第06天 面向对象

今日内容介绍

* 面向对象概述
* 面向对象内存图
* private、this关键字和封装
* 面向对象之构造方法

# 面向对象概述

## 面向对象思想

* 我们回想一下，前面我们完成一个需求的步骤：首先是搞清楚我们要做什么，然后在分析怎么做，最后我们再代码体现。一步一步去实现，而具体的每一步都需要我们去实现和操作。这些步骤相互调用和协作，完成我们的需求。
* 在上面的每一个具体步骤中我们都是参与者，并且需要面对具体的每一个步骤和过程，这就是面向过程最直接的体现。
* 那么什么是面向过程开发呢? 面向过程开发，其实就是面向着具体的每一个步骤和过程，把每一个步骤和过程完成，然后由这些功能方法相互调用，完成需求。
* 面向过程的代表语言：C语言
* 当需求单一，或者简单时，我们一步一步去操作没问题，并且效率也挺高。可随着需求的更改，功能的增多，发现需要面对每一个步骤很麻烦了，这时就开始思索，能不能把这些步骤和功能在进行封装，封装时根据不同的功能，进行不同的封装，功能类似的用一个类封装在一起，这样结构就清晰了很多。用的时候，找到对应的类就可以了。这就是面向对象的思想。

### 面向过程思想与面向对象思想

A:什么是面向过程

面向过程，其实就是面向着具体的每一个步骤和过程，把每一个步骤和过程完成，然后由这些功能方法相互调用，完成需求。

例如:吃煎饼果子利用面向过程的思想:

1.学习摊煎饼的技术

2.买材料鸡蛋,油,葱等等

3.开始摊

4.吃

5.收拾

B:什么是面向对象

面向对象思想就是不断的创建对象，使用对象，指挥对象做事情。（如果有对象，直接用对象，对我们直接提供服务）

例如:吃煎饼果子利用面向对象的思想

1. 找会摊煎饼的大妈(创建一个摊煎饼大妈的对象)
2. 调用其摊煎饼的技能(功能),传递进去钱参数
3. 返回给我们一个煎饼
4. 吃

### 面向过程特点与面向对象特点

A:面向过程：

强调的是过程，所有事情都需要自己完成

B:面向对象:

是一种更符合我们思想习惯的思想(懒人思想，我把事情自己不做，交给别人去做)

可以将复杂的事情简单化（对使用者来说简单了，对象里面还是很复杂的）

将我们从执行者变成了指挥者角色发生了转换

#### 案例代码一

**package** com.itheima\_01;

/\*

\* 面向对象思想：

\* 面向对象是基于面向过程的一种编程思想。

\*

\* 面向过程：强调的是每一个功能的步骤

\* 面向对象：强调的是对象，然后由对象去调用功能

\*

\* 面向对象的思想特点：

\* A:是一种符合我们思考习惯的思想

\* B:可以将复杂的事情简单化

\* C:让我们从执行者变成了指挥者

\*

\* 举例：洗衣服

\* 面向过程：把衣服脱下来--找一个盆--放水--放洗衣粉--浸泡10分钟--揉一揉--清洗衣服--拧干--晾起来

\* 面向对象：把衣服脱下来--打开全自动洗衣机--放入衣服--按钮--晾起来

\*/

**public** **class** Demo {

}

## 类与对象概述

A:我们学习编程是为了什么

为了把我们日常生活中实物用学习语言描述出来

B:我们如何描述现实世界事物

属性 就是该事物的描述信息(事物身上的名词)

行为 就是该事物能够做什么(事物身上的动词)

