Java进阶1 第1天

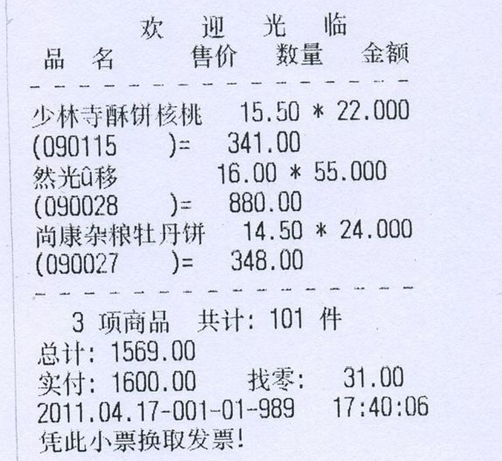
**【学习目标】理解、了解、应用、记忆**

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. **【应用】独立编写超市购物小票案例**
2. 【理解】阐述超市购物小票需求及所用到的知识点
3. 【应用】独立编码定义商品项类
4. 【应用】独立编码完成小票数据初始化功能
5. 【应用】独立编码完成购物小票主干逻辑
6. 【应用】独立编码完成为商品数量和金额赋值功能
7. 【应用】独立编码完成购物小票打印功能
8. **【理解】阐述并解释面向对象相关内容**
9. 【应用】独立编码完成一个Person类的定义
10. 【应用】能够动手操作Eclipse中的三个视图
11. 【理解】阐述基本数据类型在内存中如何存储
12. 【理解】阐述引用数据类型在内存中如何存储
13. 【理解】解释this关键字的作用及用法
14. 【理解】阐述匿名对象的使用格式及使用场景

# 基础复习超市购物小票案例

## 超市购物小票需求分析



票脚：统计和提醒信息

票体：主体内容

票头：说明信息

模拟真实购物逻辑，以上述图片为需求原型，根据不同购买物品，完成购物小票内容打印到控制台。(简化项目逻辑，票脚部分只完成数据统计)

**数据：**

 将超市购物小票案例中零散数据(名称、货号、单价、数量、计价单位、金额)封装为货物对象。

 所有的货物对象放置到集合中统一管理。

**逻辑：**

用户循环进行三个操作：

 输入购买数量，代表为所购买货物的数量赋值，从而计算每项商品金额

 打印小票，将已有数据打印

 退出系统(因为该程序为循环操作，无法终止，如果不想再进行操作，则退出系统)

## 超市购物小票知识点分析

**复习知识点:**

变量定义：记录部分数据

集合的使用：用于存储多个完整数据

运算符的使用：计算变量

流程控制语句的使用：

if语句：用于判断是否购买了商品

switch语句：用于进行三个操作(输入购买数量、打印小票、退出系统)的分支

for语句：switch外部使用for循环，当执行一个操作后，继续进行下一个操 作选择

方法封装：

由于代码量比较多，可以将输入购买数量、打印小票封装为方法使用

字符串/键盘录入：

打印小票完整过程均为字符串的拼写及打印，数量输入为键盘录入Scanner 完成

**步骤:**

定义GoodsItem类(名称,货号,单价,数量,计价单位,金额)

成员位置定义集合，定义方法向集合中存储所有GoodsItem数据

实现主干逻辑

打印欢迎语句

使用for/while进行循环进行操作

使用switch给出操作选择：1输入购买数量2打印小票3退出

完成switch中三个case的逻辑

运用方法封装功能：将1输入购买数量、2打印小票封装为独立方法，在case中 调用，提高代码可读性。

## 超市购物小票商品项类的定义

商品项类，将要购买的商品封装到一个GoodItems类中,编码如下:

