# 第九章 对象和类

## 9.1 引言

1.要点提示：面向对象编程可以有效地帮助开发大规模的软件以及图形用户界面。

## 9.2为对象定义类

1.要点提示：类为对象定义属性和行为。

2.对象：面向对象程序设计（OOP）就是使用对象进行程序设计。对象代表现实世界中可以明确标识的一个实体，例如：一个学生、一张桌子等

3.每一个对象都有自己独特的标志和行为：一个对象的状态（或属性）是由具有当前值的数据域来表示的；一个对象的行为（或动作）是由方法定义的。

4.类和对象的关系：一个对象是类的一个实例。可以从一个对象中创建多个实例（创建示例的过程称为“实例化”）。

5.Java使用变量定义数据域，使用方法定义动作。

## 9.3示例：定义类和创建对象

1.类是对象的定义，对象从类创建。

2.下面是一个求圆面积的练习，主要是用来熟悉类和对象。



3.可以把两个类放在同一个文件中，但是文件中只允许有一个公共类（与文件名重名）。源码中每个类编译成.clss文件。如果有两个类，则这一个文件编译形成两个.class文件。

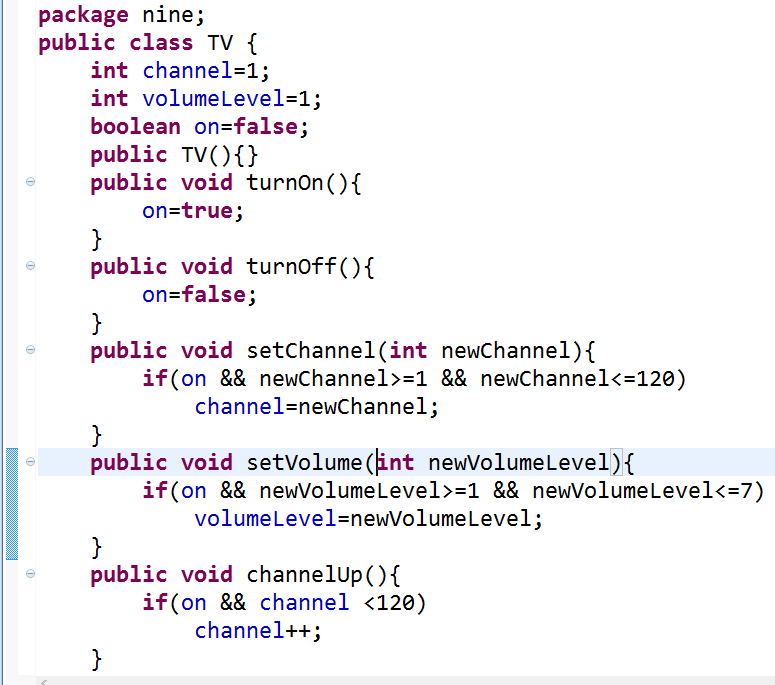
4.将上面的例子中的两个类组合到一起：

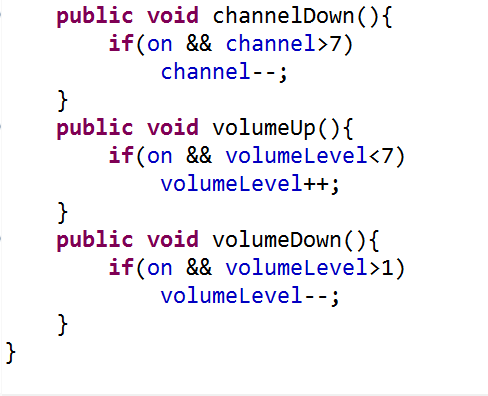


实际就是“把类名改了（自己调用自己）+把子类去掉了，保留子类中的方法”

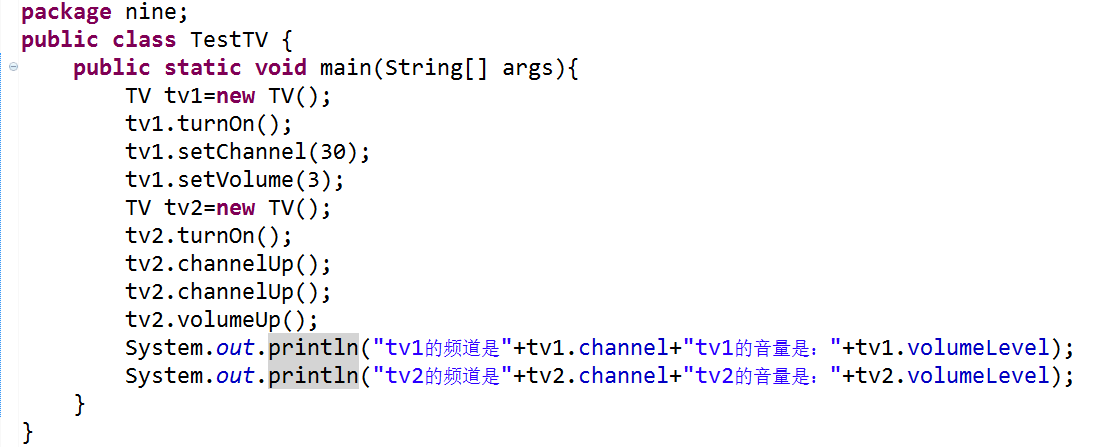
5.TV的Java程序，其中第一个声明“频道增减和音量增减的方法”，另一个有主函数，用于具体的实现。

方法类：





主函数类：



其实，主函数实现了频道和音量简单的增加。

## 9.4使用构造方法构造对象

1.要点提示：构造方法在使用new操作符创建对象的时候调用。

2.构造方法是一种特殊的方法。他们有以下三个特性：

①构造方法必须具备和所在类相同的名字。

②构造方法没有返回值类型，甚至连void 也没有。

③构造方法是在创建一个对象使用new操作符时调用的。构造方法的作用是初始化对象。

3.构造方法具有和定义它的类完全相同的名字。和所有其他方法一样，构造方法也可重载（也就是说，可以有多个重名的构造方法，但它们要有不同的签名），这样更易于用不同的初始数据值来构造对象。

4.一个常见的错误就是将关键字void 放在构造方法前面。例如：

Public void Circle（）{}这种情况下，Circle（）是一个方法，而不是构造方法。

5.构造方法是用来构造对象的。为了能够从一个类构造对象，使用new操作符调用这个类的构造方法，如下所示：

New ClassName（arguments）;

6.通常，一个类会提供一个没有参数的构造方法（例如：Circle（））。这样的构造方法称为“无参构造方法”。

7.默认构造方法：一个类可以不定义构造方法。在这种情况下，类中隐含一个方法体为空的无参构造方法，这个方法称为“默认构造方法”，当且仅当类中没有明确定义任何构造方法时才会自动提供它。

## 9.5通过引用变量访问对象

要点提示：对象的数据和方法可以运用点操作符（.）通过对象的引用变量进行访问。

### 9.5.1引用变量和引用类型

1.下面的语句声明变量myCircle的类型是Circle类型：Circle myCircle;

2.下面的语句创建一个对象，并且将它的引用赋给变量myCircle:

myCircle=new Circle( );

3.采用如下所示的语法，可以写一条包括声明对象引用变量、创建对象以及将对象的引用赋值给这个变量的语句。

Circle myCircle=new Circle( );