C:Java中最基本的单位是类,Java中用class描述事物也是如此

成员变量 就是事物的属性

成员方法 就是事物的行为

D:定义类其实就是定义类的成员(成员变量和成员方法)

a:成员变量 和以前定义变量是一样的，只不过位置发生了改变。在类中，方法外。

b:成员方法 和以前定义方法是一样的，只不过把static去掉，后面在详细讲解static的作用。

E:类和对象的概念

a:类：是一组相关的属性和行为的集合（我们班所有的同学都具备相同的属性和行为，比如：姓名，年龄，学习，这样就把所有的学生成为学生类）

b:对象：是该类事物的具体体现（说某个同学时，他都具备自己特有的属性和行为）

c:举例：

类 学生

对象 具体的某个学生就是一个对象

比如：车是一个类，具体的开的奔驰、宝马，就是对象

### 案例代码二

**package** com.itheima\_01;

/\*

\* 我们学习编程语言，其实就是为了把现实世界的事物模拟出来，实现信息化。

\*

\* 我们是如何表示现实世界的事物的呢?

\* A:属性 就是事物的描述信息

\* B:行为 就是事物能够做什么

\* 举例：

\* 学生

\*

\* Java语言最基本的组成单位是类，所以，在后面的学习过程中，我们是通过类来体现事物的。

\*

\* 类：是一组相关的属性和行为的集合

\* 对象：就是该类事物的具体体现

\* 举例：

\* 类 学生

\* 对象 张三

\*/

**public** **class** Demo2 {

}

## 类与对象案例:

### 学生类的定义

#### 案例代码三

**package** com.itheima\_02;

/\*

\* 类的定义：类是用来描述现实世界事物的

\* 事物：

\* 属性

\* 行为

\* 类如何和事物进行对应的呢?

\* 类：

\* 成员变量

\* 成员方法

\*

\* 成员变量：和我们学习过的变量的定义是一样的，有两个小区别。

\* 位置不同：类中，方法外

\* 初始化值：成员变量不需要给初始化值也可以使用，因为它有默认值

\* 成员方法：和我们学习过的方法的定义是一样的，有一个小区别。

\* 去掉了修饰符中的static关键字

\*

\* 需求：定义一个学生类。

\*

\* 学生事物：

\* 属性：姓名，年龄...

\* 行为：学习，吃饭...

\*

\* 学生类：

\* 成员变量：姓名，年龄...

\* 成员方法：学习，吃饭...

\*/

**public** **class** Student {

//成员变量

//姓名

String name;

//年龄

**int** age;

//成员方法

//学习的方法

**public** **void** study() {

System.***out***.println("好好学习，天天向上");

}

//吃饭的方法

**public** **void** eat() {

System.***out***.println("学习饿了要吃饭");

}

}

### 学生类的使用

#### 案例代码四

**package** com.itheima\_02;

/\*

\* 类的使用：

\* 使用一个类，其实就是使用该类的成员。(成员变量和成员方法)

\* 而我们要想使用一个类的成员，就必须先拥有该类的对象。

\* 那么我们如何拥有一个类的对象呢?

\* 创建对象就可以了

\* 我们如何创建对象呢?

\* 格式：

\* 类名 对象名 = new 类名();

\* 对象如何访问成员呢?

\* 成员变量：

\* 对象名.成员变量

\* 成员方法：

\* 对象名.成员方法(...)

\*/

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

//类名 对象名 = new 类名();

Student s = **new** Student();

//System.out.println("s:"+s);//com.itheima\_02.Student@da6bf4

//使用成员变量

System.***out***.println("姓名："+s.name);//null

System.***out***.println("年龄："+s.age);//0

System.***out***.println("----------");

//给成员变量赋值

s.name = "林青霞";

s.age = 30;

//再次使用成员变量

System.***out***.println("姓名："+s.name);//林青霞

System.***out***.println("年龄："+s.age);//30

System.***out***.println("----------");

//调用成员方法

s.study();

s.eat();

}

}

### 手机类案例

#### 案例代码五

**package** com.itheima\_02;

/\*

\* 手机类：

\* 成员变量：品牌，价格，颜色...

\* 成员方法：打电话，发短信...

