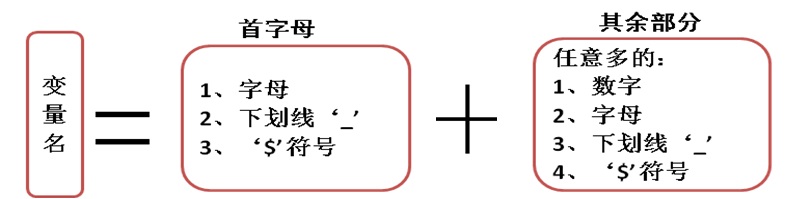
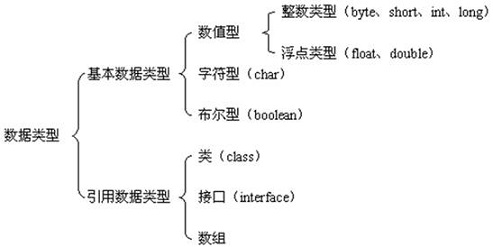
ctril+d删除行

变量名由多单词组成时，**第一个单词的首字母小写**，**其后单词的首字母大写**，俗称骆驼式命名法（也称驼峰命名法），如 myAge



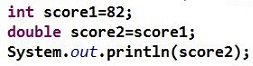
Java 语言是一种**强类型**语言。通俗点说就是，在 Java 中存储的数据都是有类型的，而且必须在编译时就确定其类型。 Java 中有两类数据类型：



Sring是封装类,不属于基本数据类型

定义字符串: String str = “Hello world”；  
String str = new String("Hello world");

可在过程中定义变量

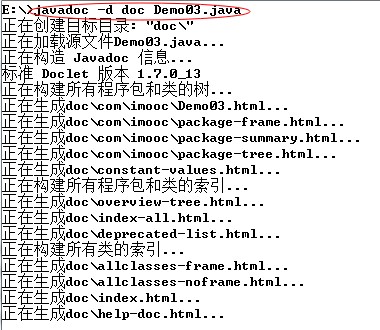
[](http://img.mukewang.com/5359d0c70001ffe202530066.jpg)，

代码中 int 型变量 score1 可以直接为 double 型变量 score2 完成赋值操作，运行结果为：**82.0**

这种转换称为**自动类型转换**

被注释的代码块在程序运行时是不会被执行的~~

我们可以通过 **javadoc** 命令从文档注释中提取内容，生成程序的 API 帮助文档。

[](http://img.mukewang.com/53689f74000187be03800330.jpg)

打开首页，查看下生成的 API 文档

[](http://img.mukewang.com/535ce323000141ff04670397.jpg)

PS：使用文档注释时还可以使用 **javadoc 标记**，生成更详细的文档信息：

       @author 标明开发该类模块的作者

       @version 标明该类模块的版本

       @see 参考转向，也就是相关主题

       @param 对方法中某参数的说明

       @return 对方法返回值的说明

       @exception 对方法可能抛出的异常进行说明

Java 语言中常用的运算符可分为如下几种：

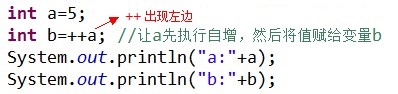
Ø 算术运算符

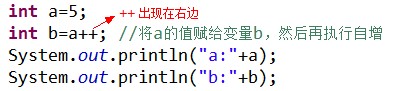
Ø 赋值运算符

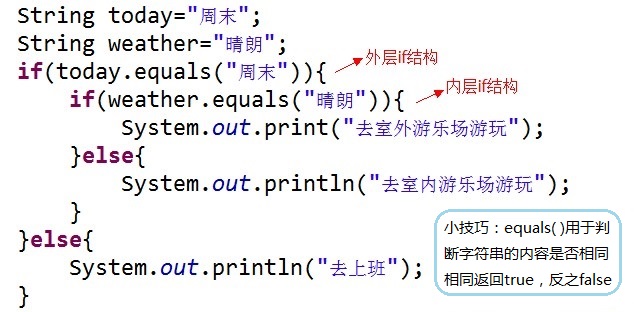
Ø 比较运算符

Ø 逻辑运算符

Ø 条件运算符







System.out.println=System.out.print+System.out.print("/n") 也就是加了一换行

 case 后面的值**可以是常量数值**，如 1、2；也**可以是一个常量表达式**，如 2+2 ；但不能是变量或带有变量的表达式，如 a \* 2

Unix、Linux：\n表示回车+换行，\r没什么意义  
Windows、DOS：\r\n表示回车+换行，顺序不能变  
Mac：\r表示回车+换行



import java.util.Scanner;// 导入包  
Scanner input = new Scanner(System.in);// 定义对象  
int score = input.nextInt(); // 获取用户输入 字符串:next 浮点数:nextDouble 整数:nextInt

**如何使用 Java 中的数组**

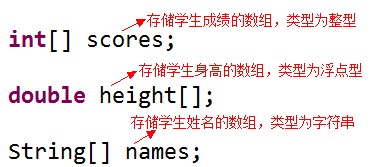
Java 中操作数组只需要四个步骤：

**1、 声明数组**

语法：  **数据类型[ ] 数组名；**

或者   **数据类型 数组名[ ]；**

其中，数组名可以是任意合法的变量名，如：

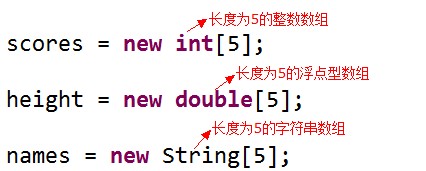
[](http://img.mukewang.com/5371e91d000147e903790167.jpg)