4.可以简单地说myCircle是一个Circle对象。

5.注意：在Java中，数组被看作是对象。数组是用new操作符创建的。

### 9.5.2 访问对象的数据和方法

1.在面向对象的编程中，对象成员可以引用该对象的数据域和方法。在创建一个对象后，它的数据和方法可以采用点操作符（.）来访问和调用，该操作符也成为“数据成员访问操作符”。

2.对象成员访问操作：

（1）objectRefVar.dataField 引用对象的数据域

（2）objectRefVar.method(arguments) 调用对象的方法

例如：myCircle.radius 引用myCircle的半径；

myCircle.getArea( )调用myCircle 的getArea方法。

3.数据域radius被称作“实例变量”；getArea被称作“实例方法”。

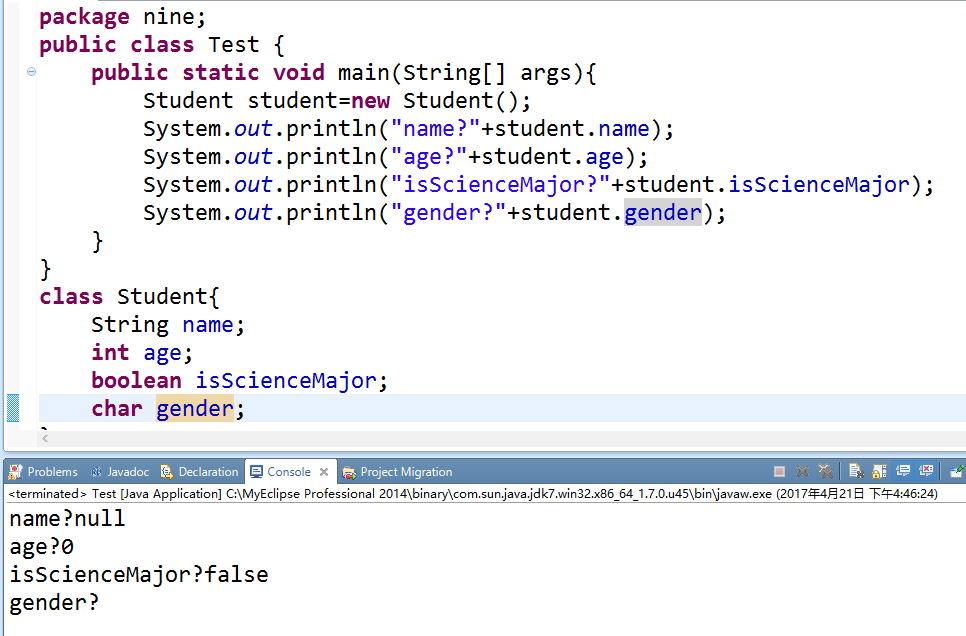
4.类不可以直接调用方法，但是对象可以直接调用方法。

### 9.5.3引用数据域和null值

1.如果一个引用类型的数据域没有引用任何对象，那么这个数据域就有一个特殊的Java值null。

2.引用类型数据域的默认值是null；数值类型的数据域默认值是0；boolean类型数据域的默认值是false；而char类型的数据域的默认值是“\u0000”。

3.但是Java没有给方法中的局部变量赋默认值。



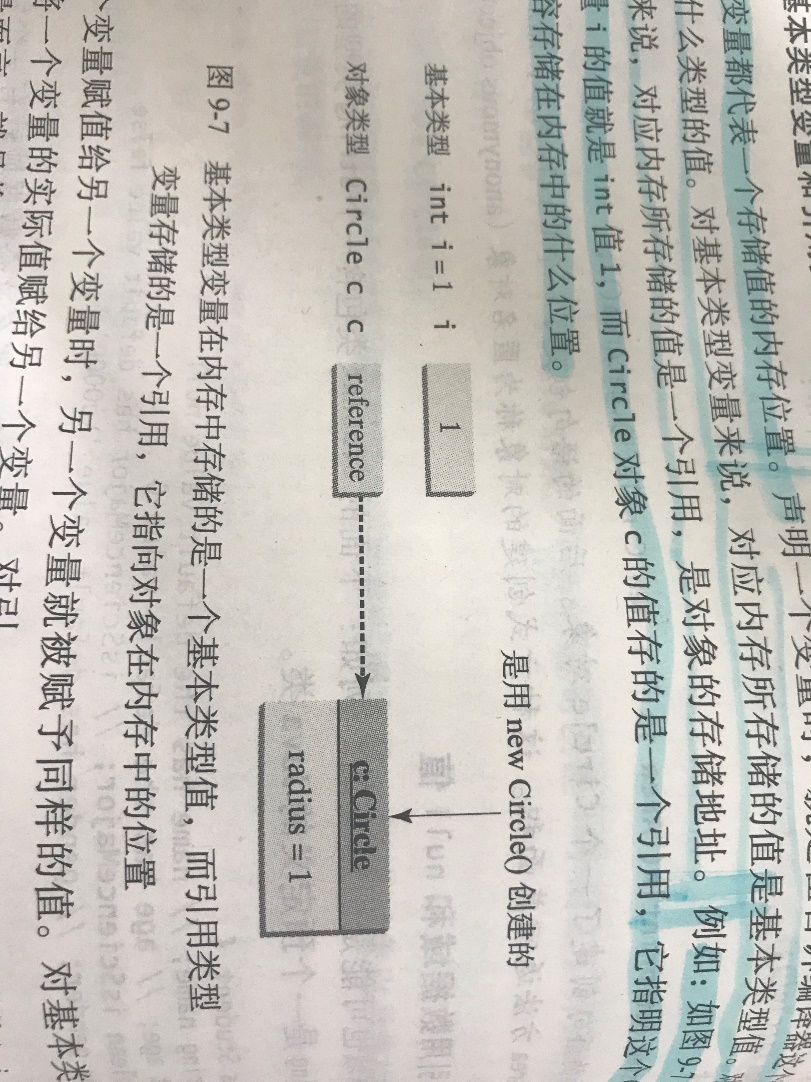
4.NullPointerException是一种常见的运行时错误，当调用值为null 的引用变量上的方法时会发生此类异常。