\*/

**public** **class** Phone {

//品牌

String brand;

//价格

**int** price;

//颜色

String color;

//打电话

**public** **void** call(String name) {

System.***out***.println("给"+name+"打电话");

}

//发短信

**public** **void** sendMessage(String name) {

System.***out***.println("给"+name+"发短信");

}

}

**手机案例使用：**

**package** com.itheima\_02;

/\*

\* 手机类的测试类

\*/

**public** **class** PhoneTest {

**public** **static** **void** main(String[] args) {

//创建对象

Phone p = **new** Phone();

//输出成员变量的值

System.***out***.println("品牌："+p.brand);//null

System.***out***.println("价格："+p.price);//0

System.***out***.println("颜色："+p.color);//null

System.***out***.println("------------");

//给成员变量赋值

p.brand = "锤子";

p.price = 2999;

p.color = "棕色";

//再次输出成员变量的值

System.***out***.println("品牌："+p.brand);//锤子

System.***out***.println("价格："+p.price);//2999

System.***out***.println("颜色："+p.color);//棕色

System.***out***.println("------------");

//调用成员方法

p.call("林青霞");

p.sendMessage("林青霞");

}

}

# 对象的内存图

## 一个对象的内存图

### 案例代码六

**package** com.itheima\_03;

/\*

\* 手机类：

\* 成员变量：品牌，价格，颜色...

\* 成员方法：打电话，发短信...

\*/

**public** **class** Phone {

//品牌

String brand;

//价格

**int** price;

//颜色

String color;

//打电话

**public** **void** call(String name) {

System.***out***.println("给"+name+"打电话");

}

//发短信

**public** **void** sendMessage(String name) {

System.***out***.println("给"+name+"发短信");

}

}

**package** com.itheima\_03;

/\*

\* 手机测试类

\*/

**public** **class** PhoneTest {

**public** **static** **void** main(String[] args) {

Phone p = **new** Phone();

System.***out***.println(p.brand+"---"+p.price+"---"+p.color);

p.brand = "锤子";

p.price = 2999;

p.color = "棕色";

System.***out***.println(p.brand+"---"+p.price+"---"+p.color);

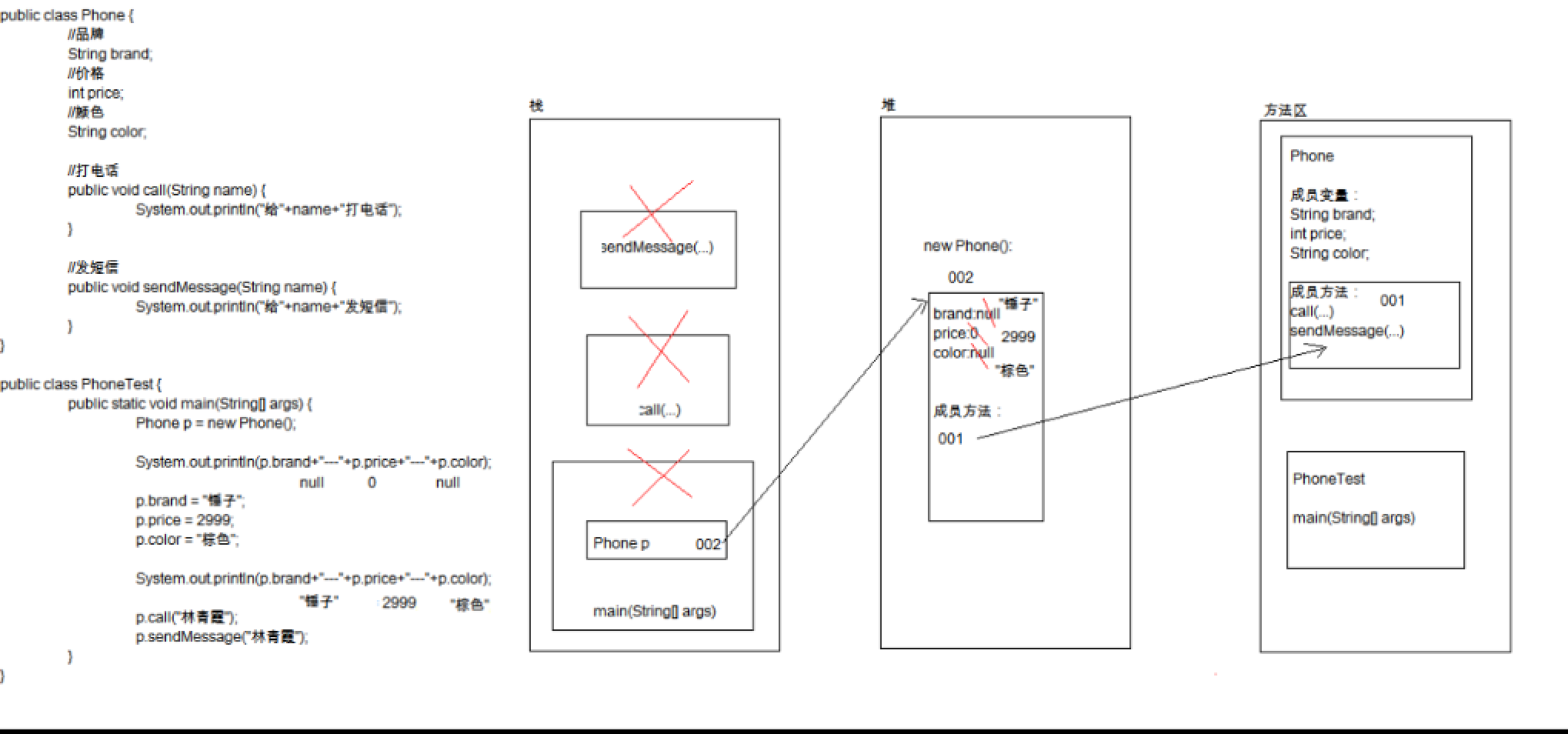
p.call("林青霞");