### 案例代码一

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: GoodsItem  \* **@Description**: 商品项类  \* **@date** 2017年11月13日 上午10:41:21  \* Company www.igeekhome.com  \*  \* 定义商品项类:  \* 名称,货号,单价,数量,计价单位,金额  \*  \*/  **public** **class** GoodsItem {  // 成员变量  /\*\*  \* **@Fields** name : 名称  \*/  **private** String name;  /\*\*  \* **@Fields** id : 货号  \*/  **private** String id;  /\*\*  \* **@Fields** price : 单价  \*/  **private** **double** price;  /\*\*  \* **@Fields** number : 数量  \*/  **private** **int** number;  /\*\*  \* **@Fields** unit : 计价单位  \*/  **private** String unit;  /\*\*  \* **@Fields** money : 金额  \*/  **private** **double** money;  // 构造方法  /\*\*  \* **@Title**: GoodsItem  \*/  **public** GoodsItem() {  }  /\*\*  \* **@Title**: GoodsItem  \* **@param** name  \* **@param** id  \* **@param** price  \* **@param** number  \* **@param** unit  \* **@param** money  \*/  **public** GoodsItem(String name, String id, **double** price, **int** number, String unit, **double** money) {  **this**.name = name;  **this**.id = id;  **this**.price = price;  **this**.number = number;  **this**.unit = unit;  **this**.money = money;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the id  \*/  **public** String getId() {  **return** id;  }  /\*\*  \* **@param** id the id to set  \*/  **public** **void** setId(String id) {  **this**.id = id;  }  /\*\*  \* **@return** the price  \*/  **public** **double** getPrice() {  **return** price;  }  /\*\*  \* **@param** price the price to set  \*/  **public** **void** setPrice(**double** price) {  **this**.price = price;  }  /\*\*  \* **@return** the number  \*/  **public** **int** getNumber() {  **return** number;  }  /\*\*  \* **@param** number the number to set  \*/  **public** **void** setNumber(**int** number) {  **this**.number = number;  }  /\*\*  \* **@return** the unit  \*/  **public** String getUnit() {  **return** unit;  }  /\*\*  \* **@param** unit the unit to set  \*/  **public** **void** setUnit(String unit) {  **this**.unit = unit;  }  /\*\*  \* **@return** the money  \*/  **public** **double** getMoney() {  **return** money;  }  /\*\*  \* **@param** money the money to set  \*/  **public** **void** setMoney(**double** money) {  **this**.money = money;  }  } |

## 超市购物小票初始化数据

成员位置定义集合，定义方法向集合中存储所有GoodsItem数据

### 案例代码二

|  |
| --- |
| **package** com.igeek\_04;  **import** java.util.ArrayList;  **import** com.igeek\_03.GoodsItem;  /\*\*  \* **@ClassName**: ShoppingReceipt  \* **@Description**: 超市购物小票类  \* **@date** 2017年11月13日 上午10:48:20  \* Company www.igeekhome.com  \*  \*/  **public** **class** ShoppingReceipt {    /\*\*  \* **@Fields** data : 成员位置定义集合,存储所有的商品项对象  \*/  **static** ArrayList<GoodsItem> *data* = **new** ArrayList<GoodsItem>();    **public** **static** **void** main(String[] args) {  //为集合准备数据  *initData*();  }  /\*\*  \* **@Title**: initData  \* **@Description**: 定义方法,向成员位置的集合赋值  \*/  **public** **static** **void** initData() {  //创建多个商品项对象  GoodsItem sls = **new** GoodsItem("少林寺核桃", "090115", 15.5, 0, "个", 0);  GoodsItem shk = **new** GoodsItem("尚康饼干", "090027", 14.5, 0, "个", 0);  //将多个商品项对象放入集合中  *data*.add(sls);  *data*.add(shk);  }  } |

