

非贷款, 0元入学, 不1万就业不给1分钱学费, 我们已于四年了!

笔记总链接: <http://bbs.itheima.com/thread-200600-1-1.html>

1、Java语言基础组成-Part 5

1.9 数组

1.9.5 数组中的数组

二维数组[] []

格式1:

```
int[] [] arr = new int[3][2];
```

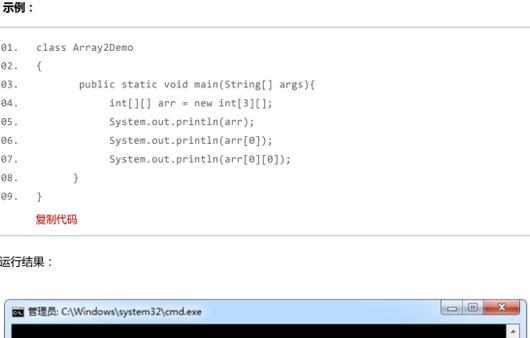
1. 定义了名称为arr的二维数组。
2. 二维数组中有3个一维数组。
3. 每一个一维数组中有2个元素。
4. 一维数组的名称分别为arr[0], arr[1], arr[2]。
5. 给第一个一维数组第一个索引位赋值为78写法是: arr[0][1] = 78。
6. arr存储的是二维数组的初始地址, arr[0], arr[1], arr[2]存储的是一维数组的初始地址。

示例1:

```
01. class Array2Demo
02. {
03.     public static void main(String[] args){
04.         int[] [] arr = new int[3][2];
05.         //直接打印二维数组
06.         System.out.println(arr);
07.         //直接打印二维数组中的角标为0的一维数组
08.         System.out.println(arr[0]);
09.         //直接打印二维数组中的角标为0的一维数组的角标为0的元素
10.         System.out.println(arr[0][0]);
11.     }
12. }
```

复制代码

运行结果:



说明: 打印的内容中@左边是实体的类型 ("["代表一维数组, "["代表二维数组, "I"代表int类型), @右边是实体的哈希值。

示例2: 打印二维数组和一维数组的长度

```
01. class Array2Demo
02. {
03.     public static void main(String[] args){
04.         int [] [] arr = new int[3][2];
05.         //打印二维数组的长度, 其实就是一维数组的个数
06.         System.out.println(arr.length);
07.         //打印二维数组中角标为1一位数组的长度
08.         System.out.println(arr[1].length);
09.     }
10. }
```

复制代码

运行结果:



格式2:

```
int[] [] arr = new int[3][];
```

二维数组中有3个一维数组, 每个一维数组都是默认初始化为null, 可以对这个三个一维数组分别进行初始化。

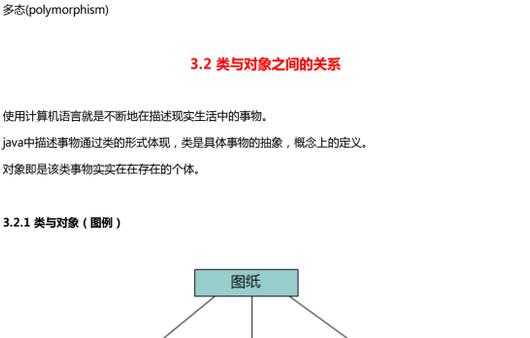
```
arr[0] = new int[3];
arr[1] = new int[1];
arr[2] = new int[2];
```

示例:

```
01. class Array2Demo
02. {
03.     public static void main(String[] args){
04.         int[] [] arr = new int[3][];
05.         System.out.println(arr);
06.         System.out.println(arr[0]);
07.         System.out.println(arr[0][0]);
08.     }
09. }
```

复制代码

运行结果:



说明:

示例中arr[0]为null的原因是没有为它初始化一位数组。

由于arr[0]为null, 通过它获取一维数组角标为0的元素肯定会报空指针异常错误!