### 9.5.4基本类型变量和引用类型变量的区别

1.每个变量变量都代表一个存储值的内存位置。声明一个变量时就是告诉编译器这个变量可以存放什么类型的值。

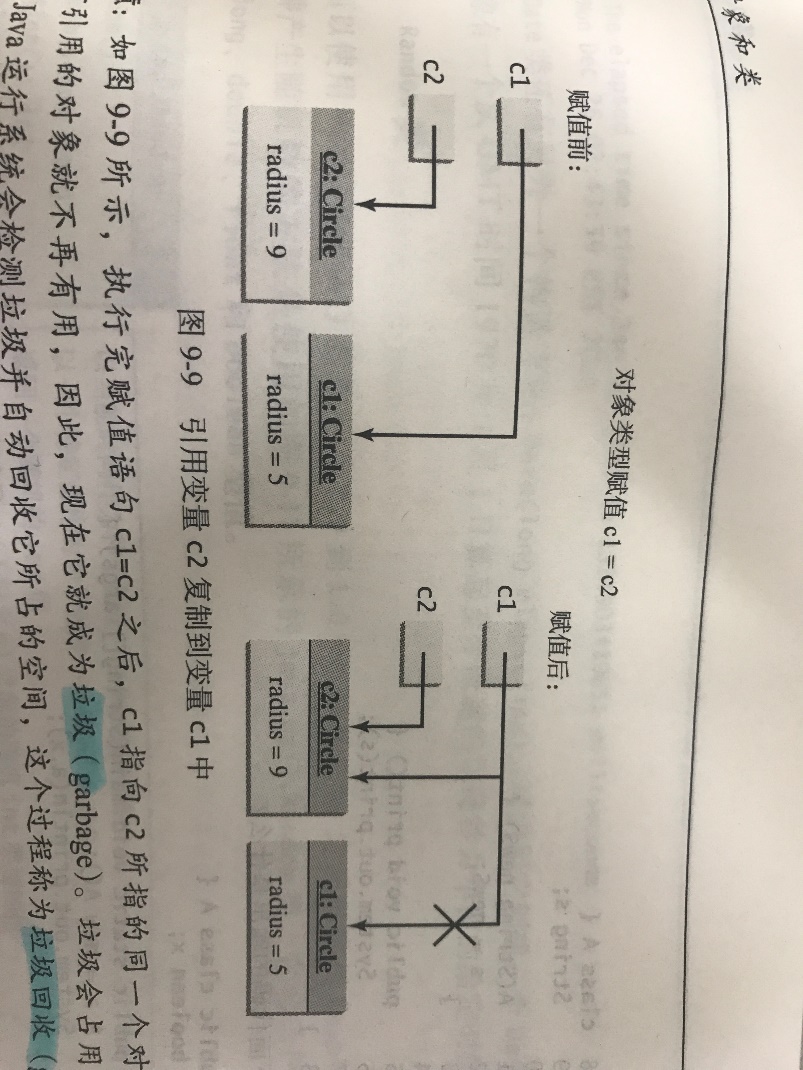
2.对于基本类型的变量来说，对应内存所存储的值是基本类型值；对于引用类型的变量来说，对应内存所存储的值是一个引用，是对象的存储地址。

例如：如图所示，int 型变量i的值就是int 值1；而Circle 对象c的值存的是一个引用，它指明这个Circle 对象的内容存储在内存中的什么位置。



3.将一个变量赋值给另一个变量时，另一个变量就被赋予同样的值。对于基本类型的变量而言，就是将一个变量的实际值赋给另一个变量；对于引用类型的变量而言，就是将一个变量的引用赋值给另一个变量。

4.如图所示，



执行完c1=c2之后，c1指向c2所指向的同一个对象，c1以前引用的对象就不再有用，因此它现在就变成垃圾。垃圾会占用内存空间。Java运行系统会检测垃圾并自动回收它所占的空间，这个过程称为“垃圾回收”。

5.如果你认为不再需要某个对象时，可以显示的给该对象的引用变量赋值null。如果该对象没有被任何引用变量所引用，Java虚拟机将自动回收它所占的空间。

## 9.6使用Java库中的类

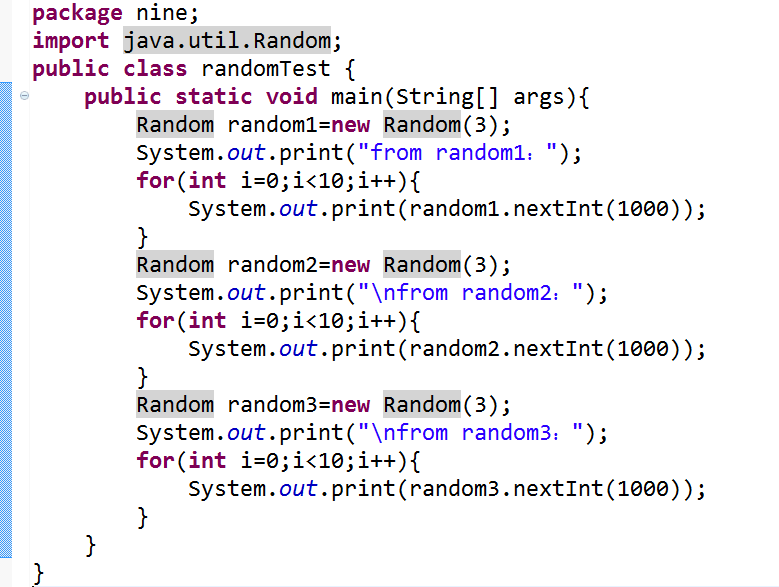
### 9.6.1 Date类

1.java 在java.util.Date类中还提供了与系统无关的对日期和时间的封装。

### 9.6.2 Random类

1.可以使用Math.random( )获取 一个0.0到1.0（不包括1.0）之间的随机double型值；另一种产生随机数的方法是用 java.util.Random类。

2.创建一个Random对象时，必须指定一个种子或者使用默认的种子。种子是一个用于初始化一个随机数字生成器的数字。无参构造方法使用当前已经逝去的时间作种子，创建一个Random对象。如果这两个Random对象有相同的种子，那么它们将产生相同的数列。例如：下面的代码都用相同的种子3来产生两个Random对象。

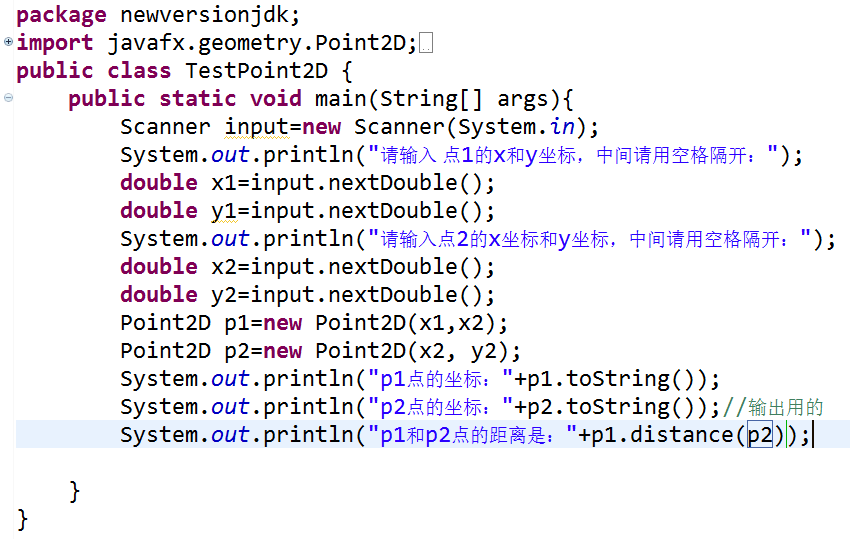


注意：产生相同随机值数列的能力在软件测试以及其它许多应用中是很有用的。在软件测试中，经常需要从一组固定顺序的随机数中来重复产生测试案例。

### 9.6.3 Point2D类

1.Point2D类用于表示二维平面上的点。

2.下面用给定x和y坐标的点来创建一个Point2D对象，使用distance方法计算该点到另一个点之间的距离，并且用toString( )方法来返回该点的字符串表示。



主要涉及jdk版本的问题，jdk1.7中不支持新的javafx，安装jdk1.8，需要在javahome中添加路径。并且在新建项目的时候选择jdk的版本。另外，注意查询Java文档和谷歌学习解决问题。

## 9.7静态变量、常量和方法

1.要点提示：静态变量被类中所有的对象所共享。静态方法不能访问类中的实例成员。

2.实例变量：Circle类的数据域radius称为实例变量。实例变量是绑定到类的某个特定实例的，它是不能被同一个类的不同对象所共享的。例如，假设创建了如下两个对象

Circle circle1=new Circle（）；

Circle circle2=new Circle( )；

Circle1 和circle2中的radius是不相关的，它们存储在不同的内存位置。Circle1中的radius变化不会影响到circle中的，反之亦然。

3.静态变量：如果想要一个类的所有实例共享数据，就要使用静态变量。静态变量将变量值存储在一个公共的内存地址。因为它是公共的地址，所以如果某一个对象修改了静态变量的值，那么同一个类的所有对象都会受到影响。