## 超市购物小票主干逻辑

### 实现主干逻辑

打印欢迎语句

使用for/while进行循环进行操作

使用switch给出操作选择：1输入购买数量2打印小票3退出

完成switch中三个case的逻辑

运用方法封装功能：将1输入购买数量、2打印小票封装为独立方法，在case中调用，提高代码可读性。

### 案例代码三

|  |
| --- |
| **package** com.igeek\_05;  **import** java.util.ArrayList;  **import** java.util.Scanner;  **import** com.igeek\_03.GoodsItem;  /\*\*  \* **@ClassName**: ShoppingReceipt  \* **@Description**: 超市购物小票类  \* **@date** 2017年11月13日 上午10:48:20  \* Company www.igeekhome.com  \*  \*/  **public** **class** ShoppingReceipt {    /\*\*  \* **@Fields** data : 成员位置定义集合,存储所有的商品项对象  \*/  **static** ArrayList<GoodsItem> *data* = **new** ArrayList<GoodsItem>();    **public** **static** **void** main(String[] args) {  //为集合准备数据  *initData*();    //打印欢迎语句  System.***out***.println("欢迎使用超市管理系统!");    //使用死循环完成反复操作的逻辑  //for(;;) {}  **while**(**true**) {  //使用switch给出操作选择：1输入购买数量2打印小票3退出  //提示信息  System.***out***.println("请输入您要进行的操作:1输入购买数量 2打印小票 3退出");    //通过键盘录入得到数据  Scanner sc = **new** Scanner(System.***in***);  **int** optNumber = sc.nextInt();  //完成switch中三个case的逻辑  **switch** (optNumber) {  **case** 1:  System.***out***.println("给所有数据量赋值");  **break**;  **case** 2:  //调用打印购物小票的方法  System.***out***.println("打印购物小票");  **break**;  **case** 3:  System.***out***.println("欢迎下次光临");  //退出程序  System.*exit*(0);  **default**:  System.***out***.println("请输入正确的数字!");  **break**;  }  }  }  /\*\*  \* **@Title**: initData  \* **@Description**: 定义方法,向成员位置的集合赋值  \*/  **public** **static** **void** initData() {  //创建多个商品项对象  GoodsItem sls = **new** GoodsItem("少林寺核桃", "090115", 15.5, 0, "个", 0);  GoodsItem shk = **new** GoodsItem("尚康饼干", "090027", 14.5, 0, "个", 0);  //将多个商品项对象放入集合中  *data*.add(sls);  *data*.add(shk);  }  } |