**2、 分配空间**

简单地说，就是指定数组中最多可存储多少个元素

语法：  **数组名 = new  数据类型 [ 数组长度 ];**

其中，数组长度就是数组中能存放元素的个数，如：

[](http://img.mukewang.com/5371ecc10001c97804260171.jpg)

话说，我们也可以将上面的两个步骤合并，在声明数组的同时为它分配空间，如：[http://img.mukewang.com/5371ed4200012bee03590033.jpg](http://img.mukewang.com/5371ed4200012bee03590033.jpg)

**3、 赋值**

分配空间后就可以向数组中放数据了，数组中元素都是通过下标来访问的，例如向 scores 数组中存放学生成绩

[http://img.mukewang.com/5371ed9f0001accb02100053.jpg](http://img.mukewang.com/5371ed9f0001accb02100053.jpg)

**4、 处理数组中数据**

我们可以对赋值后的数组进行操作和处理，如获取并输出数组中元素的值

[http://img.mukewang.com/5371eec50001332707460043.jpg](http://img.mukewang.com/5371eec50001332707460043.jpg)

在 Java 中还提供了另外一种直接创建数组的方式，它将声明数组、分配空间和赋值合并完成，如

[http://img.mukewang.com/5371ee8b000101aa04540056.jpg](http://img.mukewang.com/5371ee8b000101aa04540056.jpg)

它等价于：

[http://img.mukewang.com/5371ef0f0001528c04830059.jpg](http://img.mukewang.com/5371ef0f0001528c04830059.jpg)

**数组名.length** 用于获取数组的长度

字符串名.length( ); 用于获取字符串的长度

语法：  **Arrays.sort(数组名);**

可以使用 sort( ) 方法实现对数组的排序，只要将数组名放在 sort( ) 方法的括号中，就可以完成对该数组的排序（按升序排列）

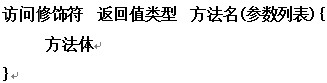
语法：  **Arrays.toString(数组名);**

可以使用 toString( ) 方法将一个数组转换成字符串，该方法按顺序把多个数组元素连接在一起，多个元素之间使用逗号和空格隔开



所谓方法，就是用来解决一类问题的代码的有序组合，是一个功能模块。

一般情况下，定义一个方法的语法是：

[](http://img.mukewang.com/537b5a130001982103270083.jpg)

其中：

1、 访问修饰符：方法允许被访问的权限范围， 可以是 public、protected、private 甚至可以省略 ，其中 public 表示该方法可以被其他任何代码调用，其他几种修饰符的使用在后面章节中会详细讲解滴

2、 返回值类型：方法返回值的类型，如果方法不返回任何值，则返回值类型指定为 void ；如果方法具有返回值，则需要指定返回值的类型，并且在方法体中使用 return 语句返回值

3、 方法名：定义的方法的名字，必须使用合法的标识符

4、 参数列表：传递给方法的参数列表，参数可以有多个，多个参数间以逗号隔开，每个参数由参数类型和参数名组成，以空格隔开

根据方法是否带参、是否带返回值，可将方法分为四类：

Ø 无参无返回值方法

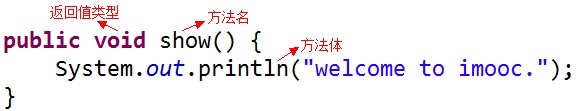
Ø 无参带返回值方法

Ø 带参无返回值方法

Ø 带参带返回值方法

**第一步，定义方法**

例如：下面代码定义了一个方法名为 show ，没有参数，且没有返回值的方法，执行的操作为输出 “ welcome to imooc. ”

[](http://img.mukewang.com/537d655b0001a44305790111.jpg)

注意哦：

1、 方法体放在一对大括号中，实现特定的操作

2、 方法名主要在调用这个方法时使用，需要注意命名的规范，一般采用第一个单词首字母小写，其它单词首字母大写的形式

**第二步，调用方法**

当需要调用方法执行某个操作时，可以先创建类的对象，然后通过  **对象名.方法名();**  来实现（关于类和对象的概念在后面章节中会详细讲解滴，先熟悉语法，表着急哦~~）

例如：在下面的代码中，我们创建了一个名为 hello 的对象，然后通过调用该对象的 show( ) 方法输出信息

[](http://img.mukewang.com/537d65be0001aecb06640362.jpg)

运行结果为：  **welcome to imooc.**

 通过 (int)(Math.random() \* 100) 生成 100 以内的随机数

如果同一个类中包含了两个或两个以上方法名相同、方法参数的个数、顺序或类型不同的方法，则称为方法的重载，也可称该方法被重载了。与返回值无关.

“public class Xxxx”首字母要大写

成员变量和局部变量区别:

1. 初始值不同:Java会给成员变量一个初始值, 引用型默认为null，值型默认为0，布尔型默认为faulse。Java不会给局部变量赋予初始值。
2. 作用域不同
3. 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。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。

对象属性=实例变量 定义在类中但在任何方法之外。（New出来的均有初始化）

Float型变量赋值时需加f,否则默认浮点数是double型

float num2 = 3.5f

构造方法 作为规范 用来初始化

当没有指定构造方法时,系统会自动添加无参构造方法;

当有指定构造方法,无论是有参/无参的构造方法,都不会自动添加无参的构造方法

构造方法的重载:方法名相同,但参数不同的多个方法,调用时会根据不同的参数选择相应的方法

构造方法不但可以给对象的属性赋值,还可以保证给对象的属性赋一个合理的值(在构造方法中用if语句)