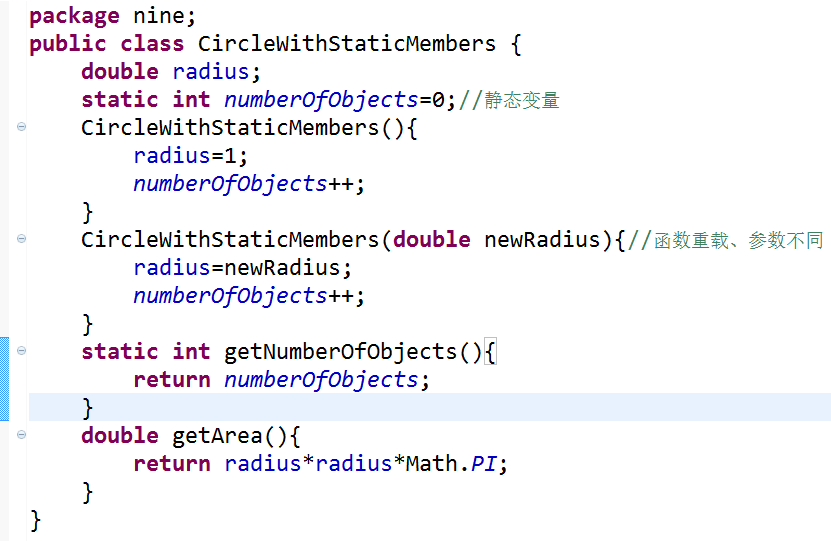
4.要声明一个静态变量或者定义一个静态方法，就要在这个变量或方法的声明中加上修饰符static 。

例如：static int numberOfObject ;

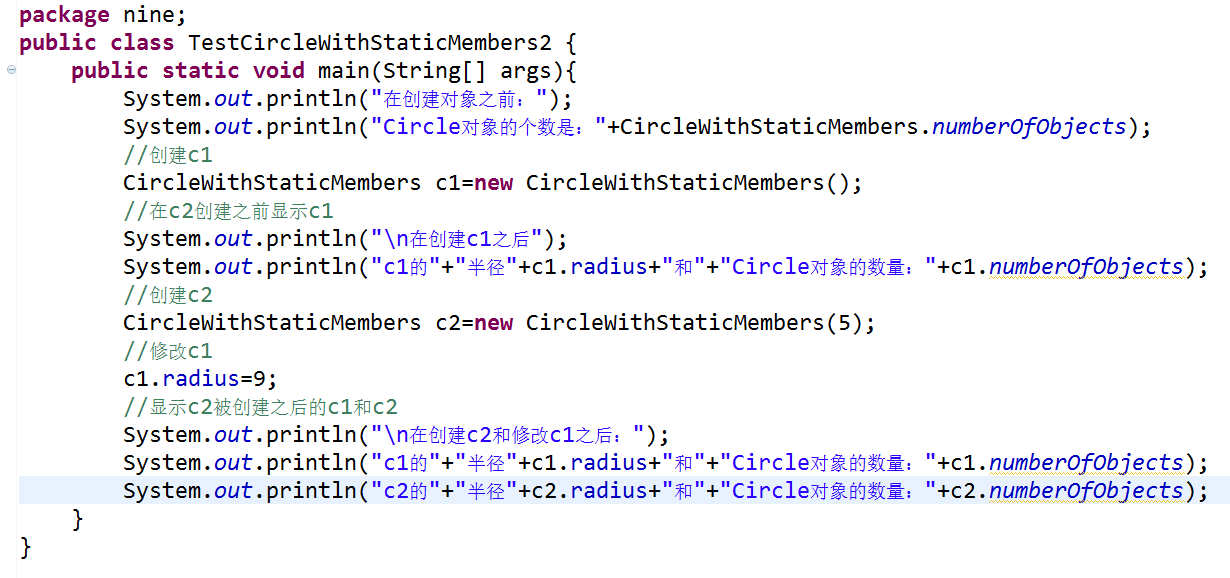
5.类中的常量是被该类的所有对象所共享。因此，常量应该声明为 final static ，例如，Math中常用的常量PI是这样定义的：

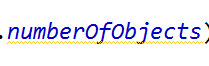
Final static double PI =3.1254455；

6.声明程序示例：



7.如下程序演示了如何使用实例变量、静态变量、实例方法和静态方法，还演示了使用它们的效果。



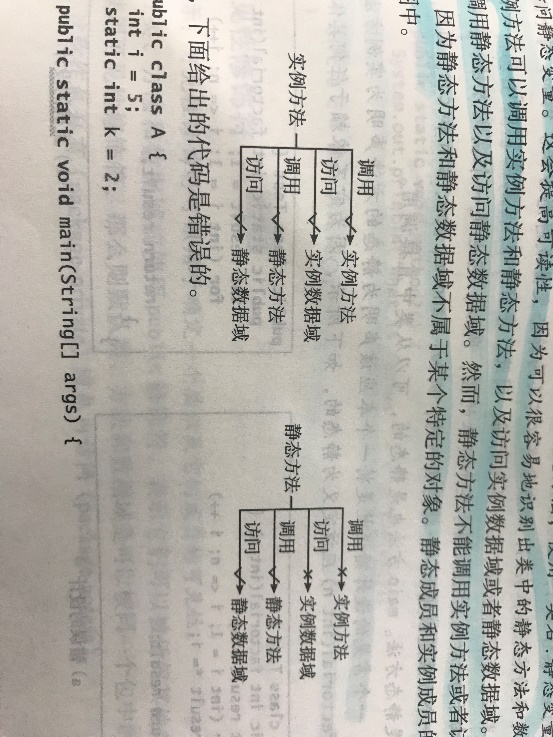
这个方法能够显示对象的个数，这个方法还是比较有趣的。

8.PI是一个定义在Math中的常量，可以使用Math.PI来访问这个常量。最好使用CircleWithStaticMembers.numberOfObjects来代替c1.numberOfObjects。这样可以提高可读性，因为其他程序员很容易识别静态变量。

9.使用“类名.方法名（参数）”的方式调用静态方法，使用“类名.静态变量”的方式访问静态变量。这样会提高可读性，因为这样可以很容易的识别出类中的静态方法和数据。

10.实例化方法可以调用实例方法和静态方法，以及访问实例数据或者静态数据。

静态方法可以调用静态方法以及访问静态数据域。然而静态方法不能调用实例方法或者访问实例数据域。因为静态方法和静态数据域不属于某个特定的对象。静态成员和实例成员的关系总结在下图中：



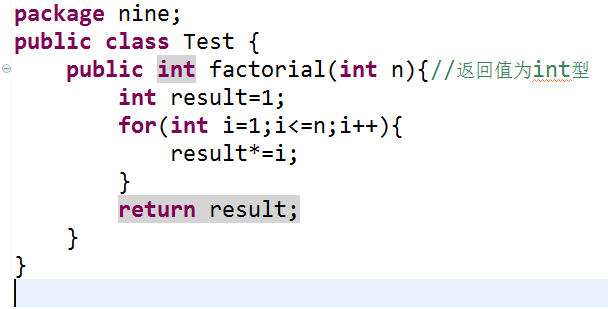
11.如何判断一个变量或方法应该是实例的还是静态的？

如果一个变量或方法依赖于类的某个具体实例，那就应该将它定义为实例变量或实例方法；如果一个变量或者方法不依赖于类的某个具体的实例，那就应该将它定义为静态变量或静态方法。