格式3:

```
int[] [] arr = {{3,8,2},{2,7},{9,0,1,6}};
```

定义一个名称为arr的二维数组, 二维数组中的有三个一维数组, 每一个一维数组中具体元素也都已初始化。

第一个一维数组 arr[0] = {3,8,2}; 第二个一维数组 arr[1] = {2,7}; 第三个一维数组 arr[2] = {9,0,1,6}。

第三个一维数组的长度表示方式: arr[2].length;。

应用: 求二维数组所有元素的和。

```
01. class Array2Demo
02. {
03.     public static void main(String[] args){
04.         int sum = 0;
05.         int[] [] arr = {{3,1,7},{5,8,2,9},{4,1}};
06.
07.         for(int x = 0; x < arr.length; x++){
08.             for(int y = 0; y < arr[x].length; y++){
09.                 sum += arr[x][y];
10.             }
11.         }
12.         System.out.println("sum = " + sum);
13.     }
14. }
```

复制代码

运行结果:



3、面向对象

3.1 面向对象概念

3.1.1 面向对象概念

面向对象是相对面向过程而言, 面向对象和面向过程都是一种思想。

面向过程:

强调的是功能行为。代表语言: C语言。

例子: 把大象装进冰箱

1. 打开冰箱。
2. 存储大象。
3. 关上冰箱。

"打开"、"存储"、"关上"都是功能行为, 在代码中的直观体现就是函数或者方法, 这就是一种面向过程的功能行为为主体的思想体现。

面向对象:

将功能封装进对象, 强调具备了功能的对象。代表语言: Java、C++、C#。

例子: 把大象装进冰箱

1. 冰箱打开。
2. 冰箱存储。
3. 冰箱关闭。

可以看到, 所有的操作都是以"冰箱"为主体, 而不是功能行为。也就是说冰箱自己已经具备"打开"、"存储"、"关上"的行为功能, 我们只需要让冰箱执行它具有的功能就可以了。这就是一种面向对象的以执行功能的对象为主体的思想体现。

3.1.2 面向对象的特点

是一种符合人们思考习惯的思想, 可以将复杂的事情简单化, 将程序员从执行者转换成了指挥者。

完成需求时:

1. 先要去寻找具有所需功能的对象来用。
2. 如果该对象不存在, 那么创建一个具有所需功能的对象。

这样可以简化开发并提高复用。

3.1.3 面向对象开发, 设计, 特征

开发的过程:

其实就是不断的创建对象, 使用对象, 指挥对象做事情。

设计的过程:

其实就是在管理和维护对象之间的关系。

面向对象特征:

- 封装(encapsulation)
- 继承(inheritance)
- 多态(polymorphism)

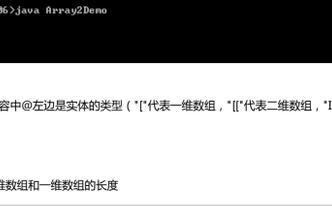
3.2 类与对象之间的关系

使用计算机语言就是不断地在描述现实生活中的事物。

java中描述事物通过类的形式体现, 类是具体事物的抽象, 概念上的定义。

对象即是该类事物实实在在存在的个体。

3.2.1 类与对象 (图例)



可以理解为: 类就是图纸, 汽车就是堆内存中的对象。

对于同一类事物可以抽取它们的共性的内容, 定义在类中。如: 生活中的汽车, 每一台车都有轮胎数和颜色。

那么在通过java描述汽车这类事物时, 就可以将这两个共性属性作为类中的属性进行定义。

通过该类建立的每一个汽车实体都具有该属性, 并可以有对象特有的属性值。

3.2.2 类的定义

生活中描述事物无非就是描述事物的属性和行为。如: 人有身高, 体重等属性, 有说话, 打球等行为。

Java中用类class来描述事物也是如此。

属性: 对应类中的成员变量。

行为: 对应类中的成员函数。

定义类其实在定义类中的成员(成员变量和成员函数)。

3.2.3 成员变量和局部变量的区别?

成员变量:

1. 成员变量定义在类中, 在整个类中都可以被访问。
2. 成员变量随着对象的建立而建立, 随着对象的消失而消失, 存在于对象所在的堆内存中。
3. 成员变量有默认初始化值。

局部变量:

1. 局部变量只定义在类方法范围内, 如: 函数内, 语句内等, 只在所属的区域有效。
2. 局部变量存在于栈内存中, 作用的范围结束, 变量空间会自动释放。
3. 局部变量没有默认初始化值。

