Java面向对象 第7天

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【理解】面向过程和面向对象的思想**
2. 【理解】过程的概念
3. 【理解】面向过程解决问题的步骤
4. 【理解】对象的概念
5. 【理解】面向对象解决问题的步骤
6. 【理解】举例解释面向对象的思想
7. 【理解】面向过程和面向对象的联系和区别
8. **【理解】能够理解类和对象的区别**
9. 【理解】java中类的作用
10. 【应用】java中对象的作用
11. 【理解】java中类和对象的关系
12. **【应用】能够独立使用成员变量和局部变量**
13. 【理解】成员变量和局部变量的概念
14. 【应用】成员变量和局部变量的用法
15. 【理解】成员变量和局部变量的区别
16. **【应用】能够独立使用封装**
17. 【理解】封装的概念
18. 【理解】java中封装的体现
19. 【理解】使用封装的好处
20. 【理解】private关键字的作用
21. 【应用】封装的代码实例
22. **【应用】能够独立使用构造函数**
23. 【理解】构造函数的作用
24. 【应用】构造函数的写法
25. 【应用】构造函数的调用方式
26. **【应用】能够独立使用this关键字**
27. 【理解】this关键字的作用
28. 【应用】this关键字的用法

# 面向对象基础

## 编程思想概述

### 面向过程思想概述

我们回想一下，这几天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

### 面向对象思想概述

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。

## 面向对象思想特点及举例

### 面向对象思想特点

面向对象是基于面向过程的编程思想；

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；

A：是一种更符合我们思考习惯的思想

B：可以将复杂的事情简单化

C：将我们从执行者变成了指挥者

角色发生了转换

### 面向对象思想举例

买电脑：

面向过程：我要买电脑--我要明确买电脑的意义--上网查对应的参数信息--去电脑城买电脑--讨价还价--买回电脑

面向对象：我要买电脑--让采购去给我买电脑--买回电脑

洗衣服：

面向过程：把衣服脱下来--找一个盆--放点洗衣粉--加点水--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--扔给全自动洗衣机--晾起来

## 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的事物，实现信息化。比如：去超市买东西的计费系统，去银行办业务的系统。

我们如何表示一个现实世界事物呢：

A：属性 就是该事物的描述信息

B：行为 就是该事物能够做什么

举例：学生事物

我们学习的Java语言最基本单位是类，所以，我们就应该把事物用一个类来体现。

类：是一组相关的属性和行为的集合

对象：是该类事物的具体体现

举例：

类 学生

对象 班长就是一个对象

# 类和对象

## 类的定义

### 现实世界的事物

类是用来描述现实世界的事物的。

事物：

属性 事物的描述信息

行为 事物能做什么

### 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

### 定义类其实就是定义类的成员(成员变量和成员方法)

1: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。

初始化值：不需要给初始化值。

2:成员方法 和以前定义方法是一样的。

只不过把static去掉，面向对象进阶在详细讲解static的作用。

## 类与对象的案例

### 学生类(讲解)

#### 如何定义

按照事物到类的过程一步步分析

#### 案例代码一

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: Demo  \* **@Description**: 学生类的分析  \* **@date** 2017年11月3日 下午4:17:21  \* Company www.igeekhome.com  \*  \* 需求：写一个学生类  \*  \* 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  \*  \* 1: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。  \* 初始化值：不需要给初始化值。  \* 2:成员方法 和以前定义方法是一样的。  \* 只不过把static去掉，面向对象进阶在详细讲解static的作用。  \*  \*/  **public** **class** Student {  //成员变量    /\*\*  \* **@Fields** name : 姓名  \*/  String name;    /\*\*  \* **@Fields** age : 年龄  \*/  **int** age;    //成员方法  /\*\*  \* **@Title**: study  \* **@Description**: 学习的方法  \*/  **public** **void** study(){  System.***out***.println("好好学习，天天向上！");  }    /\*\*  \* **@Title**: eat  \* **@Description**: 吃饭的方法  \*/  **public** **void** eat(){  System.***out***.println("学习饿了，要吃饭。");  }  } |