例如：每一个圆都有自己的半径，半径都依赖于某个具体的圆。因此，半径radius就是Circle类的一个实例变量。由于getArea方法依赖于具体的圆，所以它也是一个实例方法。

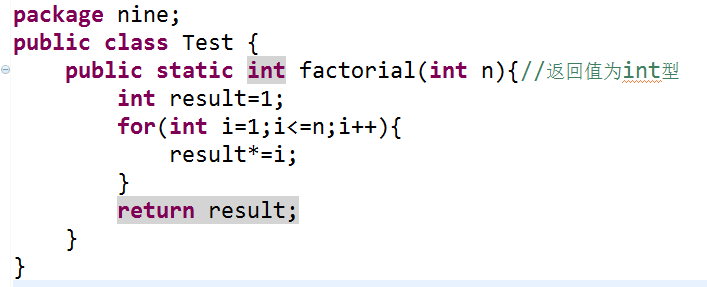
在Math类中没有一个方法是依赖于一个特定的实例的，例如：random、pow、sin、cos。因此这些方法都是静态方法。此外，main方法也是静态的。

12.警告：一个常见的设计错误就是将一个本应该声明为静态的方法声明为实例方法。例如：方法factorial(int n)应该定义为静态的，如下所示，因为它不依赖于任何具体的实例。



错误的设计。（虽然不报错、无警告，但是设计是错误的）

下面是正确的设计：



## 9.8可见性修饰符

1.要点提示：可见性修饰符可以用于确定一个类以及它的成员的可见性。

2.可以在类、方法和数据域前使用public 修饰符，表示它们可以被任何其他的类访问。

3.如果没有使用可见性修饰符，那么则默认类、方法和数据域是可以被同一个包中的任何一个类访问的。这称作“包私有”或包内访问。

4.包可以用来组织类。如果定义类时没有声明包，就表示把它放在默认包里。Java建议最好将类放入包中，而不要使用默认包。

5.private 修饰符限定方法和数据域只能在它自己的类中被访问。

6.私有的修饰符将访问权限限定在自己的类内，默认修饰符将访问权限限定在包内，而公共的修饰符可以无限制地访问。

7.修饰符private只能用在类的成员上。修饰符public可以用在类或类的成员上。在局部变量上使用修饰符public 和 private 都会 导致编译错误。

8.大多数情况下，构造方法应该是公共的。

## 9.9数据域封装

1.要点提示：将数据域私有保护数据，并且使类易于维护。

2.上节的程序中，数据域radius和numberOfObjects可以直接修改，这不是一个好的做法，原因有以下两点：

第一，首先，数据可能被篡改；

第二，其次，它使类变得难以维护，同时容易出现错误。例如，程序员想确保半径不为负数，他不仅要修改本类中的变量，还要修改用到这个变量的其他程序。

（？）

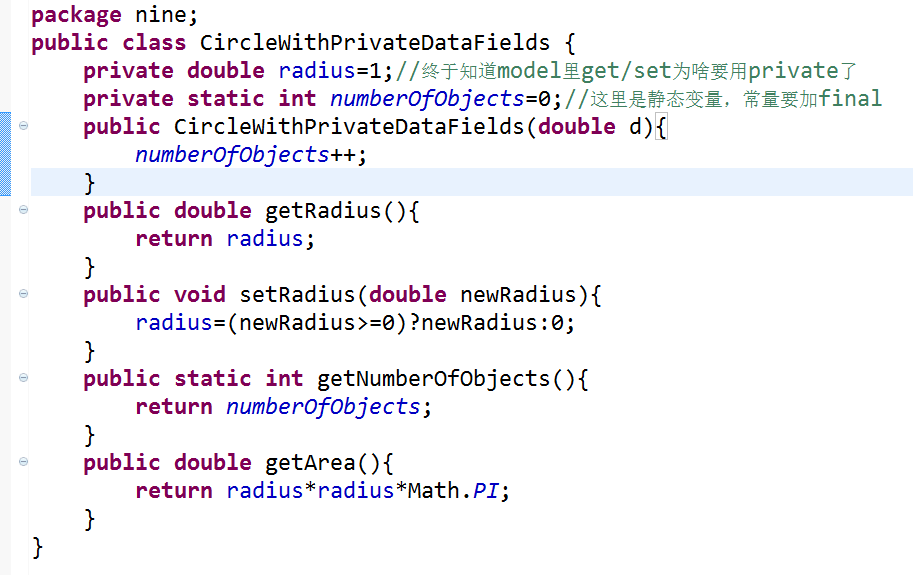
3.数据域封装：为了避免对数据域的直接修改，应该使用private修饰符将数据域声明为私有的，这称为“数据域封装”。

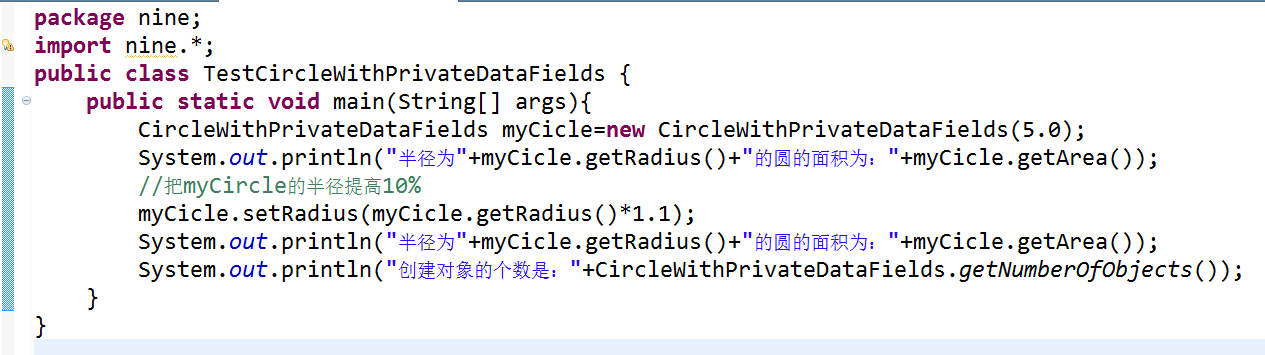
**4.get/set方法的意义**

私有数据域的类外对象是不能访问这个数据域的。但是经常会有客户端需要存取、或者修改数据域的情况。为了能够访问私有数据域，可以提供一个get方法返回数据域的值，可以提供一个set方法给数据域设置新的值。Get方法也被称为访问器，而set方法被称为修改器。

