# 面向对象

## 概述

如今主流的软件开发思想有两种：一个是**面向过程**，另一个是**面向对象**。面向过程出现得较早，典型代表为C语言，开发中小型项目的效率很高，但是很难适用于如今主流的大中型项目开发场景。面向对象则出现得更晚一些，典型代表为Java或C++等语言，更加适合用于大型开发场景。两种开发思想各有长短。

对于**面向过程**的思想：当需要实现一个功能的时候，看中的是每一个步骤怎么做，整体的过程是怎样的。每一个操作都需要自己亲力亲为。

对于**面向对象**的思想：当需要实现一个功能的时候，不看重具体的过程和步骤是怎样的，而是关心“谁能帮我做这件事情”。（偷懒儿，找人帮我做事儿。）

面向对象的**三大特征**有：封装性、继承性、多态性。

## 生活举例

1. 洗衣服
   * 面向过程（手洗）：脱衣服、找一个盆、加水、加洗衣粉、浸泡30分钟、搓洗、拧衣服、倒掉水、再加水、漂洗、拧衣服、倒掉水、晾衣服。
   * 面向对象（机洗）：脱衣服、放入洗衣机、按下开关、拿出衣服晾晒。
2. 买电脑
   * 面向过程（自己买）：需要电脑、查询参数信息、横向比较机型、了解打折信息、与店家讨价还价、下单、收快递、开机验货、确认收货。
   * 面向对象（找人买）：需要电脑、找秘书帮我买、收电脑。
3. 吃饭
   * 面向过程（自己做）：我饿了、买菜、洗菜择菜、切菜、起火、炒菜、盛菜、吃饭、刷碗。
   * 面向对象（出去买）：我饿了、买饭、吃饭。

## 代码体验

**package** cn.itcast.day06.demo01;

**import** java.util.Arrays;

/\*

\* 面向过程：

\* 当需要实现一个功能的时候，所有的步骤都要亲力亲为，处理每一个代码细节。

\*

\* 面向对象：

\*/

**public** **class** Demo01PrintArray {

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

**int**[] array = { 10, 20, 30, 40, 50 };

// 首先使用面向过程的思想，每一个步骤细节都要亲自处理

System.***out***.print("[");

**for** (**int** i = 0; i < array.length; i++) {

**if** (i == array.length - 1) {

System.***out***.println(array[i] + "]");

} **else** {

System.***out***.print(array[i] + ", ");

}

}

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

// 使用面向对象的思想，达到同样的效果。

// 自己不处理详细的步骤，而是找人帮我做事儿。

// 在JDK当中已经给我们提供好了一个工具，叫做Arrays

// 通过这个工具就可以将数组转换成为指定格式的字符串

// 我自己不做，而是找Arrays帮我把数组转换成为指定格式的字符串

System.***out***.println(Arrays.*toString*(array));

}

}

## 类与对象的关系

我们学习编程语言，就是为了模拟现实世界中的事物，实现信息化来提高工作效率。例如银行的业务系统、超市的结账系统等，都是如此。那么现实世界的事物通常分成两个组成部分：

1. **属性**：事物的特征描述信息，“是什么”。
2. **行为**：事物的能力行动方案，“能做什么”。

面向对象的语言当中，“类”就是用来模拟现实事物的。类中也有属性、行为两个组成部分，而“对象”是类的具体实例。例如：

1. **类**：抽象的，是一张“手机设计图”。
2. **对象**：具体的，是一个“真正的手机实例”。

## 类的定义

类是用来模拟现实事物的代码手段，事物分成属性、行为两个部分，类当中也对应地包含两个部分：

1. **成员变量**（属性）：将变量位置直接定义在类中，在方法外，即为成员变量。
2. **成员方法**（行为）：将普通的方法去掉static关键字，即为成员方法。

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

// 成员变量

String name; // 姓名

**int** age; // 年龄

// 成员方法

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

System.***out***.println("吃饭饭！");

}

**public** **void** sleep() {

System.***out***.println("睡觉觉！");

}

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

System.***out***.println("学习！");

}

}

## 创建对象

类如同一张设计图纸，通常情况下不能直接使用；要想使用，则需要根据类创建一个真正的对象，然后使用对象中的功能。根据类创建对象的格式为：

**类名称 对象名 = new 类名称();**

**public** **class** Demo02Student {

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

// 希望根据Student类来创建一个学生对象

// 格式：类名称 对象名 = new 类名称();

// 创建一个对象，是一个学生，对象名称叫stu1

// 对象的名称stu1也叫做对象的“引用名”。

Student stu1 = **new** Student();