静态变量实际上就是全局变量

1. **静态方法中可以直接调用同类中的静态成员，但不能直接调用非静态成员。**

**如果希望在静态方法中调用非静态变量，可以通过创建类的对象，然后通过对象来访问非静态变量**。如：

[](http://img.mukewang.com/5392d7390001a10806150193.jpg)

1. **在普通成员方法中，则可以直接访问同类的非静态变量和静态变量**
2. **静态方法中不能直接调用非静态方法，需要通过对象来访问非静态方法。**

### ****成员变量和静态变量的区别****

      1、两个变量的生命周期不同

            成员变量随着对象的创建而存在，随着对象被回收而释放。

            静态变量随着类的加载而存在，随着类的消失而消失。

      2、调用方式不同

            成员变量只能被对象调用。

            静态变量可以被对象调用，还可以被类名调用。

      3、别名不同

            成员变量也称为实例变量。

            静态变量也称为类变量。

      4、数据存储位置不同

            成员变量存储在堆内存的对象中，所以也叫对象的特有数据。

            静态变量数据存储在方法区（共享数据区）的静态区，所以也叫对象的共享数据。

 先分配静态对象的空间， 继而再对非静态对象分配空间,也就是初始化顺序是先静态再非静态.

 成员变量、局部变量、静态变量的区别

|  |  |  |  |
| --- | --- | --- | --- |
|  | **成员变量** | **局部变量** | **静态变量** |
| 定义位置 | 在类中,方法外 | 方法中,或者方法的形式参数 | 在类中,方法外 |
| 初始化值 | 有默认初始化值 | 无,先定义,赋值后才能使用 | 有默认初始化值 |
| 调用方式 | 对象调用 | --- | 对象调用，类名调用 |
| 存储位置 | 堆中 | 栈中 | 方法区 |
| 生命周期 | 与对象共存亡 | 与方法共存亡 | 与类共存亡 |
| 别名 | 实例变量 | --- | 类变量 |

**封装 变量前加private，通过getter、setter方法调用**

Main函数中 直接使用方法名调用方法只能在一个类中调用，并且该方法必须是静态的(main是静态的,,this是个隐式对象)

在自定义类中不是静态也可以, 直接方法名也能调用，因为默认省略this

用 包名.\* 导入包时,若与本包类名重复,则调用本包的类;

用 包名.具体类名 导入包时,若重复,则调用被导入包的类



内部类的主要作用如下：

1. 内部类提供了更好的封装，可以把内部类隐藏在外部类之内，不允许同一个包中的其他类访问该类

2. 内部类的方法可以直接访问外部类的所有数据，包括私有的数据

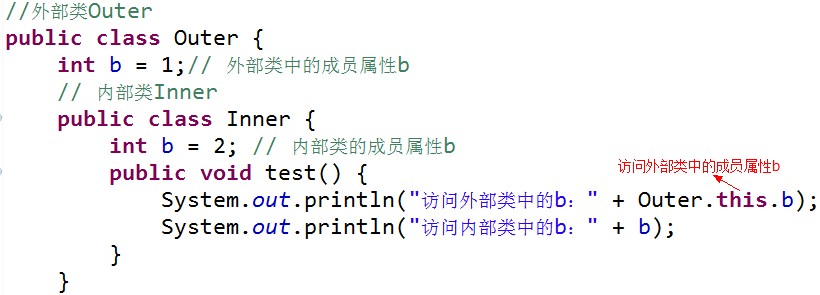
3. 内部类所实现的功能使用外部类同样可以实现，只是有时使用内部类更方便

内部类可分为以下几种：

* 成员内部类 ?
* 静态内部类 ?
* 方法内部类 ?
* 匿名内部类 ?

类可以使用任意访问控制符，如 public 、 protected 、 private 等

定义了成员内部类后，必须使用外部类对象来创建内部类对象，而不能直接去 new 一个内部类对象，即：内部类 对象名 = 外部类对象.new 内部类( ); { Outer outer = new Outer(); Outer.Inner inner = outer.new Inner(); }

[](http://img.mukewang.com/539e638b0001ab1208200295.jpg)

你可以将方法和变量都声明为static。static 成员的最常见的 例子是main( ) 。声明为static的方法有以下几条限制（main也是？？）：  ·

A，它们仅能调用其他的static 方法

B，它们只能访问static数据

C，它们不能以任何方式引用this 或super(this涉及到对象，super 与继承有关）

　Java中的静态变量和静态代码块是在类加载的时候就执行的，实例化对象时，先声明并实例化变量再执行构造函数。如果子类继承父类，则先执行父类的静态变量和静态代码块，再执行子类的静态变量和静态代码块。同样，接着在执行父类和子类非静态代码块和构造函数。

注意：（静态）变量和（静态）代码块的也是有执行顺序的，与代码书写的顺序一致。在（静态）代码块中可以使用（静态）变量，但是被使用的（静态）变量必须在（静态）代码块前面声明。

 最后给出执行步骤：

1、父类静态变量和静态代码块（先声明的先执行）；

2、子类静态变量和静态代码块（先声明的先执行）；

3、父类的变量和代码块（先声明的先执行）；

4、父类的构造函数；

5、子类的变量和代码块（先声明的先执行）；

6、子类的构造函数。

静态内部类是 static 修饰的内部类，这种内部类的特点是：

1、 静态内部类不能直接访问外部类的非静态成员，但可以通过**new 外部类().成员** 的方式访问

2、 如果外部类的静态成员与内部类的成员名称相同，可通过“类名.静态成员”访问外部类的静态成员；如果外部类的静态成员与内部类的成员名称不相同，则可通过“成员名”直接调用外部类的静态成员