**可以简单的理解为：类内的私有对象对外提供的访问接口。**

以下是相关实例：





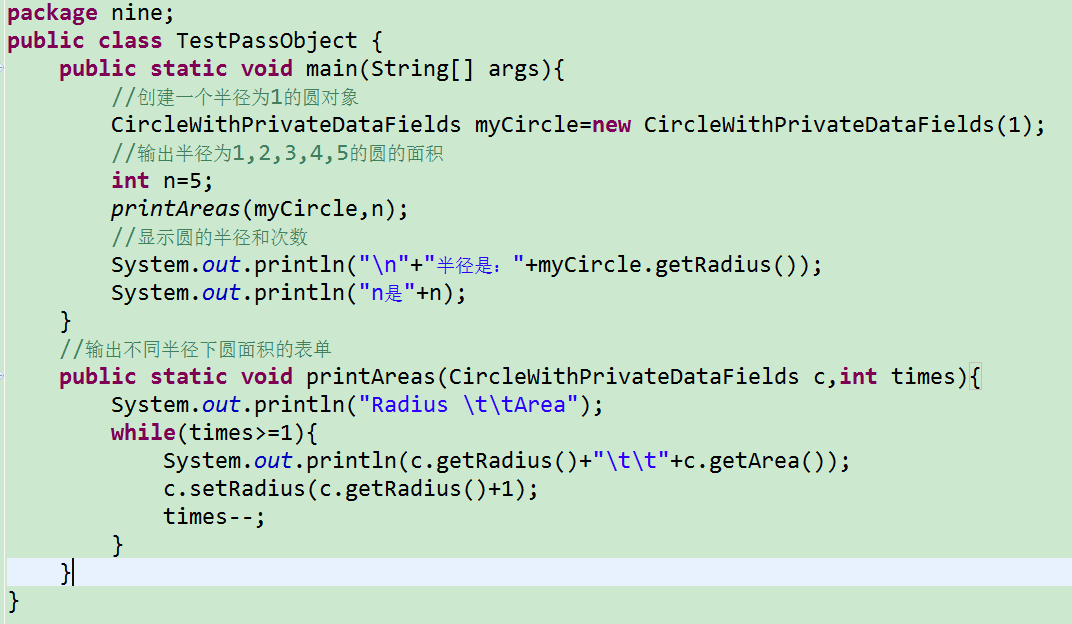
数据域radius被声明为私有的，私有数据只能在定义它的类中被访问，因为设置了get/set方法，所以类外也可以实现对他的访问和修改。由于，numberOfObjects是私有的，所以它是不能修改的。

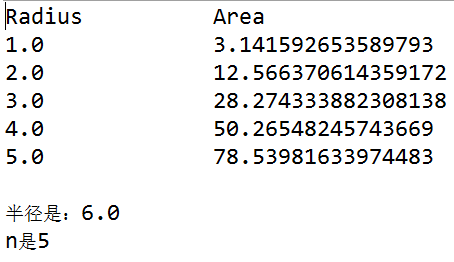
5.设计指南：为防止数据被篡改以及使类更容易维护，最好将它的数据域声明为私有的。

## 9.10向方法传递对象参数

1.要点提示：给方法传递一个对象，是将对象的引用传递给方法。

2.如下程序展示了 传递基本类型值 和 传递引用值的差异：





3.传递基本数据类型参数时，传递的是实参的值；传递引用类型的参数时，传递的是对象的引用。

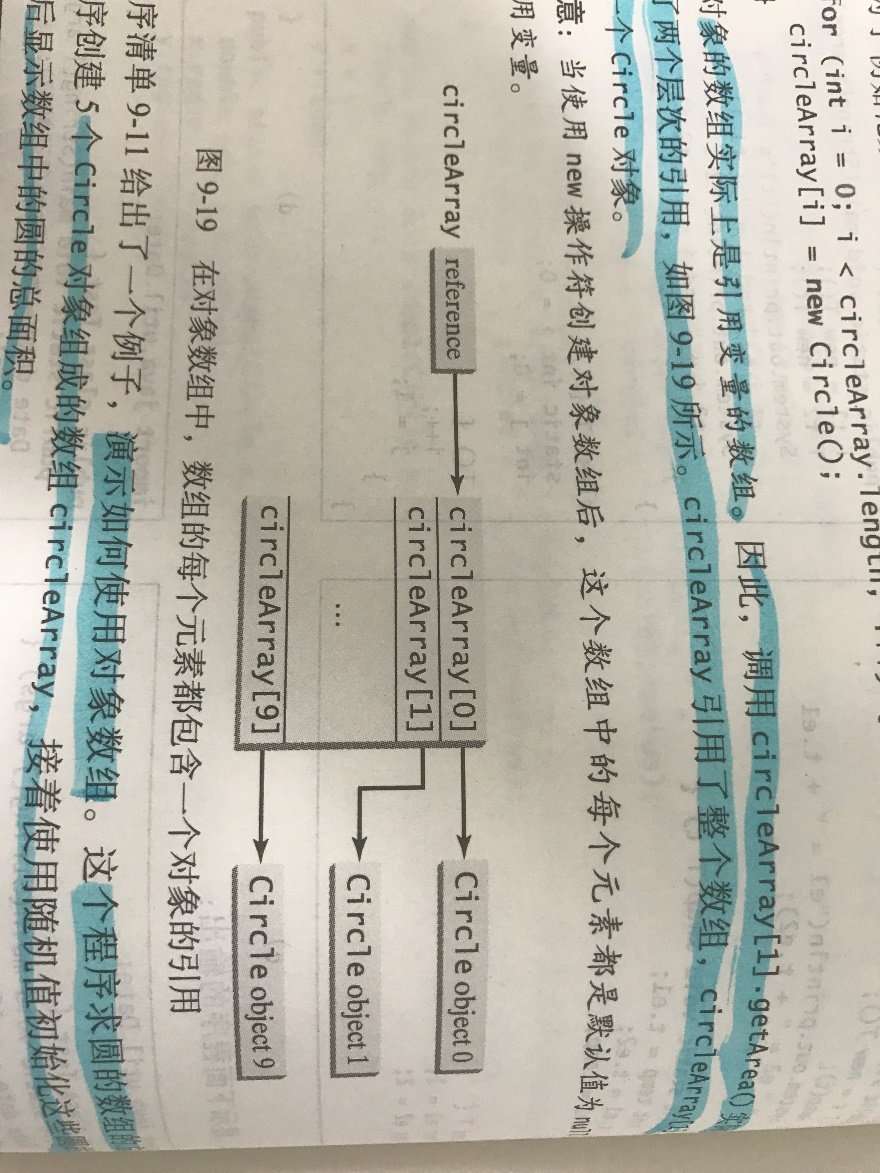
## 9.11对象数组

1.要点提示：数组既可以存储基本类型值，也可以存储对象。

2.例如，下面的语句声明并创建了10个Circle对象的数组：

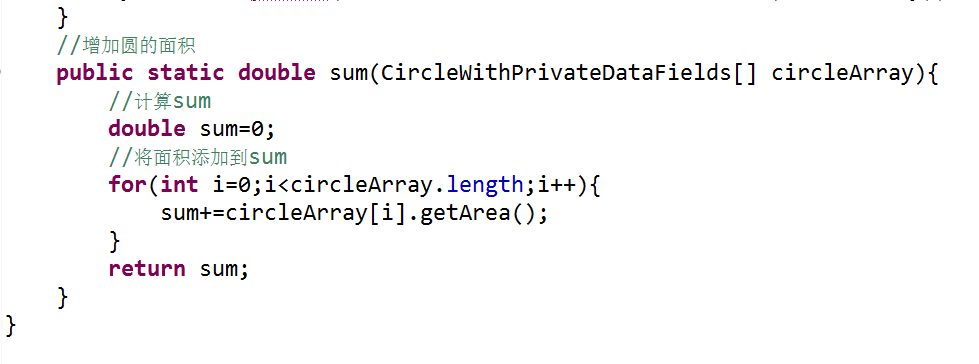
Circle[ ] circleArray = new Circle[10] ;

3.对象的数组实际上是引用变量的数组。因此，调用circleArray[1].getArea( )实际上调用了两个层次的引用，如图所示：circleArray调用了整个数组，circleArray[1]引用了一个Circle对象。