#### 如何使用

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，就必须首先拥有该类的对象。

我们该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？

创建对象的格式：

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

#### 代码案例二

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: Student类的使用  \* **@date** 2017年11月10日 上午11:03:26  \* Company www.igeekhome.com  \*  \* Student类是一个学生事物描述类，所以main方法不适合放在它里面  \* 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，就必须首先拥有该类的对象。  \*  \* 我们该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？  \* 创建对象的格式：  \* 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** StudentDemo {  **public** **static** **void** main(String[] args) {  //格式：类名 对象名=new 类名();  Student s=**new** Student();  System.***out***.println("s:"+s);//s:com.igeek\_02.Student@feb48    //直接输出成员变量值  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="吴京";  s.age=35;    //再次输出成员变量的值  System.***out***.println("姓名："+s.name);//吴京  System.***out***.println("年龄："+s.age);//35  System.***out***.println("---------------");    //调用成员方法  s.study();  s.eat();  }  } |

### 手机类练习

#### 代码案例三

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: Phone  \* **@Description**: 手机类  \* **@date** 2017年11月10日 下午4:10:51  \* Company www.igeekhome.com  \*  \* 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 {  /\*\*  \* **@Fields** brand : 品牌  \*/  String brand;  /\*\*  \* **@Fields** price : 价格  \*/  **int** price;  /\*\*  \* **@Fields** color : 颜色  \*/  String color;    /\*\*  \* **@Title**: call  \* **@Description**: 打电话  \* **@param** name  \*/  **public** **void** call(String name){  System.***out***.println("给"+name+"打电话");  }    /\*\*  \* **@Title**: sendMessage  \* **@Description**: 发短信  \*/  **public** **void** sendMessage(){  System.***out***.println("群发短信");  }  } |

#### 代码案例四

手机类的使用

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: PhoneDemo  \* **@Description**: 手机类的测试类  \* **@date** 2017年11月10日 下午4:19:28  \* Company www.igeekhome.com  \*  \*/  **public** **class** PhoneDemo {  **public** **static** **void** main(String[] args) {  //创建对象  Phone p=**new** Phone();    //输出成员变量  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="华为Mate10";  p.price=3999;  p.color="黑色";    //再次输出成员变量  System.***out***.println("品牌："+p.brand);//华为Mate10  System.***out***.println("价格："+p.price);//3999  System.***out***.println("颜色："+p.color);//黑色  System.***out***.println("-------------------");    //调用成员方法  p.call("吴京");  p.sendMessage();  }  } |