## 超市购物小票为商品项赋值数量与金额

定义一个方法为所有的数据赋值数量和金额

### 案例代码四

|  |
| --- |
| **package** com.igeek\_06;  **import** java.util.ArrayList;  **import** java.util.Scanner;  **import** com.igeek\_03.GoodsItem;  /\*\*  \* **@ClassName**: ShoppingReceipt  \* **@Description**: 超市购物小票类  \* **@date** 2017年11月13日 上午10:48:20  \* Company www.igeekhome.com  \*  \*/  **public** **class** ShoppingReceipt {    /\*\*  \* **@Fields** data : 成员位置定义集合,存储所有的商品项对象  \*/  **static** ArrayList<GoodsItem> *data* = **new** ArrayList<GoodsItem>();    **public** **static** **void** main(String[] args) {  //为集合准备数据  *initData*();    //打印欢迎语句  System.***out***.println("欢迎使用超市管理系统!");    //使用死循环完成反复操作的逻辑  //for(;;) {}  **while**(**true**) {  //使用switch给出操作选择：1输入购买数量2打印小票3退出  //提示信息  System.***out***.println("请输入您要进行的操作:1输入购买数量 2打印小票 3退出");    //通过键盘录入得到数据  Scanner sc = **new** Scanner(System.***in***);  **int** optNumber = sc.nextInt();  //完成switch中三个case的逻辑  **switch** (optNumber) {  **case** 1:  //System.out.println("给所有数据量赋值");  //调用给所有数据的数量与金额赋值方法  *enterNumber*();  **break**;  **case** 2:  //调用打印购物小票的方法  System.***out***.println("打印购物小票");  **break**;  **case** 3:  System.***out***.println("欢迎下次光临");  //退出程序  System.*exit*(0);  **default**:  System.***out***.println("请输入正确的数字!");  **break**;  }  }  }  /\*\*  \* **@Title**: initData  \* **@Description**: 定义方法,向成员位置的集合赋值  \*/  **public** **static** **void** initData() {  //创建多个商品项对象  GoodsItem sls = **new** GoodsItem("少林寺核桃", "090115", 15.5, 0, "个", 0);  GoodsItem shk = **new** GoodsItem("尚康饼干", "090027", 14.5, 0, "个", 0);  //将多个商品项对象放入集合中  *data*.add(sls);  *data*.add(shk);  }      /\*\*  \* **@Title**: enterNumber  \* **@Description**: 为所有的数据赋值数量与金额  \*/  **public** **static** **void** enterNumber() {    //集合的遍历  **for** (**int** i = 0; i <*data*.size(); i++) {  //依次获取到集合中每一个元素  GoodsItem thisGoods = *data*.get(i);    //提示用户,输入该商品项的数量  //获取该商品项名称  String thisGoodsName = thisGoods.getName();  System.***out***.println("请输入"+thisGoodsName+"的购买数量");    //键盘录入接受商品数量  Scanner sc = **new** Scanner(System.***in***);  **int** thisGoodsNumber = sc.nextInt();  //根据数量计算金额金额 = 单价 \* 数量  **double** thisGoodsMoney = thisGoods.getPrice() \* thisGoodsNumber;    //为该商品的数量与金额赋值  thisGoods.setNumber(thisGoodsNumber);  thisGoods.setMoney(thisGoodsMoney);  }  }  } |