3、 创建静态内部类的对象时，不需要外部类的对象，可以直接创建 **内部类 对象名= new 内部类();**

当内部类中定义了[静态成员](https://www.baidu.com/s?wd=%E9%9D%99%E6%80%81%E6%88%90%E5%91%98&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YduARsPWubrjwbuj7brAF-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHD1PjmYrjR1" \t "_blank)，该内部类必须是static的：静态方法随类的加载而加载，创建类的同时静态方法加载进内存，这样在类未实例化的情况下通过类名调用静态方法。  
内部类的方法若有静态方法，而内部类为非静态，必须实例化后才能加载进内存，内部类的静态方法不能通过外部类名.内部类名.静态方法调用，因为在内存中找不到内部类。  
故内部类中有静态方法时，必须同时声明为静态，让内部类随外部类的创建一起加载，内部类中的静态方法才能正常调用。

方法内部类就是内部类定义在外部类的方法中，方法内部类只在该方法的内部可见，即只在该方法内可以使用。

由于方法内部类不能在外部类的方法以外的地方使用，因此方法内部类不能使用访问控制符和 static 修饰符。

Java中的继承是单继承

继承的好处:子类拥有父类的所有属性和方法(private修饰的无效)

语法规则: class 子类 extends 父类

例:class Dog extends animal{…}

private属性不能继承

方法的重写

1. 如果子类对继承父类的方法不满意,是可以重写父类继承的方法的,当调用方法时会优先调用子类的方法.
2. 语法规则
3. 返回值类型
4. 方法名
5. 参数类型及个数

都要与父类继承的方法相同,才叫方法的重写

继承的初始化顺序

1. 初始化父类再初始子类
2. 先执行初始化对象中属性,再执行构造方法中的初始化

在创建子类对象时候，如果子类的构造函数没有显示调用父类的构造函数且父类只定义了自己的有参构造函数，则会出错（如果父类只有有参数的构造方法，则子类必须显示调用此带参构造方法）。

final关键字

使用final关键字做标识有 最终的 含义

final可以修饰类、方法、属性和变量

final修饰类，则该类不允许被继承

final修饰方法，则该方法不允许被覆盖（重写）

final修饰属性 则该类的属性不会进行隐式的初始化（类的初始化属性必须有值）或在构造方法中赋值（但只能选其一）

如果用final修饰属性则该属性不能在其他地方（包括构造方法和main函数）修改值，也就是说属性的值只能赋一次

继承

super关键字：

在对象的内部使用，可以代表父类对象。

1. 访问父类的属性

super.age

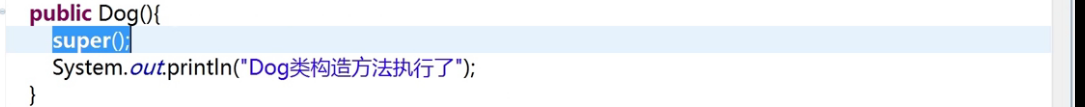
1. 访问父类的方法

super.eat().

super的应用

如果子类的构造方法中没有显示调用父类的构造方法，则系统系默认调用父类无参的构造方法

如果子类构造方法中既没有显式调用父类的构造方法，而父类又没有无参的构造方法，则编译出错。

如果显示的调用构造方法，必须在子类的构造方法的第一行（super（）；可加可不加，加时必须在第一行）

子类的构造的过程中必须调用其父类的构造方法。

如果子类的构造方法中没有显示调用父类的构造方法，则系统默认调用父类无参的构造方法。

Object类

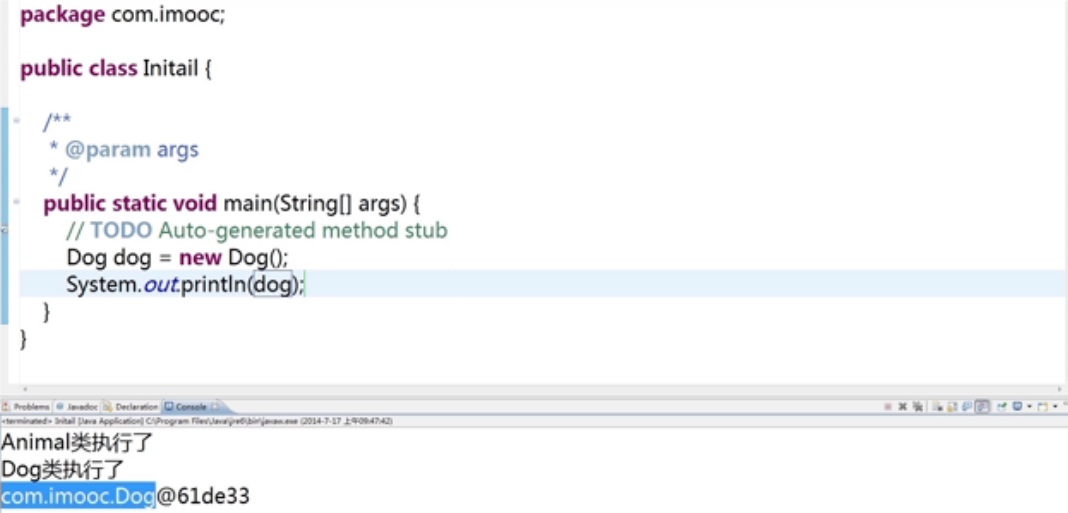
Object类是所有类的父类，如果一个类没有使用extends关键字明确标识继承另外一个类，那么这个类默认继承Object类。