## 对象内存图

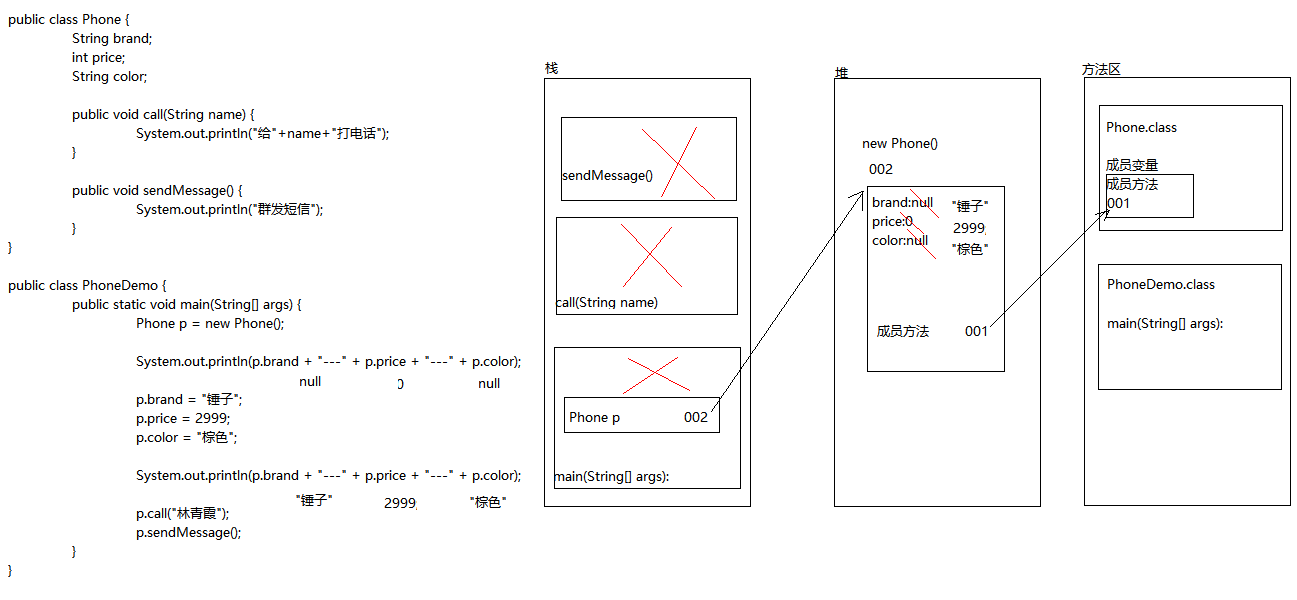
### 一个对象的情况

#### 代码案例五

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: Phone  \* **@Description**: 手机类  \* **@date** 2017年11月10日 下午4:10:51  \* Company www.igeekhome.com  \*  \* 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 {  /\*\*  \* **@Fields** brand : 品牌  \*/  String brand;  /\*\*  \* **@Fields** price : 价格  \*/  **int** price;  /\*\*  \* **@Fields** color : 颜色  \*/  String color;    /\*\*  \* **@Title**: call  \* **@Description**: 打电话  \* **@param** name  \*/  **public** **void** call(String name){  System.***out***.println("给"+name+"打电话");  }    /\*\*  \* **@Title**: sendMessage  \* **@Description**: 发短信  \*/  **public** **void** sendMessage(){  System.***out***.println("群发短信");  }  } |

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: PhoneDemo  \* **@Description**: 手机类的测试类  \* **@date** 2017年11月10日 下午4:19:28  \* Company www.igeekhome.com  \*  \*/  **public** **class** PhoneDemo {  **public** **static** **void** main(String[] args) {  //创建对象  Phone p=**new** Phone();    //输出成员变量  System.***out***.println(p.brand+"---"+p.price+"---"+p.color);    //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  System.***out***.println(p.brand+"---"+p.price+"---"+p.color);    //调用成员方法  p.call("林青霞");  p.sendMessage();  }  } |