p.sendMessage("林青霞");

}

}

### 一个对象的内存图图解



## 两个对象的内存图(共用方法区)

### 案例代码七

package com.itheima\_03;

/\*

\* 手机类的测试类

\*/

public class PhoneDemo2 {

public static void main(String[] args) {

Phone p = new Phone();

p.brand = "小米5s";

p.price = 1999;

p.color = "银色";

System.out.println(p.brand+"---"+p.price+"---"+p.color);

p.call("林青霞");

p.sendMessage();

Phone p2 = new Phone();

p2.brand = "IPhone7S";

p2.price = 7999;

p2.color = "土豪金";

System.out.println(p2.brand+"---"+p2.price+"---"+p2.color);

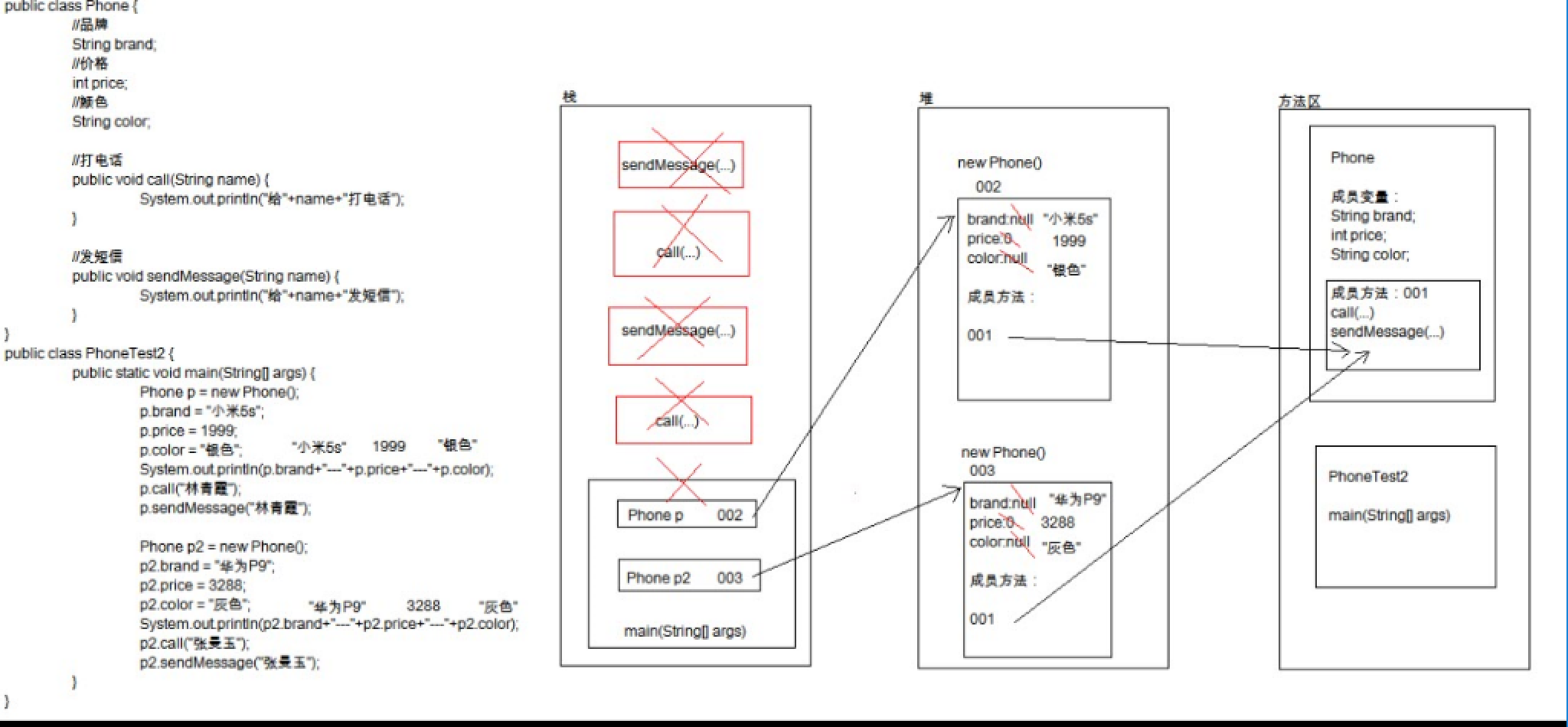
p2.call("张曼玉");

p2.sendMessage();

}

}

### 两个对象的内存图(共用方法区)图解



## 两个对象的内存图(两个引用指向同一个堆内存)

### 案例代码八

package com.itheima\_03;

/\*

\* 手机类的测试类

\*/

public class PhoneDemo3 {

public static void main(String[] args) {

Phone p = new Phone();

p.brand = "OPPO";

p.price = 2999;

p.color = "白色";

System.out.println(p.brand+"---"+p.price+"---"+p.color);

Phone p2 = p;

p2.brand = "魅族";

p2.price = 1999;

p2.color = "蓝色";