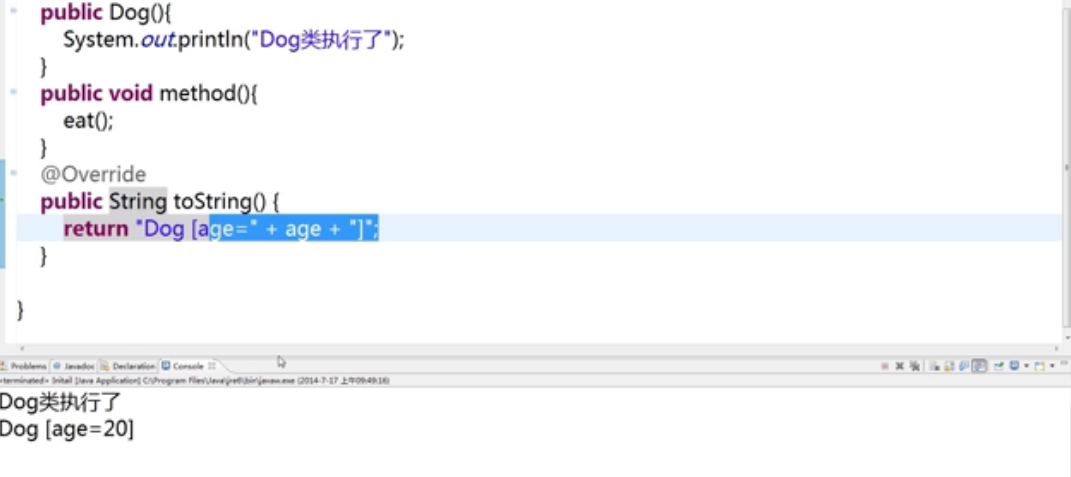
Object类中的方法，适合所有子类。

1. toSring（）方法

在Object类里面定义toString（）方法的时候返回的对象的哈希code码（对象地址字符串）

哈希码用来区分对象

可以通过重写toString（）方法表示出对象的属性



1. equals（方法）

比较的是对象的引用是否指向同一块内存地址

返回值是布尔类型

Dog dog = new Dog（）；

Dog dog 2 = new Dog（）；

if（dog.equals（dog2））{ }。。。//（两个对象是不相同的，返回值0）

if（dog == dog2）{ }。。。 //（两个对象是不相同的，返回值0）

dog并不是创建的对象，它仅仅是对象在内存中的地址。在操作地址时相当于间接操作了对象的值。

类似于家里的遥控器。当你按遥控的时候，电视会切换节目，其实遥控器并不能代表对象的本身，你只是通过遥控器间接地操作我们的电视。

准确的说dog只是一个引用，引用了内存中的一块地址，所以说equals方法默认只是比较了对象的引用是否是相同的，也就是说是否指向同一块内存区域。

重写equals方法（菜单 -> Sourse -> hashcode()…）比较对象中每个属性的值



第一个if:比较地址是否相同

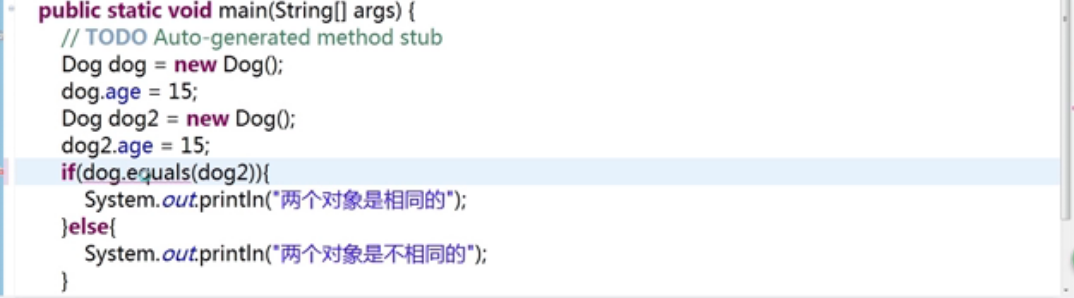
第二个if: 如果第二个对象是空值就不比较

第三个if: 得到类对象(使用new时得到的是类的对象,使用getClass方法得到的是类对象.类的对象描述的是类的代码信息,而类对象关注的是数据信息),这句话用来判断类的类型是否相等

第三.四个if之间： 把传进来的对象转为Dog类型的对象

第四个if： 判断两个对象的属性值是否相同

然后重新调用equals方法即可



输出两个对象是相同的

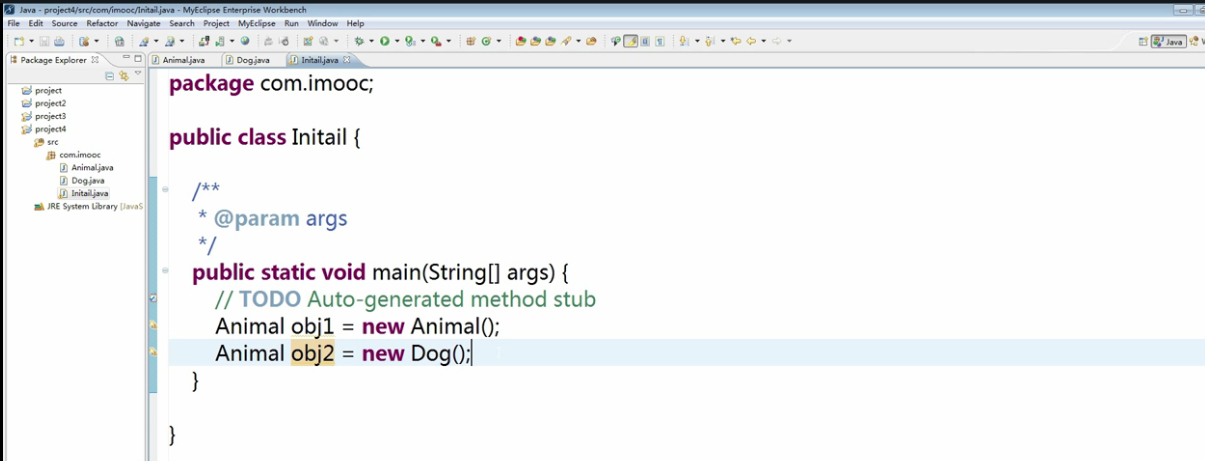
多态(对象的多种形态)