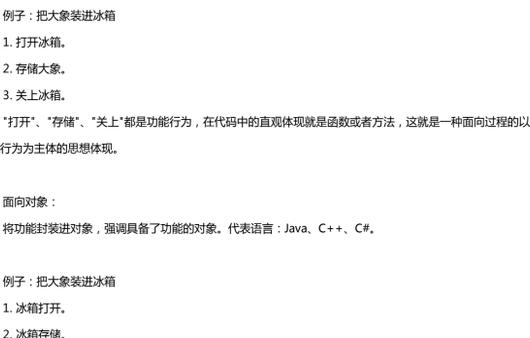
3.2.4 创建对象, 使用对象

示例:

```
01.
02. /*
03.  * 描述小汽车
04.  * 分析:
05.  * 1. 属性:
06.  *   轮胎数。
07.  *   颜色。
08.  * 2. 行为
09.  *   运行。
10.  *
11.  * 定义类其实就是在定义类中的成员。
12.  * 成员: 成员变量<-->属性, 成员函数<-->行为。
13.  */
14. //对Car这类事物进行描述
15. class Car
16. {
17.     //成员变量
18.     String color = "red";
19.     int num = 4;
20.
21.     //成员方法
22.     void show()
23.     {
24.         //临时变量
25.         int num = 10;
26.         //输出临时变量的值
27.         System.out.println("color = " + color + "...num = " + num);
28.     }
29. }
30.
31. class CarDemo
32. {
33.     public static void main(String[] args)
34.     {
35.         //通过new关键字, 建立对象
36.         //c就是一个类型的引用变量, 指向了该类的对象
37.         Car c = new Car();
38.         //对对象的属性进行修改
39.         c.color = "black";
40.         //输出成员变量的默认值
41.         System.out.println(c.num);
42.         //通过对象成员的形式来使用对象的功能
43.         c.show();
44.     }
45. }
```

复制代码

运行结果:

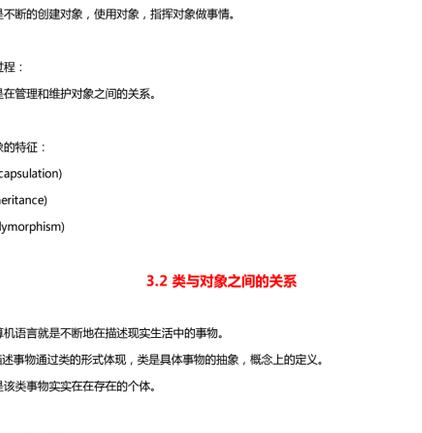


3.2.5 对象内存结构

示例:

```
Car c1 = new Car();
c1.color="blue";
Car c2 = new Car();
```

内存结构示意图:



只要是new操作符定义的实体就会在堆内存中开辟一个新的空间, 并且每一个对象中都有一份属于自己的属性。

通过对对象成员的方式操作对象中的成员, 对其中一个对象的成员进行了修改, 和另一个对象没有任何关系。

需要提到的是c1、c2都是对实体的引用变量, 如果执行c2 = c1, 那么c2也就指向了c1引用的实体。c2原来引用的实体因为没有被引用变量引用, 就会被垃圾回收器回收。

3.2.6 匿名对象

匿名对象是对象的简化形式。

匿名对象两种使用情况:

1. 当对对象方法仅进行一次调用时;
2. 匿名对象可以作为实际参数进行传递。

示例:

```
01.
02. class Car
03. {
04.     String color = "red";
05.     int num = 4;
06.
07.     public static void run()
08.     {
09.         System.out.println("function run is running!");
10.     }
11. }
12.
13. class CarDemo{
14.     public static void main(String[] args){
15.         //对对象方法仅进行一次调用时, 就可以使用匿名对象
16.         new Car().run();
17.         //匿名对象可以作为实际参数进行传递
18.         show(new Car());
19.     }
20.
21.     public static void show(Car c){
22.         c.num = 3;
23.         c.color = "black";
24.         System.out.println("function show is running!");
25.         System.out.println(c.num + "... " + c.color);
26.     }
27. }
```

复制代码

运行结果:



3.2.7 基本数据类型参数及引用数据类型参数传递

示例1: 基本数据类型参数传递

```
01. //基本数据类型参数传递
02. class Demo
03. {
04.     public static void main(String[] args){
05.         int x = 3;
06.         show(x);
07.         System.out.println("x = " + x);
08.     }
09.
10.     public static void show(int x){
11.         x = 4;
12.     }
13. }
```