#### 1个对象的内存图

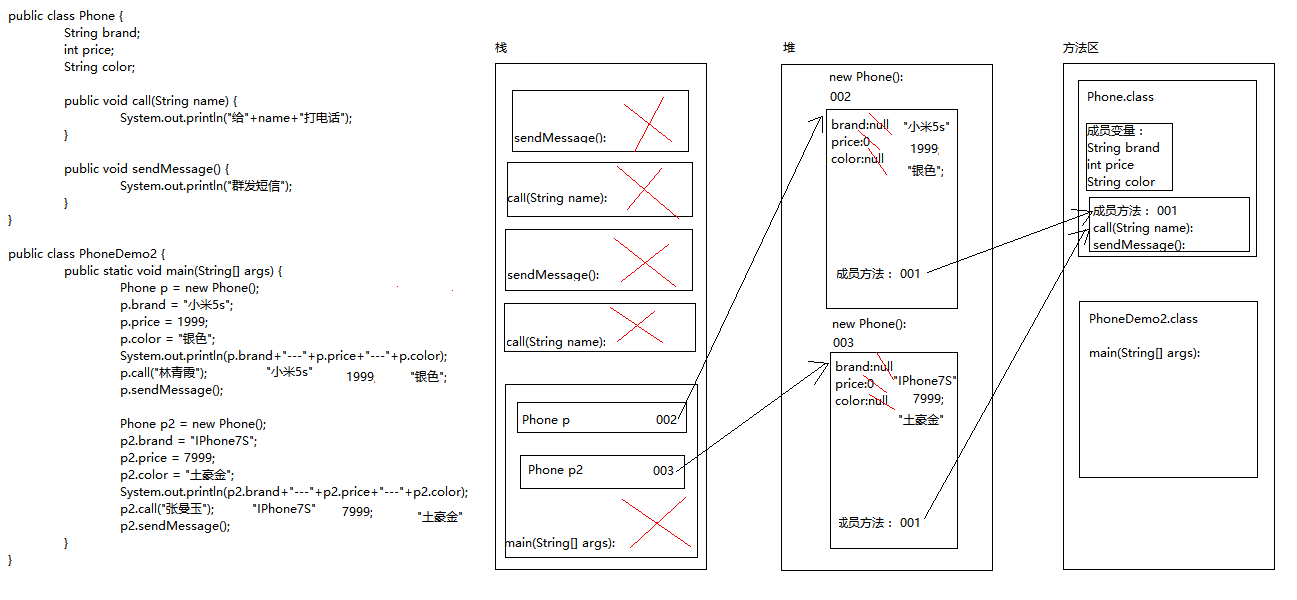
一个对象的基本初始化过程

### 两个对象的情况一

#### 代码案例六

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: PhoneDemo2  \* **@Description**: 手机类的测试类  \* **@date** 2017年11月10日 下午4:19:28 Company www.igeekhome.com  \*  \*/  **public** **class** PhoneDemo2 {  **public** **static** **void** main(String[] args) {  // 创建对象  Phone p = **new** Phone();  // 给成员变量赋值  p.brand = "小米5s";  p.price = 1999;  p.color = "银色";  // 输出成员变量  System.***out***.println(p.brand + "---" + p.price + "---" + p.color);  // 调用成员方法  p.call("林青霞");  p.sendMessage();  // 创建对象  Phone p2 = **new** Phone();  // 给成员变量赋值  p2.brand = "IPhone7S";  p2.price = 7999;  p2.color = "土豪金";  // 输出成员变量  System.***out***.println(p2.brand + "---" + p2.price + "---" + p2.color);  // 调用成员方法  p2.call("张曼玉");  p2.sendMessage();  }  } |