1. 引用多态

父类的引用可以指向本类的对象

父类的引用可以指向子类的对象

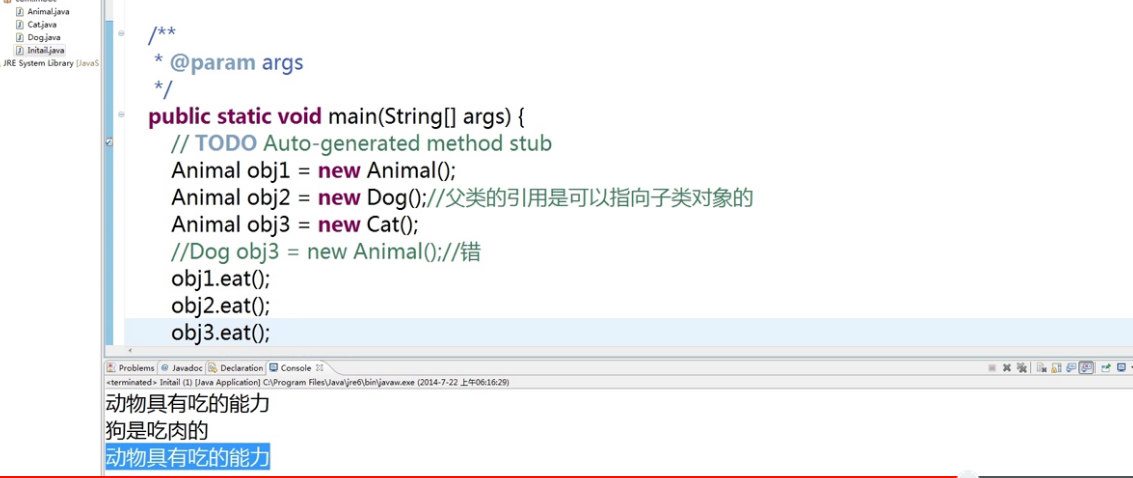
子类的引用**不**可以指向父类的对象



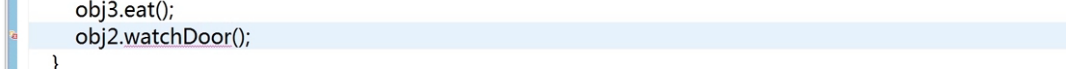
1. 方法多态

创建本类对象时,调用的方法为本类方法

创建子类对象时,调用的方法为子类重写的方法或者继承的方法



父类的引用可以指向子类对象,但是不能调用子类特有的方法



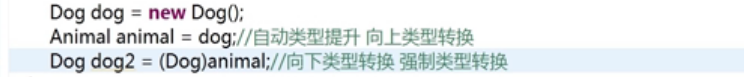
**java.lang.Float.floatToIntBits()**方法返回根据IEEE754浮点“单一格式”位布局，返回指定浮点值的表示。它包括以下要点：

* 如果参数为负无穷小，其结果是 0x7f800000.
* 如果参数为负无穷大，其结果是 0xff800000.
* 如果参数为NaN，那么结果是 0x7fc0000

引用类型转换

1. 向上类型转换(隐式/自动类型转换),是小类型到大类型的转换 无风险
2. 向下类型转换(强制类型转换),是大类型到小类型 有风险
3. instanceof 运算符,来解决引用对象的类型,避免类型转换的安全性问题

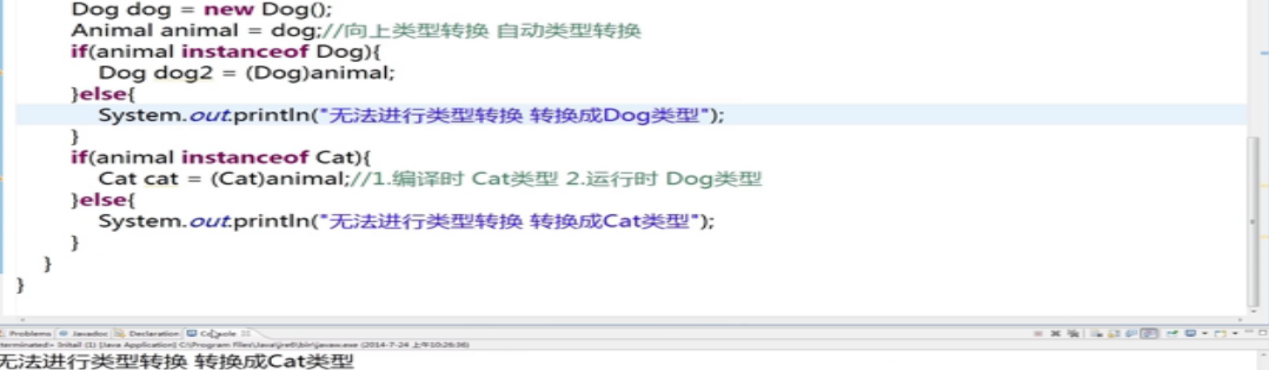






上图第四行编译不出错,运行出错

instanceof是[Java](http://baike.baidu.com/subview/29/12654100.htm" \t "_blank)、[php](http://baike.baidu.com/subview/99/5828265.htm" \t "_blank)的一个二元操作符（运算符），和==，>，<是同一类东西。由于它是由字母组成的，所以也是Java的保留[关键字](http://baike.baidu.com/view/390935.htm" \t "_blank)。它的作用是判断其左边对象是否为其右边类的实例，返回boolean类型的数据。可以用来判断[继承](http://baike.baidu.com/view/125322.htm" \t "_blank)中的子类的实例是否为父类的实现