复制代码

运行结果:



```
管理员: C:\Windows\system32\cmd.exe
D:\code\day06>javac Demo.java
D:\code\day06>java Demo
x=3
```

执行过程说明：

1. jvm调用main方法，main方法入栈。
2. 将x变量设置为3。
3. main方法调用show方法，3作为基本数据类型参数赋值给show方法参数x，也就是说，此时show方法的参数x值为3。
4. show方法执行x=4后，show方法的参数x值变为4。
5. show方法执行结束，show方法出栈。show方法参数x也随之出栈。
6. main方法打印x的值。此时x指的是main方法中的x变量的值（show方法中的参数x已经随show方法一块出栈了）。所以，打印出来的x值为4而不是3。
7. main方法执行结束，出栈。

示例2：引用数据类型参数传递

```
01. //引用数据类型参数传递
02. class Demo
03. {
04.     int x = 3;
05.
06.     public static void main(String[] args){
07.         Demo d = new Demo();
08.         d.x = 9;
09.         show(d);
10.         System.out.println(d.x);
11.     }
12.
13.     public static void show(Demo d){
14.         d.x = 4;
15.     }
16. }
```

[复制代码](#)

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day06>javac Demo.java
D:\code\day06>java Demo
4
```

执行过程说明：

1. jvm调用main方法，main方法入栈。
2. 创建Demo对象d（在堆内存中创建，d作为引用变量，指向堆内存中创建的实体对象），并将d指向的实体对象中的属性x的值设置为9。
3. main方法调用show方法，d作为引用数据类型参数赋值给show方法参数d，也就是说，此时show方法的参数d和main方法中的变量d同时指向了堆内存中同一个实体对象。
4. show方法执行d.x=4后，堆内存中的实体对象的x属性值变为4。
5. show方法执行结束，show方法出栈，show方法参数d也随之出栈。虽然show方法参数d出栈了，但是由于main方法的变量d依然引用着堆内存中的实体对象，因此堆内存中的实体对象不会被垃圾回收器清除。
6. main方法打印d.x的值。此时，d指的是main方法中的引用变量x，d.x指的依然是堆内存中的实体对象x的值。所以，打印出来的值为4而不是9。
7. main方法执行结束，出栈。

总结：

在java中，方法参数的传递永远都是传值，而这个值，对于基本数据类型，值就是你赋给变量的那个值。而对于引用数据类型，这个值是对象的引用，而不是这个对象本身。

3.3 封装

封装：是指隐藏对象的属性和实现细节，仅对外提供公共访问方式。

好处：

1. 将变化隔离。
2. 便于使用。
3. 提高重用性。
4. 提高安全性。

封装原则：

1. 将不需要对外提供的内容都隐藏起来。
2. 把属性都隐藏，提供公共方法对其访问。如，getXXX，setXXX。

示例：

```
01. /*
02. 人：
03. 属性：年龄
04. 行为：说话
05. */
06. class Person{
07.     //private: 私有，是一个权限修饰符，用于修饰
08.     //不希望别人直接访问赋值，需要通过私有化把属性进行隐藏
09.     private int age ;
10.
11.     //通过提供set、get公共方法对其访问
12.     public void setAge( int a){
13.         //在set方法内可以对属性的赋值进行限制
14.         if (a > 0 && a < 130){
15.             age = a;
16.         } else
17.             System.out.println("错误的数据");
18.     }
19.
20.     public int getAge(){
21.         return age ;
22.     }
23.
24.     void speak(){
25.         System.out.println("age = " + age);
26.     }
27. }
28.
29. class PersonDemo{
30.     public static void main(String[] args){
31.         Person p = new Person();
32.         //通过其他方式访问
33.         p.setAge(20);
34.         p.speak();
35.         //赋值不合法，set方法就不允许成功赋值
36.         p.setAge(-20);
37.     }
38. }
```

[复制代码](#)

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day06>javac PersonDemo.java
D:\code\day06>java PersonDemo
age = 20
错误的数据
```

P.S.

1. 私有仅仅是封装的一种体现而已。
2. private关键字：是一个权限修饰符，用于修饰成员(成员变量和成员函数)，被私有化的成员只在本类中有效。
3. 常用场景之一：将成员变量私有化，对外提供对应的set、get方法对其访问，提高对数据访问的安全性。

~END~



~爱上海，爱黑马~