## 超市购物小票打印小票

定义打印超市购物小票方法完成小票打印功能

### 案例代码五

|  |
| --- |
| **package** com.igeek\_07;  **import** java.util.ArrayList;  **import** java.util.Scanner;  **import** com.igeek\_03.GoodsItem;  /\*\*  \* **@ClassName**: ShoppingReceipt  \* **@Description**: 超市购物小票类  \* **@date** 2017年11月13日 上午10:48:20  \* Company www.igeekhome.com  \*  \*/  **public** **class** ShoppingReceipt {    /\*\*  \* **@Fields** data : 成员位置定义集合,存储所有的商品项对象  \*/  **static** ArrayList<GoodsItem> *data* = **new** ArrayList<GoodsItem>();    **public** **static** **void** main(String[] args) {  //为集合准备数据  *initData*();    //打印欢迎语句  System.***out***.println("欢迎使用超市管理系统!");    //使用死循环完成反复操作的逻辑  //for(;;) {}  **while**(**true**) {  //使用switch给出操作选择：1输入购买数量2打印小票3退出  //提示信息  System.***out***.println("请输入您要进行的操作:1输入购买数量 2打印小票 3退出");    //通过键盘录入得到数据  Scanner sc = **new** Scanner(System.***in***);  **int** optNumber = sc.nextInt();  //完成switch中三个case的逻辑  **switch** (optNumber) {  **case** 1:  //System.out.println("给所有数据量赋值");  //调用给所有数据的数量与金额赋值方法  *enterNumber*();  **break**;  **case** 2:  //System.out.println("打印购物小票");  //调用打印购物小票的方法  *printReceipt*();  **break**;  **case** 3:  System.***out***.println("欢迎下次光临");  //退出程序  System.*exit*(0);  **default**:  System.***out***.println("请输入正确的数字!");  **break**;  }  }  }  /\*\*  \* **@Title**: initData  \* **@Description**: 定义方法,向成员位置的集合赋值  \*/  **public** **static** **void** initData() {  //创建多个商品项对象  GoodsItem sls = **new** GoodsItem("少林寺核桃", "090115", 15.5, 0, "个", 0);  GoodsItem shk = **new** GoodsItem("尚康饼干", "090027", 14.5, 0, "个", 0);  //将多个商品项对象放入集合中  *data*.add(sls);  *data*.add(shk);  }      /\*\*  \* **@Title**: enterNumber  \* **@Description**: 为所有的数据赋值数量与金额  \*/  **public** **static** **void** enterNumber() {    //集合的遍历  **for** (**int** i = 0; i <*data*.size(); i++) {  //依次获取到集合中每一个元素  GoodsItem thisGoods = *data*.get(i);    //提示用户,输入该商品项的数量  //获取该商品项名称  String thisGoodsName = thisGoods.getName();  System.***out***.println("请输入"+thisGoodsName+"的购买数量");    //键盘录入接受商品数量  Scanner sc = **new** Scanner(System.***in***);  **int** thisGoodsNumber = sc.nextInt();  //根据数量计算金额金额 = 单价 \* 数量  **double** thisGoodsMoney = thisGoods.getPrice() \* thisGoodsNumber;    //为该商品的数量与金额赋值  thisGoods.setNumber(thisGoodsNumber);  thisGoods.setMoney(thisGoodsMoney);  }  }    /\*\*  \* **@Title**: printReceipt  \* **@Description**: 打印小票方法  \*/  **private** **static** **void** printReceipt() {  //static ArrayList<GoodsItem> data = new ArrayList<GoodsItem>();  //票头  System.***out***.println(" 欢迎光临");  System.***out***.println("品名\t售价\t数量\t单位\t金额");  System.***out***.println("-------------------------------------------");  //票体  //定义变量,记录所有的商品数量  **int** totalNumber = 0;  //定义变量,记录所有的商品金额  **double** totalMoney = 0;  //遍历集合  **for** (**int** i = 0; i <*data*.size(); i++) {  //依次获取每一个商品项  GoodsItem g = *data*.get(i);  //打印商品项  System.***out***.println(g.getName()+g.getId()+"\t"+g.getPrice()+"\t"+g.getNumber()+" "+g.getUnit()+"\t"+g.getMoney());    //累加数量与金额  totalNumber += g.getNumber();  totalMoney += g.getMoney();  }  System.***out***.println("-------------------------------------------");  //票脚  System.***out***.println("共"+*data*.size()+"项商品");  System.***out***.println("共"+totalNumber+"件商品");  System.***out***.println("共"+totalMoney+"元");  System.***out***.println();  }  } |

# 面向对象的其他内容

## 一个类的定义及使用回顾

回顾定义一个类的的基本步骤和定义哪些内容

使用类映射现实生活中的事物，其功能封装为方法，属性封装为成员变量。

一个类拥有的成员包括：

成员变量

构造方法

普通方法

getters/setters

### 案例代码六

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Person  \* **@Description**: Person类  \* **@date** 2017年11月13日 上午11:13:43  \* Company www.igeekhome.com  \*  \*/  **public** **class** Person {  // 成员变量  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;  // 构造方法:构造方法不仅仅是为了给成员变量赋值,构造方法只是定义的为创建对象而产生的方法,在构造方法执行过程当中,通过其逻辑是为了给成员变量赋值,也可以有其他的逻辑存在  /\*\*  \* **@Title**: Person  \*/  **public** Person() {  }  /\*\*  \* **@Title**: Person  \* **@param** name  \* **@param** age  \*/  **public** Person(String name, **int** age) {  **this**.name = name;  **this**.age = age;  }  // 普通方法  /\*\*  \* **@Title**: eat  \* **@Description**: 吃饭的方法  \*/  **public** **void** eat() {  System.***out***.println("吃饭");  }  /\*\*  \* **@Title**: showName  \* **@Description**: 显示个人信息  \*/  **public** **void** showName() {  System.***out***.println("我的名字是:" + **this**.name + " 我的年龄是:" + **this**.age);  }  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: PersonDemo  \* **@Description**: Person类的测试类  \* **@date** 2017年11月13日 上午11:17:18  \* Company www.igeekhome.com  \*  \*/  **public** **class** PersonDemo {  **public** **static** **void** main(String[] args) {  //创建对象  Person p = **new** Person("Jack",18);  //调用方法  p.eat();  p.showName();  }  } |