抽象类

1. 语法定义:

抽象类前使用abstract关键字修饰,则该类为抽象类

1. 应用场景:

a.在某些情况下,某个父类只知道其子类应该包含怎样的方法,但无法准确知道这些子类如何实现这些方法.凡是实现类必须要完整的重写所有的方法。（实现类！）部分不重写的话子类只能也是抽象类直至所有[抽象方法](https://www.baidu.com/s?wd=%E6%8A%BD%E8%B1%A1%E6%96%B9%E6%B3%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvnAc4njbzuHfzujw-uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1m4PjfvrH0Lrj04P1nLnW0Y" \t "_blank)被全部重写为止。

b.从多个具有相同特征的类中抽象出一个抽象类,以这个抽象类作为子类的模板,从而避免了子类设计的随意性

1. 作用:

限制规定子类必须实现某些方法,但不关注实现细节

4. 使用规则

a. abstract定义抽象类

b. abstract定义抽象方法，只有声明，不需要实现

c. 包含抽象方法的类就是抽象类

d. 抽象类中可以包含普通的方法，也可以没有抽象方法

e. 抽象类不能直接创建，可以定义引用变量

接口（Interface）

1. 接口概念

接口可以理解为一种特殊的类,由全局常量和公共的抽象方法所组成.

类是一种具体实现体,而接口定义了某一批类所需要遵守的规范,接口不关心这些类的内部数据,也不关心这些类里方法的实现细节,它只规定这些类里必须提供某些方法

1. 接口定义

和类定义不同,定义接口不再使用class关键字,而是使用interface关键字

接口定义的基本语法:

[修饰符] interface 接口名 [extends 父接口1, 父接口2 …]

{

零个到多个常量定义…

零个到多个抽象方法的定义…

}

接口就是用来被继承、被实现的，修饰符一般建议用public

注意不能使用private和protected修饰接口

1. 接口定义

常量：

接口中的属性是常量，即使定义时不添加public static final 修饰符，系统也会自动加上。

方法：

接口中的方法只能是抽象方法，总是使用，即使定义时不添加 public abstract 修饰符，系统也会自动加上

1. 使用接口

一个类可以实现一个或多个接口，实现接口使用implements关键字。Java中一个类只能继承一个父类，是不够灵活的，通过实现多个接口可以做补充。

继承父类实现接口的语法为：

[修饰符] class 类名 extends 父类 implements 接口1，接口2 …

{

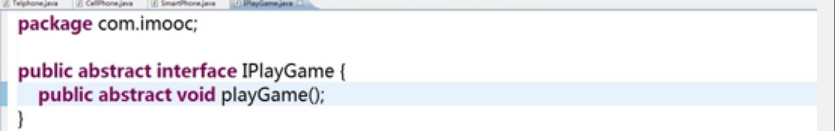
类体部分//如果继承了抽象类，需要实现继承的抽象方法；要实现接口中的抽象方法。

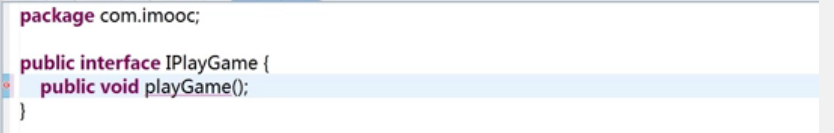
}

如果要继承父类，继承父类必须在实现接口之前

命名时在名字前加大写的 I ，与类文件区分开

声明接口时其实省略了abstract关键字，可写可不写，一般不写





接口在使用过程中，还经常与匿名内部类配合使用

匿名内部类就是没有名字的内部类，多用于关注实现而不关注实现类的名称

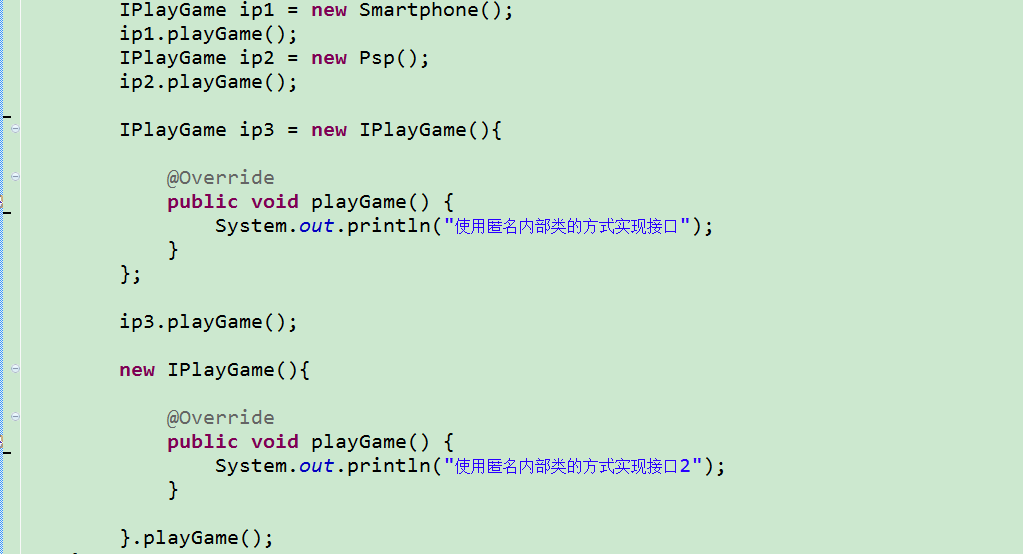
语法格式：

Interface I = new Interface(){

public void method(){

System.out.println(“匿名内部类实现接口的方式”);