4.下面的例子，演示如何使用对象数组。这个程序求圆的数组的总面积。程序创建了5个Circle对象组成的数组circleArray ，接着使用随机值初始化这些圆的半径，然后显示数组中圆的总面积。





（程序有错，没能解决The method printf(Locale, String, Object[]) in the type PrintStream is not applicable for the argume）

## 9.12不可变对象和类

1.要点提示：可以定义不可变类来产生不可变对象。不可变对象的内容不能被改变。

2.不可变对象：创建一个一旦创建其内容就不能再改变的对象，我们称之为“不可变对象”。

3.不可变类：不可变对象的类称之为“不可变类”。例如String就是不可变类。

4.要使一个类称为不可变的，它必须满足下面的要求：

①所有的数据都是私有的。

②没有修改器方法。

③没有一个返回指向可变数据域的引用的访问器方法。

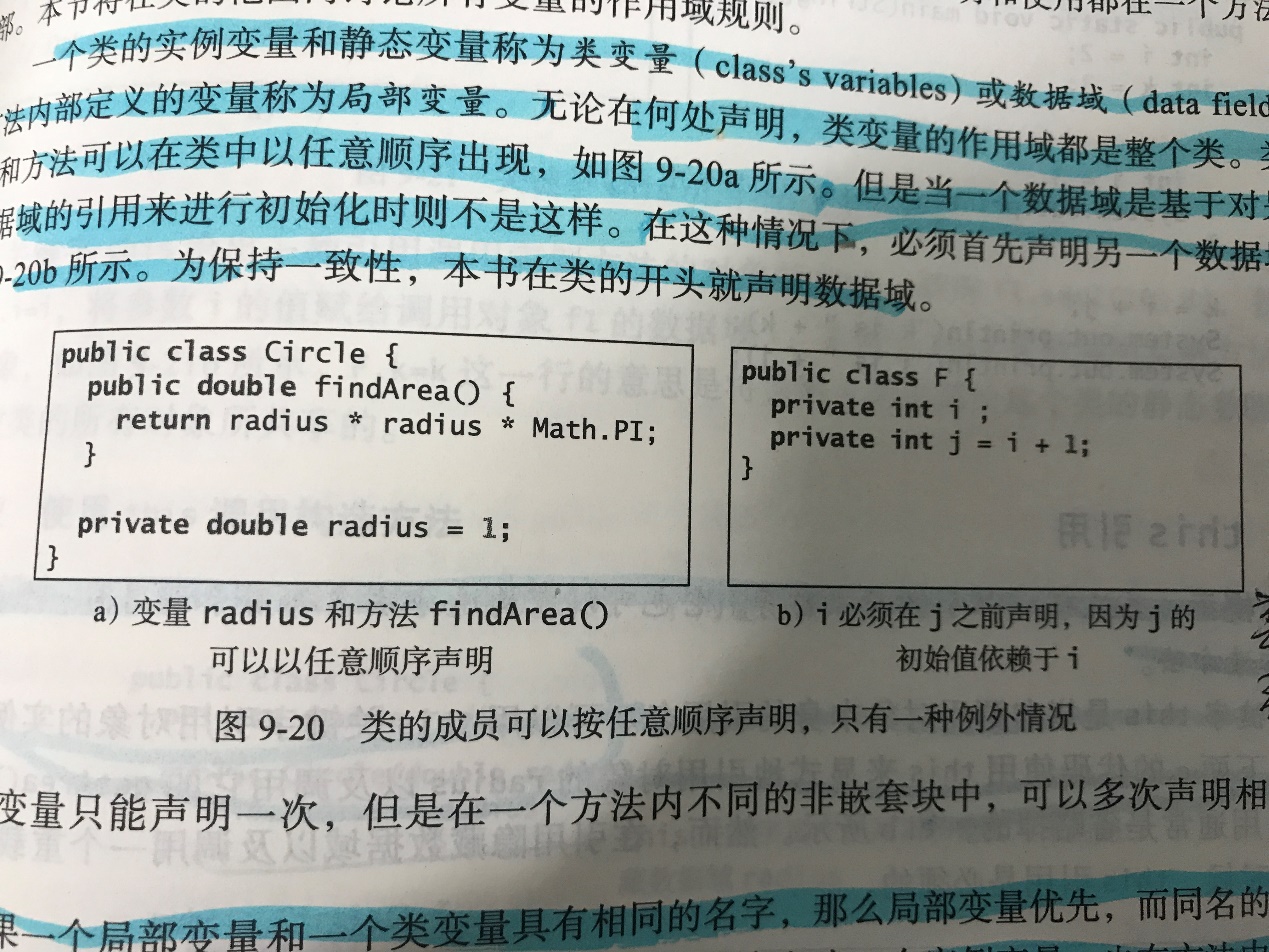
## 9.13变量的作用域

1.要点提示：实例变量和静态变量的作用域是整个类，无论变量是在哪里声明的。

2.类变量/数据域：一个类的实例变量和静态变量称为类变量或数据域。

3.局部变量：在方法内部定义的变量称为“局部变量”。

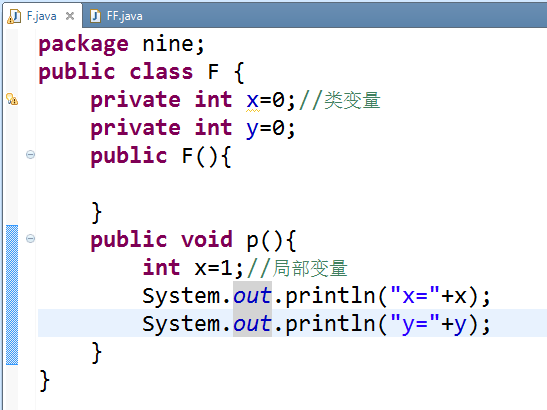
4.无论在何处声明，类变量的作用域都是整个类。类的变量和方法可以在类中任意顺序出现，如图a所示。但是当一个数据域是基于对另一个数据域的引用来进行初始化时则不是这样的。在这种情况下，必须首先声明另一个数据域，如图b所示。



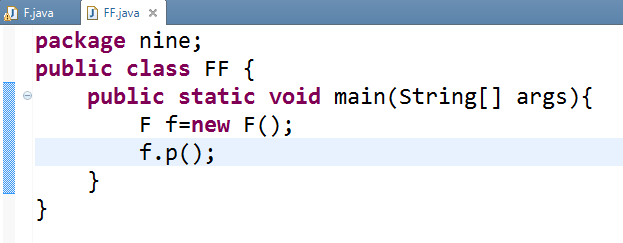
（简而言之，类变量可以任意顺序出现；当一个数据域基于另一个数据域时，需要先后顺序）

5.如果一个局部变量和一个类变量具有相同的名字，那么局部变量优先，而同名的类变量将被隐藏。例如：在下面程序中，x被定义为一个实例变量，也在方法中被定义为局部变量。