## Eclipse视图与颜色介绍

Eclipse的不同视图:

Package视图:为了方便开发,只显示java文件的视图

Navigator视图:硬盘上真正存储的文件视图

Outline:大纲视图,显示该类的基本成员

Outline视图中显示的成员和方法颜色表示了不同的权限修饰符:

红色:private修饰

蓝色:默认没有修饰符

黄色:protected修饰

绿色:public修饰

## 基本类型的内存解释

### 基本数据的内存图

|  |
| --- |
| C:\Users\Administrator\AppData\Roaming\Tencent\Users\1102780712\QQ\WinTemp\RichOle\HL@)06}ZPZ05QGAM53Q2OMW.png |

**基本类型赋值代码解释**

变量名 具体数值

int p = 100;

变量数据类型 赋值运算符

**基本类型赋值对应内存图**

int p = 100

栈内存栈内存

堆内存

### 案例代码七

|  |
| --- |
| **package** com.igeek\_03;  **import** com.igeek\_01.Person;  /\*\*  \* **@ClassName**: PersonDemo  \* **@Description**: Person类的测试类  \* **@date** 2017年11月13日 上午11:17:18  \* Company www.igeekhome.com  \*  \*/  **public** **class** PersonDemo {  **public** **static** **void** main(String[] args) {  //基本类型数据  **int** a = 100;    //创建对象  Person p = **new** Person("Jack",18);  //调用方法  p.eat();  p.showName();  }  } |

## 引用类型的内存解释

### C:\Users\Administrator\AppData\Roaming\Tencent\Users\1102780712\QQ\WinTemp\RichOle\V7$%CJ~8~E@IN%S3042{XW0.png引用数据内存图

**引用类型赋值代码解释**

变量名 创建对象固定关键字

Person p = new Person(); 具体数值为内存地址

变量数据类型 赋值运算符 具体数值 (0x1234)

创建对象目前固定写法

### 案例代码八

|  |
| --- |
| **package** com.igeek\_04;  **import** com.igeek\_01.Person;  /\*\*  \* **@ClassName**: PersonDemo  \* **@Description**: Person类的测试类  \* **@date** 2017年11月13日 上午11:17:18  \* Company www.igeekhome.com  \*  \*/  **public** **class** PersonDemo {  **public** **static** **void** main(String[] args) {  //引用类型数据  //创建对象  Person p = **new** Person("Jack",18);  //调用方法  p.eat();  p.showName();  }  } |

## this关键字

### this关键字概述

this一般出现在类的一个方法的定义当中，代表当前对象的引用。我们通常用其将本对象中的成员与其他作用域区分开来。

当一个类定义好之后，如果不创建对象this是没有意义的。

一个类可以有多个对象。每个对象都有各自的属性值，各个对象的方法均是类中定义的那个方法逻辑。定义中的this就代表调用方法的这个对象。

Person类

class Person{

private String name;

private int age;

public void showName() {

System.out.println(this.name);

}

set/get方法

}

Person p = new Person();

p.setName(“郭德纲”);

p.setAge(34);

p.showName(); //打印郭德纲

Person p2 = new Person();

p2.setName(“林志颖”);

p2.setAge(34);

p2.showName (); //打印林志颖

p对象 p2对象

String name; 郭德纲

int age; 34

public void showName() { System.out.println(this.name);

}

get/set方法

System.out.println(this.name);

}

get/set方法

String name; 林志颖

int age; 34

public void showName() { System.out.println(this.name);

}

get/set方法

System.out.println(this.name);