#### 2个对象的内存图



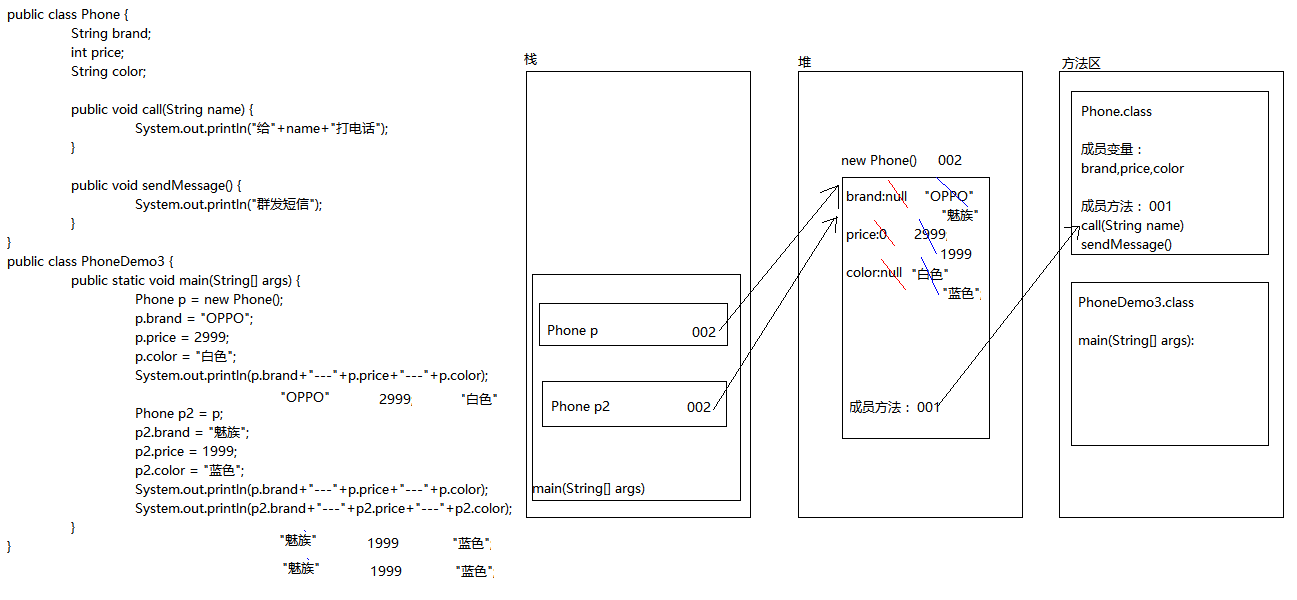
方法的共用

### 两个对象的情况二

#### 代码案例七

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: PhoneDemo3  \* **@Description**: 手机类的测试类  \* **@date** 2017年11月10日 下午4:19:28 Company www.igeekhome.com  \*  \*/  **public** **class** PhoneDemo3 {  **public** **static** **void** main(String[] args) {  // 创建对象  Phone p = **new** Phone();  // 给成员变量赋值  p.brand = "OPPO";  p.price = 2999;  p.color = "白色";  // 输出成员变量  System.***out***.println(p.brand + "---" + p.price + "---" + p.color);  // 把p对象赋值给p2  Phone p2 = p;  // 给成员变量赋值  p2.brand = "魅族";  p2.price = 1999;  p2.color = "蓝色";  // 再次输出成员变量  System.***out***.println(p.brand + "---" + p.price + "---" + p.color);  System.***out***.println(p2.brand + "---" + p2.price + "---" + p2.color);  }  } |

#### 2个对象的内存图



两个引用指向同一个对象

## 成员变量和局部变量的区别

### 在类中的位置不同

成员变量： 类中方法外

局部变量： 方法内或者方法声明上(形式参数)

### 在内存中的位置不同

成员变量： 堆内存

局部变量： 栈内存

### 生命周期不同

成员变量： 随着对象的存在而存在，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

### 初始化值不同

成员变量： 有默认的初始化值

局部变量： 没有默认的初始化值，必须先定义，赋值，才能使用。

### 代码案例八

|  |
| --- |
| **package** com.igeek\_04;  /\*\*  \* **@ClassName**: Variable  \* **@Description**: 成员变量和局部变量的区别：  \* **@date** 2017年11月10日 下午5:03:05  \* Company www.igeekhome.com  \* 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： 类中方法外  \* 局部变量： 方法内或者方法声明上(形式参数)  \* B：在内存中的位置不同  \* 成员变量： 堆内存  \* 局部变量： 栈内存  \* C：生命周期不同  \* 成员变量： 随着对象的存在而存在，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： 有默认的初始化值  \* 局部变量： 没有默认的初始化值，必须先定义，赋值，才能使用。  \*/  **public** **class** Variable {  /\*\*  \* **@Fields** x : 成员变量x  \*/  **int** x;    /\*\*  \* **@Title**: show  \* **@Description**: 显示变量  \*/  **public** **void** show(){  //局部变量  **int** y=0;    System.***out***.println(x);  System.***out***.println(y);  }  } |

## private关键字

### private关键字：

是一个权限修饰符。

可以修饰成员(成员变量和成员方法)