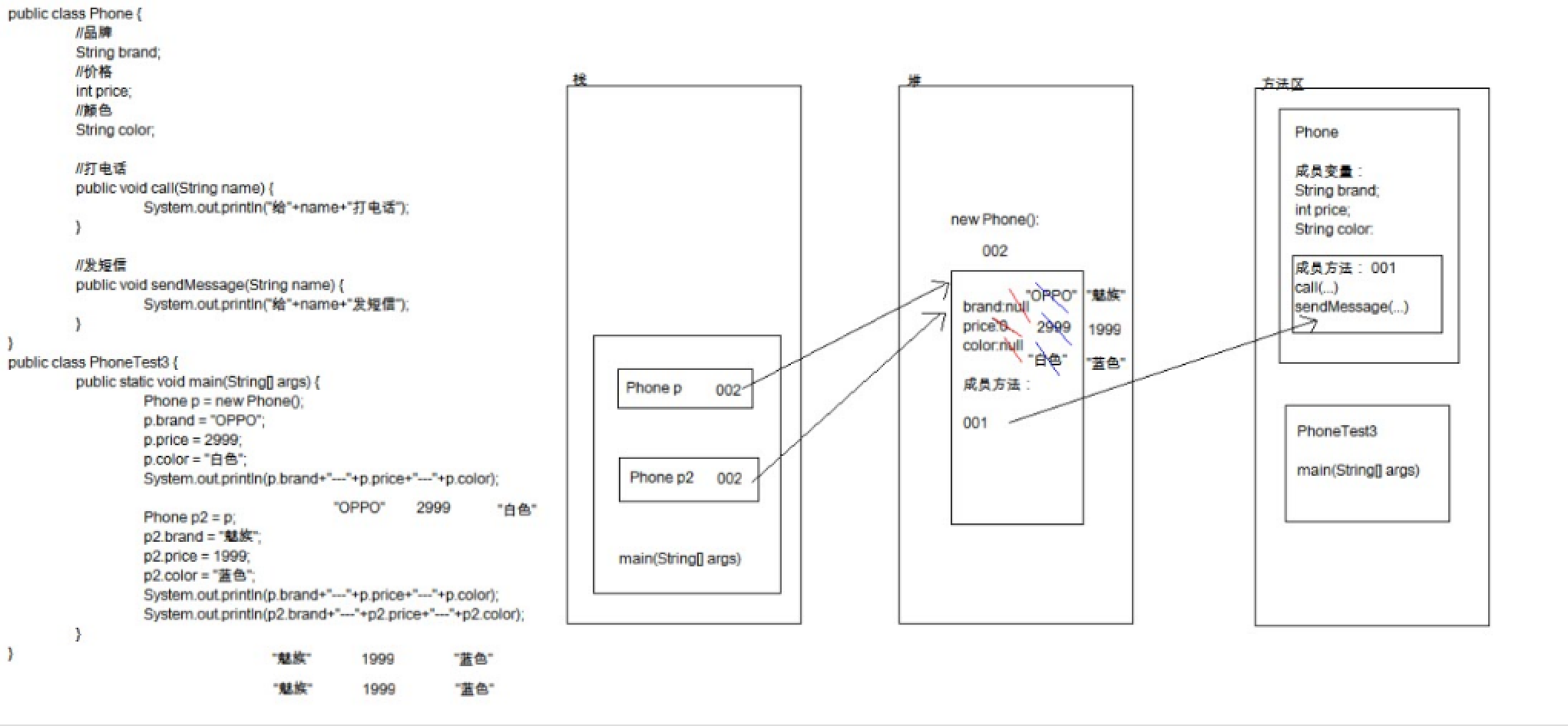
System.out.println(p.brand+"---"+p.price+"---"+p.color);

System.out.println(p2.brand+"---"+p2.price+"---"+p2.color);

}

}

### 两个对象的内存图(两个引用指向同一个堆内存)图解



## 成员变量和局部变量区别

### 案例代码九

**package** com.itheima;

/\*

\* 成员变量和局部变量的区别：

\* A:在类中的位置不同

\* 成员变量：在类中，方法外

\* 局部变量：在方法中或者方法声明上(形式参数)

\* B:在内存中的位置不同

\* 成员变量：在堆内存

\* 局部变量：在栈内存

\* C:生命周期不同

\* 成员变量：随着对象的创建而存在，随着对象的消失而消失

\* 局部变量：随着方法的调用而存在，随着方法的调用完毕而消失

\* D:初始化值不同

\* 成员变量：有默认值

\* 局部变量：没有默认值，必须先定义，赋值，最后使用

\*/

**public** **class** VariableDemo {

**int** x;

**public** **void** show() {

**int** y = 10;

System.***out***.println(x);

System.***out***.println(y);

}

}

# private、this关键字和封装

## private关键字的概述和特点

### private关键字

A:private关键字：

a:是一个权限修饰符。

b:可以修饰成员(成员变量和成员方法)

c:被private修饰的成员只在本类中才能访问。

#### 案例代码十

**package** com.itheima\_01;

/\*

\* 直接使用对象名访问成员变量，会存在数据的安全问题

\* 这个时候，我们就应该思考能不能让外界不要直接访问成员变量?