}

}

**1、**使用匿名内部类时，我们必须是继承一个类或者实现一个接口，但是两者不可兼得，同时也只能继承一个类或者实现一个接口。

**2、**匿名内部类中是不能定义构造函数的。(构造函数需要与类名相同,而匿名内部类无类名)

**3、**匿名内部类中不能存在任何的静态成员变量和静态方法。

**4、**匿名内部类为局部内部类，所以局部内部类的所有限制同样对匿名内部类生效。

**5、**匿名内部类不能是抽象的，它必须要实现继承的类或者实现的接口的所有抽象方法。

UML

1. UML概念

Unified Modeling Language (UML)

又称统一建模语言或标准建模语言

是一个支持模型化和软件系统开发的图形化语言

为软件开发的所有阶段提供模型化和可视化支持

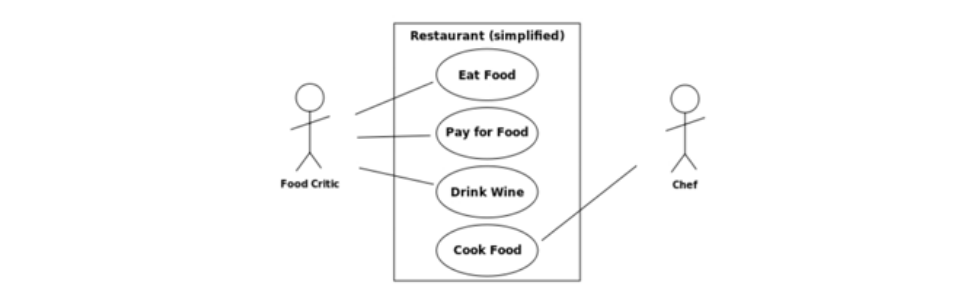
1. UML图示

UML 2.2中一共定义了14种图示（diagrams）。

1. 常用UML图

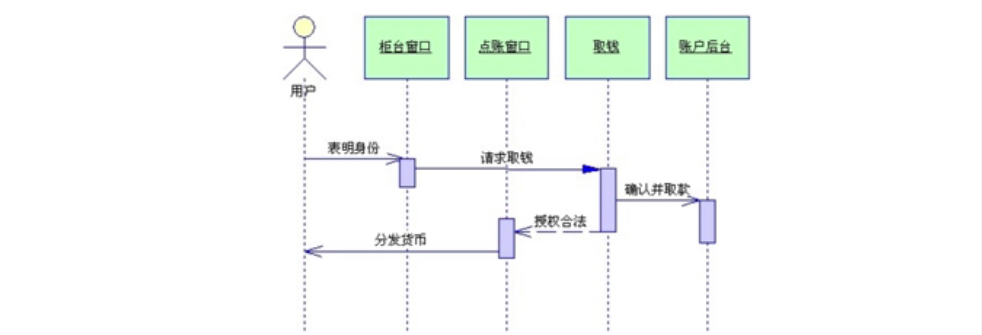
用例图（The Use Case Diagram）

用例图能够以可视化的方式，表达系统如何满足所收集的业务规则，以及特定的用户需求等信息。



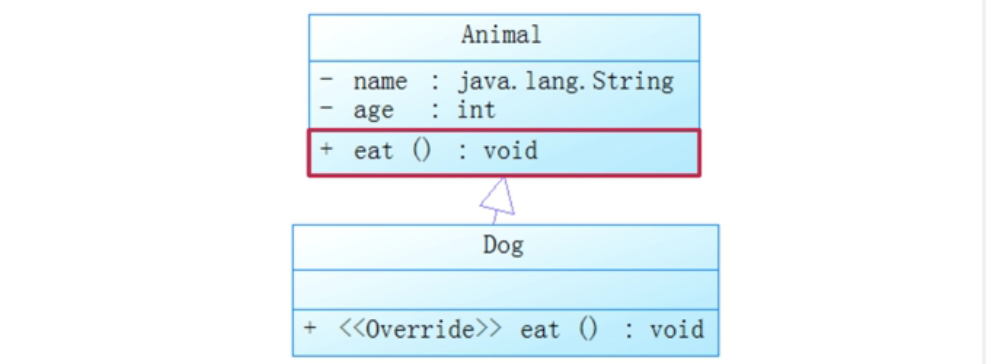
序列图（The Sequence Diagram）

序列图用于按照交互发生的一系列顺序，显示对象之间的这些交互。



类图（The Class Diagram）

UML类图、业务逻辑和所有支持结构一同被用于定义全部的代码结构



1. UML建模工具

Visio、Rational Rose、PowerDesign 三种建模工具应用最广

* 1. Rational Rose，它是IBM的。

http://www.uml.org.cn/RequirementProject/200604043.htm

* 1. Microsoft 的Microsoft Office Visio

<http://www.microsoft.com/china/office/xp/visio/default.asp>

* 1. PowerDesigner

http://www.uml.org.cn/UMLTools/powerDesigner/powerDesignerToolIntroduction.htm