被private修饰的成员只在本类中才能访问。

### 代码案例九

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月10日 下午5:14:25  \* Company www.igeekhome.com  \*  \* 通过对象直接访问成员变量，会存在数据安全问题  \* 这个时候，我们就想能不能不让外界的对象直接访问成员变量呢？  \* 能！  \* 如何实现呢？  \* private关键字  \*  \* private：  \* 是一个权限修饰符。  \* 可以修饰成员(成员变量和成员方法)  \* 被private修饰的成员只能在本类中才能访问。  \*  \* 针对private修饰的成员变量，我们会相应的提供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 {  /\*\*  \* **@Fields** name : 姓名  \*/  String name;  /\*\*  \* **@Fields** age : 年龄  \*/  //int age;  **private** **int** age;  /\*\*  \* **@Title**: setAge  \* **@Description**: 设置年龄  \* **@param** a  \*/  **public** **void** setAge(**int** a){  **if**(a<0||a>200){  System.***out***.println("你给的年龄有误");  }**else**{  age=a;  }  }    /\*\*  \* **@Title**: getAge  \* **@Description**: 获取年龄  \* **@return**  \*/  **public** **int** getAge(){  **return** age;  }    /\*\*  \* **@Title**: show  \* **@Description**: 显示学生信息  \*/  **public** **void** show(){  System.***out***.println("姓名是："+name+"，年龄是："+age);  }  } |

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 学生测试类  \* **@date** 2017年11月10日 下午5:17:34  \* Company www.igeekhome.com  \*  \*/  **public** **class** StudentDemo {  **public** **static** **void** main(String[] args) {  //创建学生对象  Student s=**new** Student();  s.show();    s.name="林青霞";  //s.age=28;  //s.age=-28;  //s.setAge(-28);  s.setAge(28);  s.show();  }  } |

### 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

一个标准的案例的使用

### 代码案例十

|  |
| --- |
| **package** com.igeek\_05\_03;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月10日 下午5:14:25  \* Company www.igeekhome.com  \*  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;    /\*\*  \* **@Title**: setName  \* **@Description**: 设置姓名  \* **@param** n  \*/  **public** **void** setName(String n){  name=n;  }    /\*\*  \* **@Title**: getName  \* **@Description**: 获取姓名  \* **@return**  \*/  **public** String getName(){  **return** name;  }    /\*\*  \* **@Title**: setAge  \* **@Description**: 设置年龄  \* **@param** a  \*/  **public** **void** setAge(**int** a){  age=a;  }    /\*\*  \* **@Title**: getAge  \* **@Description**: 获取年龄  \* **@return**  \*/  **public** **int** getAge(){  **return** age;  }    } |

|  |
| --- |
| **package** com.igeek\_05\_03;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 学生类的测试类  \* **@date** 2017年11月10日 下午5:17:34  \* Company www.igeekhome.com  \*  \*/  **public** **class** StudentDemo {  **public** **static** **void** main(String[] args) {  //创建学生对象  Student s=**new** Student();  System.***out***.println(s.getName()+"---"+s.getAge());    s.setName("吴京");  s.setAge(35);  System.***out***.println(s.getName()+"---"+s.getAge());  }  } |

# 封装

## 什么是封装

### 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。就像上章说的年龄。

### 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

### 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

this: 代表所在类的对象引用

记住： 方法被哪个对象调用，this就代表那个对象

什么时候使用this呢? 局部变量隐藏成员变量

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月10日 下午5:14:25  \* Company www.igeekhome.com  \*  \* 起名字我们要求做到见名知意。  \* 而我们现在的代码中的n和a就没有做到见名知意，所以我们要改进。  \*  \* 如果有局部变量名和成员变量名相同，在局部使用的时候，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  \*  \* 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就代表那个对象  \*  \* 使用场景： 局部变量隐藏成员变量  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;    /\*\*  \* **@Title**: setName  \* **@Description**: 设置姓名  \* **@param** n  \*/  **public** **void** setName(String name){//吴京  //name=name;  **this**.name=name;  }    /\*\*  \* **@Title**: getName  \* **@Description**: 获取姓名  \* **@return**  \*/  **public** String getName(){  **return** name;  }    /\*\*  \* **@Title**: setAge  \* **@Description**: 设置年龄  \* **@param** a  \*/  **public** **void** setAge(**int** age){  //age=age;  **this**.age=age;  }    /\*\*  \* **@Title**: getAge  \* **@Description**: 获取年龄  \* **@return**  \*/  **public** **int** getAge(){  **return** age;  }    } |