Student stu2 = **new** Student(); // 创建了第二个对象

Student stu3 = **new** Student(); // 创建更多学生对象

Student stu4 = **new** Student(); // 创建更多学生对象

Student stu5 = **new** Student(); // 创建更多学生对象

}

}

## 对象的基本使用

对象是根据类创建出来的，所以内部包含成员变量、成员方法两个组成部分。使用它们的格式如下：

1. 使用成员变量：**对象名.成员变量名**
2. 调用成员方法：**对象名.成员方法名(参数)**

简单来说，“想用谁就点儿谁”。

**public** **class** Demo03Student {

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

Student stu = **new** Student(); // 创建了一个学生对象

System.***out***.println(stu.name); // null

System.***out***.println(stu.age); // 0

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

// 改变成员变量的数据值

stu.name = "鹿晗";

stu.age = 20;

System.***out***.println(stu.name); // 鹿晗

System.***out***.println(stu.age); // 20

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

// 将对象当中的成员变量，交给name变量

String name = stu.name;

System.***out***.println(name); // 鹿晗

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

// 使用对象当中的成员方法

stu.eat(); // 调用吃饭的成员方法

stu.sleep(); // 调用睡觉的成员方法

stu.study(); // 调用学习的成员方法

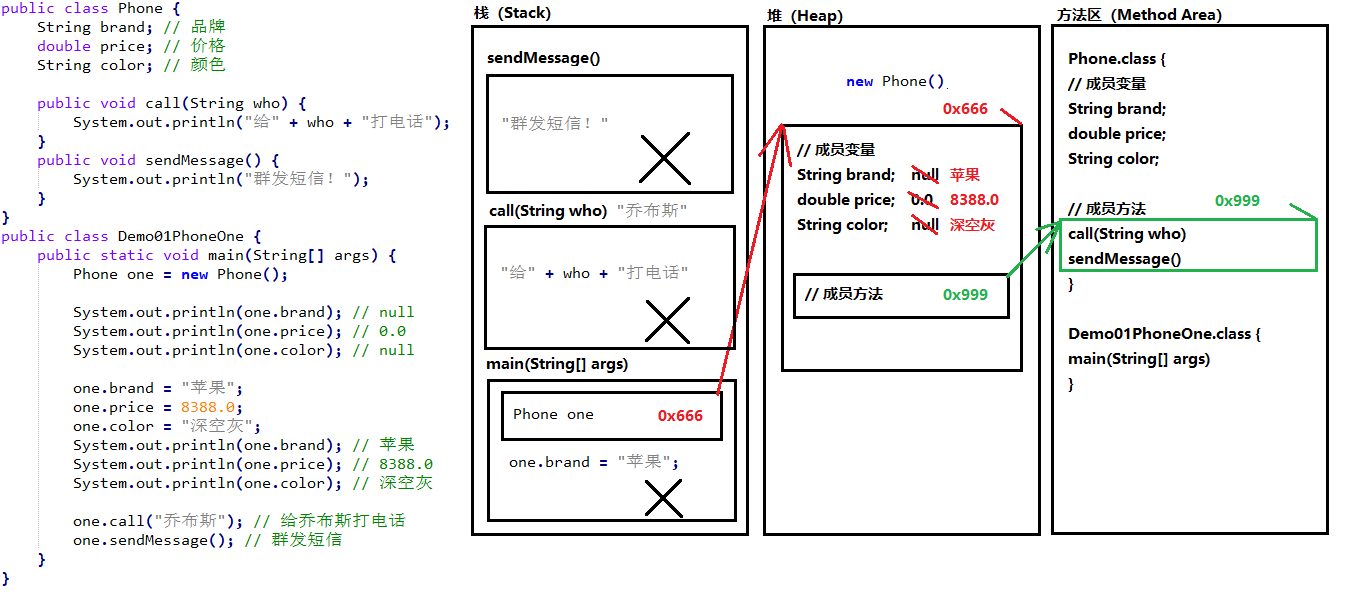
}

}

## 对象的内存图

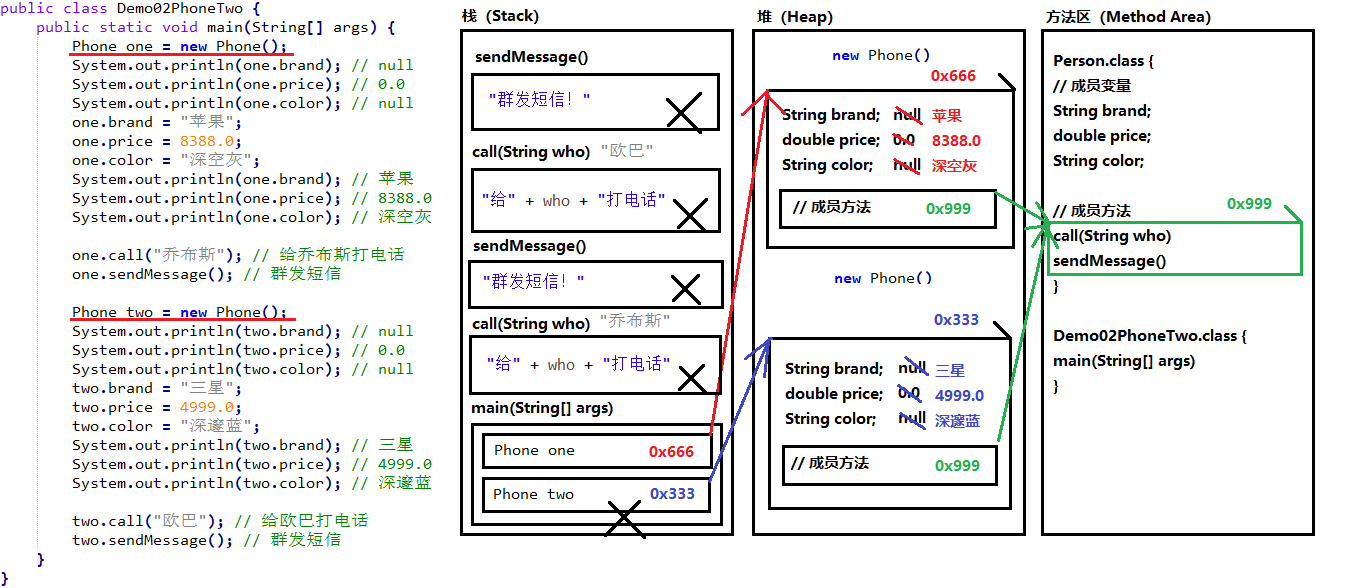
### 一个对象

对象的名称其实就是一个局部变量，里面存放的是地址值。真正new出来的对象在堆内存当中，而且成员方法保存的是方法区中的地址。



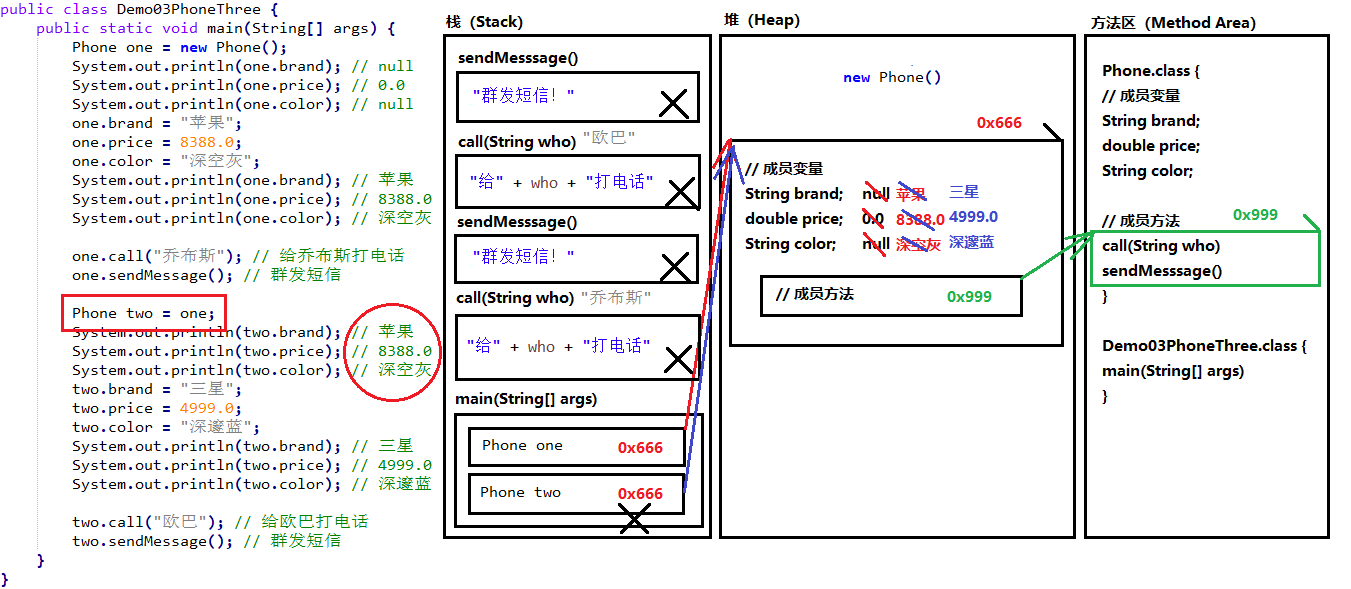