\* 能

\* 如何实现呢?

\* 用private关键字

\* private:

\* 是一个修饰符

\* 可以修饰成员变量，也可以修饰成员方法

\* 被private修饰的成员只能在本类中被访问

\*

\* 针对被private修饰的成员变量，我们会相应的给出getXxx()和setXxx()用于获取和设置成员变量的值，

\* 方法用public修饰

\*/

**public** **class** Student {

String name;

//int age;

**private** **int** age;

**public** **void** setAge(**int** a) {

**if**(a<0 || a>200) {

System.***out***.println("你给的年龄有误");

}**else** {

age = a;

}

}

**public** **int** getAge() {

**return** age;

}

**public** **void** show() {

System.***out***.println(name+"\*\*\*"+age);

}

}

**package** com.itheima\_01;

/\*

\* 测试类

\*/

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

//创建对象

Student s = **new** Student();

s.show();

//给成员变量赋值

s.name = "林青霞";

//s.age = 30;

//s.age = -30;

//通过方法赋值

s.setAge(-30);

s.show();

}

}

## private修饰后的标准类代码

* private最常见的应用：
  + 把成员变量用private修饰
  + 提供对应的getXxx()/setXxx()方法

#### 案例代码十一

**package** com.itheima\_02;

/\*

\* 学生类

\*/

**public** **class** Student {

**private** String name;

**private** **int** age;

**public** **void** setName(String n) {

name = n;

}

**public** String getName() {

**return** name;

}

**public** **void** setAge(**int** a) {

age = a;

}

**public** **int** getAge() {

**return** age;

}

}

**package** com.itheima\_02;

/\*

\* 测试类

\*/

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

//创建对象

Student s = **new** Student();

System.***out***.println(s.getName()+"\*\*\*"+s.getAge());

//给成员变量赋值

s.setName("林青霞");

s.setAge(30);

System.***out***.println(s.getName()+"\*\*\*"+s.getAge());

}

}

## 封装的概述和好处

A:封装概述

是面向对象三大特征之一

是面向对象编程语言对客观世界的模拟，客观世界里成员变量都是隐藏在对象内部的，外界无法直接操作和修改。就像刚才说的年龄。

B:封装原则：

将不需要对外提供的内容都隐藏起来。

把属性隐藏，提供公共方法对其访问。

成员变量private，提供对应的getXxx()/setXxx()方法

C:好处：

通过方法来控制成员变量的操作，提高了代码的安全性

把代码用方法进行封装，提高了代码的复用性

## this关键字的概述和使用

### this关键字由来和使用

A:this:代表所在类的对象引用

方法被哪个对象调用，this就代表那个对象

B:什么时候使用this呢

局部变量和成员变量重名

#### 案例代码十二

**package** com.itheima;

/\*

\* 学生类

\*

\* 目前代码中的n和a没有做到见名知意，需要改进。

\*

\* 如果局部变量名称和成员变量名称一致，在方法中使用的时候采用的是就近原则。

\*

\* 我们有没有办法把局部变量的name赋值给成员变量的name呢?

\* 有

\*

\* 有什么办法呢?

\* 用this关键字就可以解决这个问题

\*

\* this:代表所在类的对象引用

\* 方法被哪个对象调用，this就代表谁。

\*

\* 使用场景：

\* 就是用于解决成员变量被隐藏的问题。

\*/

**public** **class** Student {

**private** String name;

**private** **int** age;

**public** **void** setName(String name) { //"林青霞"

//name = name;

**this**.name = name;

}

**public** String getName() {

**return** name;

//return this.name;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **int** getAge() {

**return** age;

}

}

**package** com.itheima;

/\*

\* 测试类

\*/

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

// 创建对象

Student s = **new** Student();