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 学生类的测试类  \* **@date** 2017年11月10日 下午5:17:34  \* Company www.igeekhome.com  \*  \*/  **public** **class** StudentDemo {  **public** **static** **void** main(String[] args) {  //创建学生对象  Student s=**new** Student();  System.***out***.println(s.getName()+"---"+s.getAge());    s.setName("吴京");  s.setAge(35);  System.***out***.println(s.getName()+"---"+s.getAge());  }  } |

## 构造方法

### 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

#### 代码案例十一

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月11日 上午9:23:37  \* Company www.igeekhome.com  \*  \* 构造方法：  \* 给对象的数据进行初始化  \*  \* 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  \*  \*/  **public** **class** Student {  /\*\*  \* **@Title**: Student类的无参构造方法  \*/  **public** Student(){  System.***out***.println("这是构造方法");  }  } |

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 学生类的测试类  \* **@date** 2017年11月11日 上午9:26:15  \* Company www.igeekhome.com  \*  \*/  **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(...);  //也就是创建对象的格式  Student s=**new** Student();  }  } |

### 构造方法注意事项

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

#### 代码案例十二

|  |
| --- |
| **package** com.igeek\_03\_03;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月11日 上午9:23:37  \* Company www.igeekhome.com  \*  \* 构造方法：  \* 给对象的数据进行初始化  \*  \* 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  \*  \* 构造方法注意事项:  \* A:如果你不提供构造方法，系统会给出一个默认无参构造方法  \* B:如果你提供了构造方法，系统将不再提供默认无参构造方法  \* 这个时候，如果想使用无参构造方法，就必须自己提供  \* 推荐：自己给出无参构造方法  \* C:构造方法也是可以重载的  \*  \* 成员变量赋值：  \* A：setXxx()方法  \* B：带参数构造方法  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;    /\*\*  \* **@Title**: Student类的无参构造方法  \*/  /\*  public Student(){  System.out.println("这是构造方法");  }\*/    **public** Student(){}    /\*\*  \* **@Title**: Student  \* **@param** name  \*/  **public** Student(String name){  **this**.name=name;  }  /\*\*  \* **@Title**: Student  \* **@param** age  \*/  **public** Student(**int** age){  **this**.age=age;  }    /\*\*  \* **@Title**: Student  \* **@param** name  \* **@param** age  \*/  **public** Student(String name,**int** age){  **this**.name=name;  **this**.age=age;  }    /\*\*  \* **@Title**: show  \* **@Description**: 显示学生信息方法  \*/  **public** **void** show(){  System.***out***.println(name+"---"+age);  }  } |

|  |
| --- |
| **package** com.igeek\_03\_03;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 学生类的测试类  \* **@date** 2017年11月11日 上午9:26:15  \* Company www.igeekhome.com  \*  \*/  **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(...);  //也就是创建对象的格式  //public Student()  Student s=**new** Student();  s.show();    //public Student(String name)  Student s2=**new** Student("吴京");  s2.show();    //public Student(int age)  Student s3=**new** Student(35);  s3.show();    //public Student(String name,int age)  Student s4=**new** Student("吴京", 35);  s4.show();  }  } |

## 标准类的代码写法和测试

### 类

- 成员变量

- 构造方法

* 无参构造方法
* 带参构造方法

- 成员方法

* + - getXxx()
    - setXxx()

### 给成员变量赋值的方式

无参构造方法+setXxx()

带参构造方法

### 面向对象标准案例

#### 学生类(老师讲)

#### 代码案例十三

|  |
| --- |
| **package** com.igeek\_04;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月11日 上午9:23:37  \* Company www.igeekhome.com  \*  \*/  **public** **class** Student {  //成员变量  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;    //构造方法  /\*\*  \* **@Title**: Student  \*/  **public** Student(){}    /\*\*  \* **@Title**: Student  \* **@param** name  \* **@param** age  \*/  **public** Student(String name,**int** age){  **this**.name=name;  **this**.age=age;  }    //成员方法  /\*\*  \* **@Title**: setName  \* **@Description**: 设置姓名  \* **@param** name  \*/  **public** **void** setName(String name){  **this**.name=name;  }    /\*\*  \* **@Title**: getName  \* **@Description**: 获取姓名  \* **@return**  \*/  **public** String getName(){  **return** name;  }    /\*\*  \* **@Title**: setAge  \* **@Description**: 设置年龄  \* **@param** age  \*/  **public** **void** setAge(**int** age){  **this**.age=age;  }    /\*\*  \* **@Title**: getAge  \* **@Description**: 获取年龄  \* **@return**  \*/  **public** **int** getAge(){  **return** age;  }    } |