### 两个对象

两个对象的情况与一个对象是类似的，对象名称作为局部变量保存地址值，对象中的成员方法是方法区中的地址值。



### 同一个对象

如果使用“Phone two = one;”的代码写法，那么是将one当中的地址值赋值给two，于是one和two的地址相同，指向了同一个对象。



## 区分局部变量与成员变量

1. 定义的位置不同：
   1. 局部变量定义在方法内
   2. 成员变量定义在方法外，直接在类中
2. 内存的位置不同：
   1. 局部变量是在栈内存中的
   2. 成员变量是在堆内存中的
3. 生命周期不同：
   1. 局部变量随着方法进栈而出现，随着方法出栈而消失
   2. 成员变量随着对象被创建而出现，随着对象被回收而消失
4. 默认值不同：
   1. 局部变量没有默认值，必须赋值之后才能使用
   2. 成员变量如果没有赋值，那么会有默认值

## private关键字

如果类当中有一些成员不希望被外界直接访问，那么可以使用private关键字进行修饰，从而达到确保数据安全的目的。

private关键字是面向对象三大特征（封装、继承、多态）之一**封装性**的重要体现。

public class Person {

// 成员变量

String name; // 姓名

private int age; // 年龄

// 成员方法

public void show() {

System.*out*.println("我叫" + name + "，今年" + age + "岁。");

}

// 专门定义了一个成员方法用来设置成员变量的数据，Setter方法

public void setAge(int num) {

// 设置条件判断进行数据合理性检测，把关！

if (num < 0 || num > 200) {

System.*out*.println("数据错误！");

} else {

age = num;

}

}

// 专门定义了一个成员方法用来获取成员变量的数据，Getter方法

public int getAge() {

return age;

}

}

## this关键字

如果成员变量和局部变量重名，那么方法中会根据就近原则优先使用局部变量。此时可以通过this关键字进行二者的区分：

**this.成员变量名**

public class Person {

String name; // 我自己的名字

public void sayHello(String name) { // 对方的名字

System.*out*.println(name + "，你好。我是" + this.name + "。");

}

}

## 构造方法

构造方法是指在类当中专门用于创建对象的方法，定义格式：