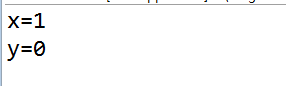
功能类



主函数



控制台

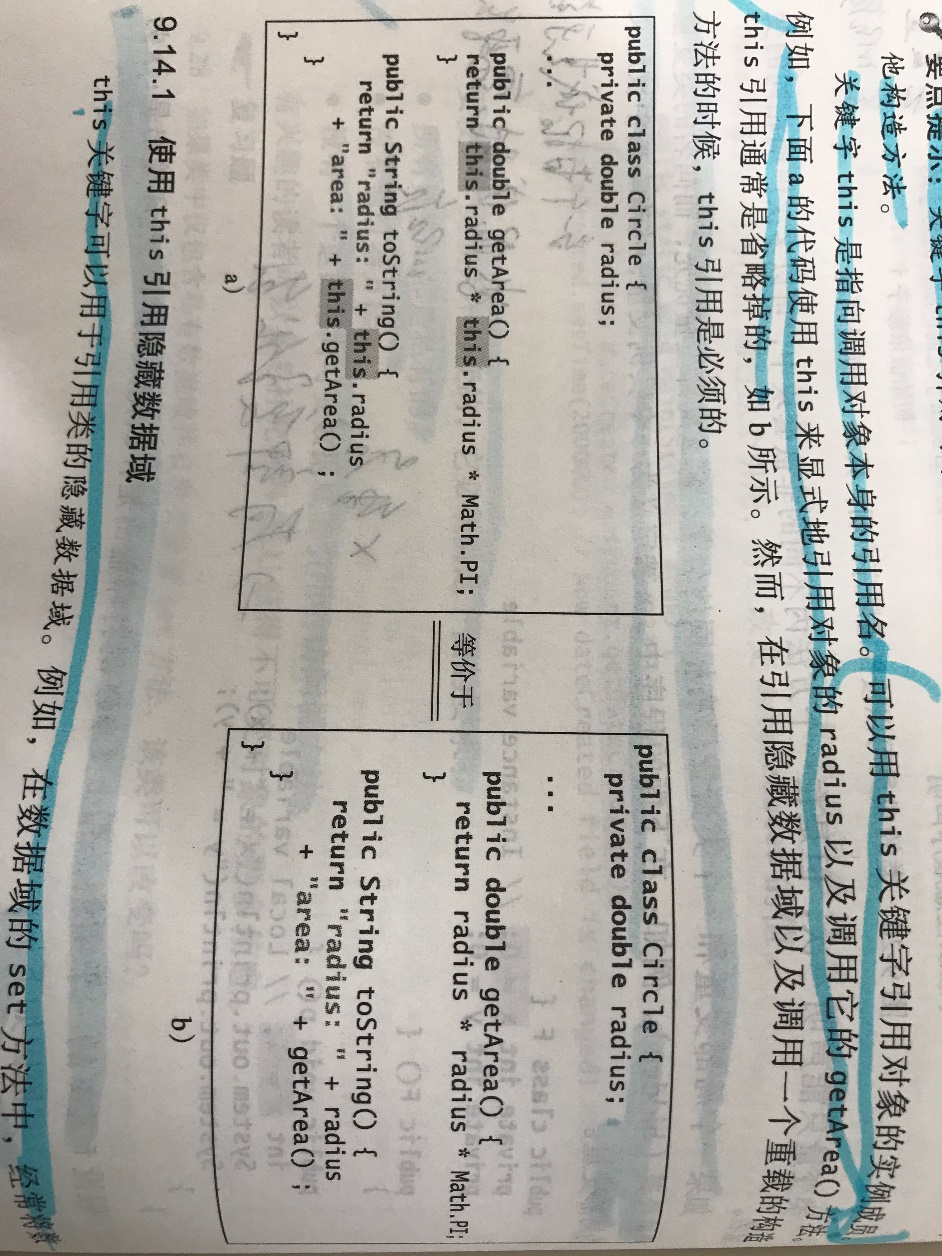


//这里X重名，既有局部变量又有类变量。两者冲突的情况下，局部变量优先。所以在输出的时候，会输出局部变量的值。

## 9.14 this引用

1.要点提示：关键字this 引用对象自身。它也可以在构造方法的内部调用同一个类的其他构造方法。

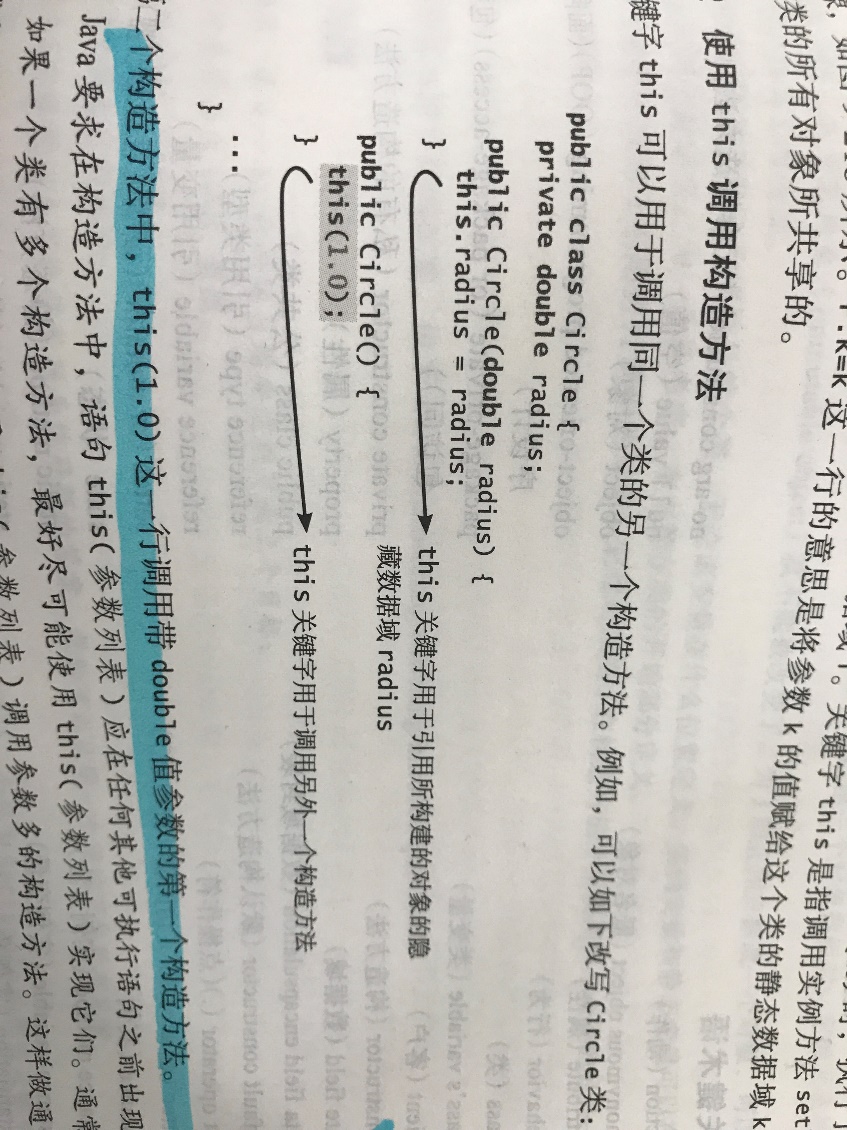
2.关键字this 是指向调用对象本身的引用名。**可以用this关键字引用对象的实例成员。**例如，下面a的代码使用this来显示的引用对象的radius以及调用它的getArea( )方法。This通常是省略掉的，如图b所示：



### 9.14.1使用this引用隐藏数据域

1.this关键字可用于引用类的隐藏数据域。例如，在数据域set方法中，经常讲数据域名用作参数名。在这种情况下，这个数据域在set方法中被隐藏。为了给它设置新值，需要在方法中引用隐藏的数据域名。隐藏的静态变量可以简单通过“类名.静态变量”的方式引用。

### 9.14.2使用this调用构造方法



在第二个构造方法中，this（1.0）这一行调用带double 值参数的第一个构造方法。

This 相当于建立了该类的一个对象，也可以说是简化了代码；