就近原则。
2. 在同一个作用域中，比如在同一个成员方法中，两个局部变量，不能重名。【举例】
3. 属生命周期较长，伴随着对象的创建而创建，伴随着对象的死亡而死亡。局部变量，生命周期较短，伴随着它的代码块的执行而创建，伴随着代码块的结束而死亡。即在一次方法调用过程中。

作用域

● 注意事项和细节使用

4. 作用域范围不同
全局变量/属性：可以被本类使用，或其他类使用（通过对象调用）
局部变量：只能在本类中对应的方法中使用
5. 修饰符不同
全局变量/属性可以加修饰符
局部变量不可以加修饰符

P240-245构造器介绍

2021年5月13日

18:51

构造方法/构造器

● 看一个需求

我们来看一个需求：前面我们在创建人类的对象时，是先把一个对象创建好后，再给他的年龄和姓名属性赋值，如果现在我要求，在创建人类的对象时，就直接指定这个对象的年龄和姓名，该怎么做？这时就可以使用构造器。

● 基本语法

[修饰符] 方法名(形参列表){
方法体;
}

老韩说明：

- 1) 构造器的修饰符可以默认，也可以是public protected private
- 2) 构造器没有返回值
- 3) 方法名 和类名字必须一样
- 4) 参数列表 和 成员方法一样的规则
- 5) 构造器的调用，由系统完成

构造方法/构造器

● 基本介绍

构造方法又叫构造器(constructor)，是类的一种特殊的方法，它的主要作用是完成对新对象的初始化。它有几个特点：

- 1) 方法名和类名相同
- 2) 没有返回值
- 3) 在创建对象时，系统会自动的调用该类的构造器完成对对象的初始化。

● 注意事项和使用细节

ConstructorDetail.java

1. 一个类可以定义多个不同的构造器，即构造器重载
比如：我们可以再给Person类定义一个构造器，用来创建对象的时候，只指定人名，不需要指定年龄
2. 构造器名和类名要相同
3. 构造器没有返回值
4. 构造器是完成对象的初始化，并不是创建对象
5. 在创建对象时，系统自动的调用该类的构造方法

构造方法

6. 如果程序员没有定义构造器，系统会自动给类生成一个默认无参构造器(也叫默认构造方法)，比如 Person (){}，使用javap指令 反编译看看
7. 一旦定义了自己的构造器，默认的构造器就覆盖了，就不能再使用默认的无参构造器，除非显式的定义一下，即：Person(){}

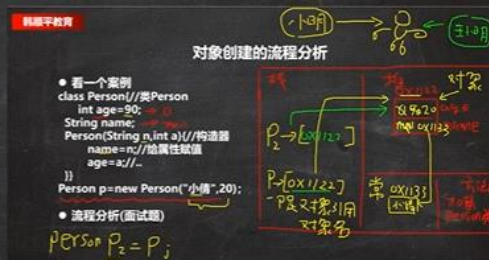
对象创建的流程分析

● 看一个案例

```
class Person{//类Person
    int age=90;
    String name;
    Person(String n,int a){//构造器
        name=n;//给属性赋值
        age=a;//..
    }
}
Person p=new Person("小倩",20);
```

● 流程分析(面试题)

1. 加载Person类信息(Person.class)，只会加载一次
2. 在堆中分配空间(地址)
3. 完成对象初始化 [3.1 默认初始化 age=0 name=null 3.2 显式初始化 age=90,name=null, 3.3 构造器的初始化 age =20, name=小倩]
4. 在对象在堆中的地址,返回给 p(p 是对象名,也可以理解成是对象的引用)

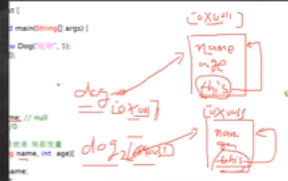


this关键字

- 深入理解this

为了进一步理解this,我们再看一个案例 (This01.java)

```
class Dog{
    public String name;
    public int age;
    public Dog(String name, int in_age){
        this.name = name;
        this.age = in_age;
    }
    public void info(){
        System.out.println(this.name + "\t" + this.age
        + "\t" + "当前对象的hashCode是: " + this.hashCode());
    }
    //使用hashCode() 看看对象的情况
}
```



this小结: 简单的说, 哪个对象调用, this就代表哪个对象

this关键字

- this的注意事项和使用细节

ThisDetail.java

1. this关键字可以用来访问本类的属性、方法、构造器
2. this用于区分当前类的属性和局部变量
3. 访问成员方法的语法: this.方法名(参数列表);
4. 访问构造器语法: this(参数列表); **注意只能在构造器中使用(即只能在构造器中访问另外一个构造器, 必须放在第一条语句)**
5. this不能在类定义的外部使用, 只能在类定义的方法中使用。

this关键字

- this的课堂案例

TestPerson.java

定义Person类, 里面有name、age属性, 并提供compareTo比较方法, 用于判断是否和另一个人相等, 提供测试类TestPerson用于测试, 名字和年龄完全一样, 就返回true, 否则返回false