**修饰符 构造方法名(参数类型 参数名称) {**

**方法体**

**// return; // 最后一行return通常省略**

**}**

当通过new关键字创建对象的时候，实际上就是在调用构造方法。

**注意事项**：

1. 构造方法根本不写返回值类型，连void都不写。
2. 构造方法的名称和所在的类名称必须完全一样，连大小写都要一样
3. 构造方法也是可以进行重载的。
4. 构造方法如果没有自定义，那么将会默认赠送一个。如果自定义了至少一个，那么就不再赠送默认的。

## 练习：两个对象的邂逅

定义一个类代表“人”，含有姓名和年龄，而且具有打招呼的功能。根据类创建两个对象并且赋值。让两个对象分别和对方打招呼。

# 知识总结

1. 面向过程：当需要实现一个功能的时候，每一个过程中的详细步骤和细节都要亲力亲为。
2. 面向对象：当需要实现一个功能的时候，不关心详细的步骤细节，而是找人帮我做事儿。
3. 类和对象的关系：
   1. 类是抽象的，通常不能直接使用。好像是一张手机设计图纸。
   2. 对象是具体的，根据类创建一个对象使用，也就是根据图纸创造一个手机。
4. 如何定义一个类：成员变量、成员方法。
   1. 成员变量：直接定义在类当中，在方法外面。
   2. 成员方法：去掉static关键字，其他和此前的普通方法一样。
5. 如何根据类创建对象，格式：类名称 对象名 = new 类名称();
6. 如何使用对象：
   1. 使用成员变量：对象名.成员变量名
   2. 调用成员方法：对象名.成员方法名(参数)
7. 局部变量和成员变量的不同：
   1. 定义的位置不同
   2. 内存的位置不同
   3. 生命周期不同
   4. 默认值不同
8. private关键字用法：直接写在成员变量前面，类外面不能直接访问，确保安全性。间接访问：编写一对儿Getter Setter方法。（特例：如果是boolean类型，getXxx必须叫做isXxx的形式。）
9. this关键字典型用法：用来区分同名的成员变量和局部变量。在成员变量的前面写上“this.”前缀即可。
10. 构造方法：专门用来创建对象的方法，通过new关键字创建对象就是在调用构造。
    1. 构造方法不能写返回值类型，连void都不能写。
    2. 构造方法的名称必须和所在的类名称完全一样，大小写也要一样。
    3. 构造方法也是可以重载的。
    4. 构造方法如果没有定义，默认送一个；如果定义了，那么默认的不再赠送。
11. 如何定义一个标准的POJO类：
    1. 所有的成员变量都用private修饰
    2. 为每一个成员变量编写一对儿Getter Setter方法
    3. 编写一个无参数的构造方法
    4. 编写一个全参数的构造方法