System.***out***.println(s.getName() + "---" + s.getAge());

s.setName("林青霞");

s.setAge(30);

System.***out***.println(s.getName() + "---" + s.getAge());

}

}

# 面向对象之构造方法

## 构造方法的概述和使用

### 构造方法作用概述

* + 给对象的数据进行初始化

### 构造方法格式

* + 方法名与类名相同
  + 没有返回值类型，连void都没有
  + 没有具体的返回值

### 案例代码十二

package com.itheima\_08;

/\*

\* 构造方法：

\* 给对象的数据进行初始化

\*

\* 格式：

\* 方法名和类名相同

\* 没有返回值类型，连void都不能写

\* 没有具体的返回值

\*

\*/

public class Student {

public Student() {

System.out.println("这是构造方法");

}

}

package com.itheima\_08;

public class StudentDemo {

public static void main(String[] args) {

//如何调用构造方法呢?

//通过new关键字调用

//格式：类名 对象名 = new 构造方法(...);

Student s = new Student();

}

}

## 构造方法的注意事项

如果你不提供构造方法，系统会给出默认构造方法

如果你提供了构造方法，系统将不再提供

构造方法也是可以重载的,重载条件和普通方法相同

### 案例代码十三

**package** com.itheima\_01;

/\*

\* 构造方法：用于给对象的数据进行初始化

\*

\* 格式：

\* 方法名和类名相同

\* 没有返回值类型，连void都不能写

\* 没有具体的返回值

\*

\* 构造方法的注意事项：

\* A:如果我们没有给出构造方法，系统将给出一个默认的无参数构造方法供我们使用

\* B:如果我们给出了构造方法，系统将不再提供默认的构造方法供我们使用。

\* 这个时候，我们如果还想使用无参数构造方法，就必须自己提供。

\* 我们推荐：自己给出无参数构造方法。

\* C:构造方法也是可以重载的。

\*

\* 给成员变量赋值的方式：

\* A:通过setXxx()

\* B:通过构造方法

\*/

**public** **class** Student {

**private** String name;

**private** **int** age;

/\*

public Student() {

System.out.println("这是构造方法");

}

\*/

**public** Student() {}

**public** Student(String name) {

**this**.name = name;

}

**public** Student(**int** age) {

**this**.age = age;

}

**public** Student(String name,**int** age) {

**this**.name = name;

**this**.age = age;

}

**public** **void** show() {

System.***out***.println(name+"---"+age);

}

}

**package** com.itheima\_01;

/\*

\* 测试类

\*/

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

//如何调用构造方法呢?

//其实通过new关键字就可以

//格式：类名 对象名 = new 构造方法名(...);

Student s = **new** Student();

s.show();

//public Student(String name)

Student s2 = **new** Student("林青霞");

s2.show();

//public Student(int age)

Student s3 = **new** Student(30);

s3.show();

//public Student(String name,int age)

Student s4 = **new** Student("林青霞",30);

s4.show();

}

}

## 标准的类代码和测试

A:类:

a:成员变量

b:构造方法

无参构造方法

带参构造方法

c:成员方法

getXxx()

setXxx()

B:给成员变量赋值的方式

a:无参构造方法+setXxx()

b:带参构造方法

### 案例代码十四

**package** com.itheima\_02;

/\*

\* 标准的学生类

\*/

**public** **class** Student {

//成员变量

**private** String name;

**private** **int** age;

//构造方法

**public** Student() {}

**public** Student(String name,**int** age) {

**this**.name = name;

**this**.age = age;

}

//成员方法

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **int** getAge() {

**return** age;

}

}

**package** com.itheima\_02;

/\*

\* 测试类

\*/

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

//无参构造方法+setXxx()

Student s = **new** Student();

s.setName("林青霞");

s.setAge(30);

System.***out***.println(s.getName()+"---"+s.getAge());

//带参构造方法

Student s2 = **new** Student("林青霞",30);

System.***out***.println(s2.getName()+"---"+s2.getAge());

}

}