|  |
| --- |
| **package** com.igeek\_04;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 学生类的测试类  \* **@date** 2017年11月11日 上午9:26:15  \* Company www.igeekhome.com  \*  \*/  **public** **class** StudentDemo {  **public** **static** **void** main(String[] args) {  //无参构造方法+setXxx()方法  Student s=**new** Student();  s.setName("吴京");  s.setAge(35);  System.***out***.println(s.getName()+"---"+s.getAge());    //带参构造方法  Student s2=**new** Student("吴京", 35);  System.***out***.println(s2.getName()+"---"+s2.getAge());  }  } |

手机类(学生练)

汽车类(学生练)

## 类名作为形式参数和返回值

### 类名作为形式参数案例

要的其实是该类的对象

#### 代码案例十四

|  |
| --- |
| **package** com.igeek\_05\_01;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月11日 上午10:20:28  \* Company www.igeekhome.com  \*  \*/  **public** **class** Student {  /\*\*  \* **@Title**: study  \* **@Description**: 学习方法  \*/  **public** **void** study(){  System.***out***.println("好好学习，天天向上");  }  } |

|  |
| --- |
| **package** com.igeek\_05\_01;  /\*\*  \* **@ClassName**: Teacher  \* **@Description**: 老师类  \* **@date** 2017年11月11日 上午10:21:13  \* Company www.igeekhome.com  \*  \*/  **public** **class** Teacher {  /\*\*  \* **@Title**: test  \* **@Description**: 测试学生学习的方法  \* **@param** s  \*/  **public** **void** test(Student s){  s.study();  }  } |

|  |
| --- |
| **package** com.igeek\_05\_01;  /\*\*  \* **@ClassName**: Test  \* **@Description**: 调用Teacher的test()方法  \* **@date** 2017年11月11日 上午10:22:21  \* Company www.igeekhome.com  \*  \* 类名作为形式参数时：其实这里实际需要传递的是该类的对象  \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  Teacher t=**new** Teacher();  Student s=**new** Student();  t.test(s);  }  } |

### 类名作为返回值案例

返回的其实是该类的对象

#### 代码案例十五

|  |
| --- |
| **package** com.igeek\_05\_02;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年11月11日 上午10:20:28  \* Company www.igeekhome.com  \*  \*/  **public** **class** Student {  /\*\*  \* **@Title**: study  \* **@Description**: 学习方法  \*/  **public** **void** study(){  System.***out***.println("好好学习，天天向上");  }  } |

|  |
| --- |
| **package** com.igeek\_05\_02;  /\*\*  \* **@ClassName**: Teacher  \* **@Description**: 老师类  \* **@date** 2017年11月11日 上午10:21:13  \* Company www.igeekhome.com  \*  \*/  **public** **class** Teacher {  /\*\*  \* **@Title**: getStudent  \* **@Description**: 获取学生对象  \* **@return**  \*/  **public** Student getStudent(){  Student s=**new** Student();  **return** s;  }  } |

|  |
| --- |
| **package** com.igeek\_05\_02;  /\*\*  \* **@ClassName**: Test  \* **@Description**: 通过Teacher对象得到Student对象，然后调用Student对象的方法  \* **@date** 2017年11月11日 上午10:22:21  \* Company www.igeekhome.com  \*  \* 如果方法的返回值是类名：其实返回的是该类的对象  \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  Teacher t=**new** Teacher();  Student s=t.getStudent();  s.study();  }  } |

重点和总结

1、面向对象的基本知识

2、类和对象的关系

3、什么是封装