}

get/set方法

### 案例代码九

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: Person  \* **@Description**: Person类  \* **@date** 2017年11月13日 上午11:13:43  \* Company www.igeekhome.com  \*  \*/  **public** **class** Person {  // 成员变量  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;  // 构造方法:构造方法不仅仅是为了给成员变量赋值,构造方法只是定义的为创建对象而产生的方法,在构造方法执行过程当中,通过其逻辑是为了给成员变量赋值,也可以有其他的逻辑存在  /\*\*  \* **@Title**: Person  \*/  **public** Person() {  }  /\*\*  \* **@Title**: Person  \* **@param** name  \* **@param** age  \*/  **public** Person(String name, **int** age) {  **this**.name = name;  **this**.age = age;  }  // 普通方法  /\*\*  \* **@Title**: eat  \* **@Description**: 吃饭的方法  \*/  **public** **void** eat() {  System.***out***.println("吃饭");  }  /\*\*  \* **@Title**: showName  \* **@Description**: 显示个人信息  \*/  **public** **void** showName() {  System.***out***.println("我的名字是:" + **this**.name + " 我的年龄是:" + **this**.age);  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the age  \*/  **public** **int** getAge() {  **return** age;  }  /\*\*  \* **@param** age the age to set  \*/  **public** **void** setAge(**int** age) {  **this**.age = age;  }  } |

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: ThisDemo  \* **@Description**: this关键字概述  \* **@date** 2017年11月13日 下午2:07:58  \* Company www.igeekhome.com  \*  \* 使用this可以区分成员变量与局部变量,处理成员变量与局部变量同名的情况  \*  \* 如果一个方法内访问不带this的变量,则遵循以下顺序:先查找局部变量,再查找成员变量  \* 如果一个方法内访问带this的变量,则只在成员位置找  \*  \* this的意义:代表当前对象的引用  \*  \* 类的方法定义过程当中可以定义this,代表当前调用方法的对象,是一个不确定的对象.  \* 即简单记忆为哪个对象调用方法,则方法中的this代表哪个对象  \*/  **public** **class** ThisDemo {  **public** **static** **void** main(String[] args) {  //创建对象  Person p = **new** Person("郭德纲",34);  p.showName();    //创建对象  Person p2 = **new** Person("林志颖",34);  p2.showName();  }  } |

## 匿名对象

### 匿名对象的概述

匿名对象即没有引用的对象，没有名字的对象，与有名对象对比如下：

Person p = new Person();

该对象是有名字为p的一个Person对象，可以调用方法。

如： p.showName();

new Person(); 该对象是没有名字的一个对象，同样可以调用方法。

如：new Person().showName();

### 案例代码十

|  |
| --- |
| **package** com.igeek\_06;  **import** com.igeek\_05.Person;  /\*\*  \* **@ClassName**: AnonymousObjectDemo  \* **@Description**: 匿名对象的使用  \* **@date** 2017年11月13日 下午2:13:30  \* Company www.igeekhome.com  \*  \* 匿名对象：没有引用的对象，没有名字的对象  \* 有名字(引用)的对象，使用引用调用该对象的方法  \* 匿名对象创建之后，立即使用对象自身调用对象即可  \*  \* 有引用的对象，可以通过引用，反复操作同一个对象，可以重用  \* 匿名对象因为没有引用变量指向，所以一般情况下只能使用一次  \*/  **public** **class** AnonymousObjectDemo {  **public** **static** **void** main(String[] args) {  //有名字的对象  Person p = **new** Person("Rose",16);  p.showName();  Person p2 = **new** Person("Jack",18);  p2.setName("牛顿");  p2.showName();    //匿名对象  **new** Person("怀特",12).showName();  **new** Person("特朗普",64).setName("川普");  **new** Person("特朗普",64).showName();  }  } |

重点和总结

1、前面基础知识点的回顾

2、基本类型和引用类型的内存解释