Interface

Interface 有点像abstract class的极端案例，但是Interface 不是class而是一种可以通过implement(接口)了interface的任意class来满足的type

用来描述interface的语法与描述class的相同

唯一区别就是用interface代替了class

一个interface制定了一系列method， interface必须

1. 他有只有method heading和constant definition常规描述
2. 他不包含实例变量或者任何完整的Method 描述

（implement的效果：同时接口多个·“父类”）

Interface的功能与base class相同，虽然他不是base class

Multiple inheritance：一些语言允许一个class同时集成两个或多个base class，然而在JAVA里，这是不被允许的，JAVA通过interface来近似的达到此效果

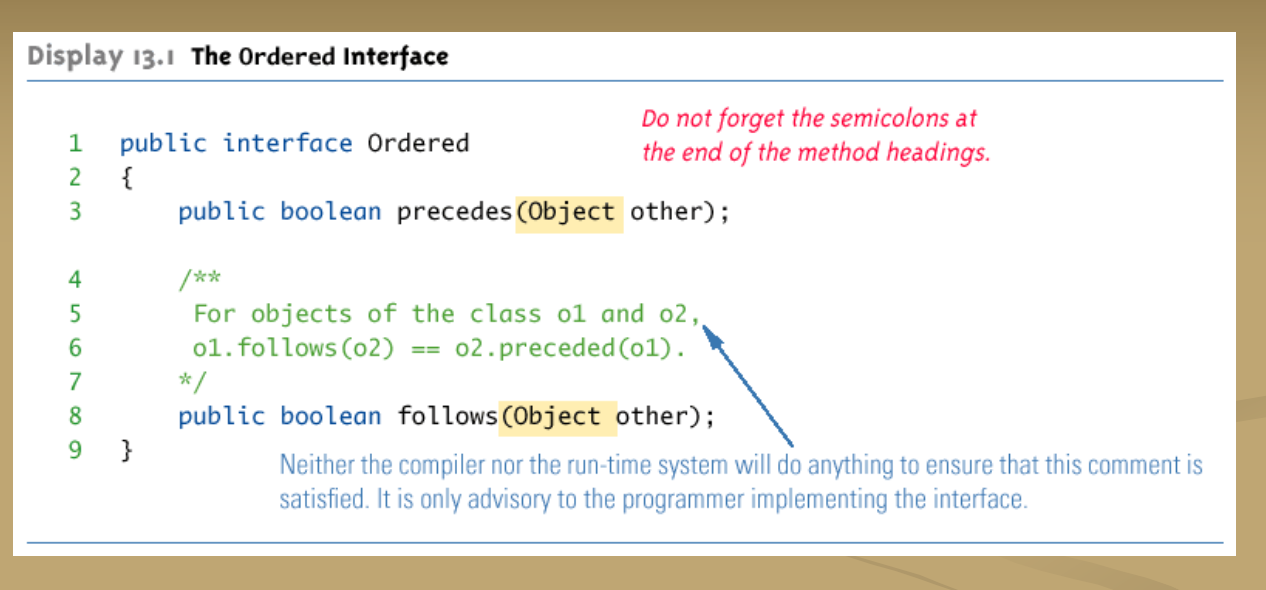
一个interface和他的所有method heading的前缀应该是Public

他们无法定义为private ,protected或者package access

当一个class implement（接口）了一个Interface时，他必须使interface里的所有Method 是public的

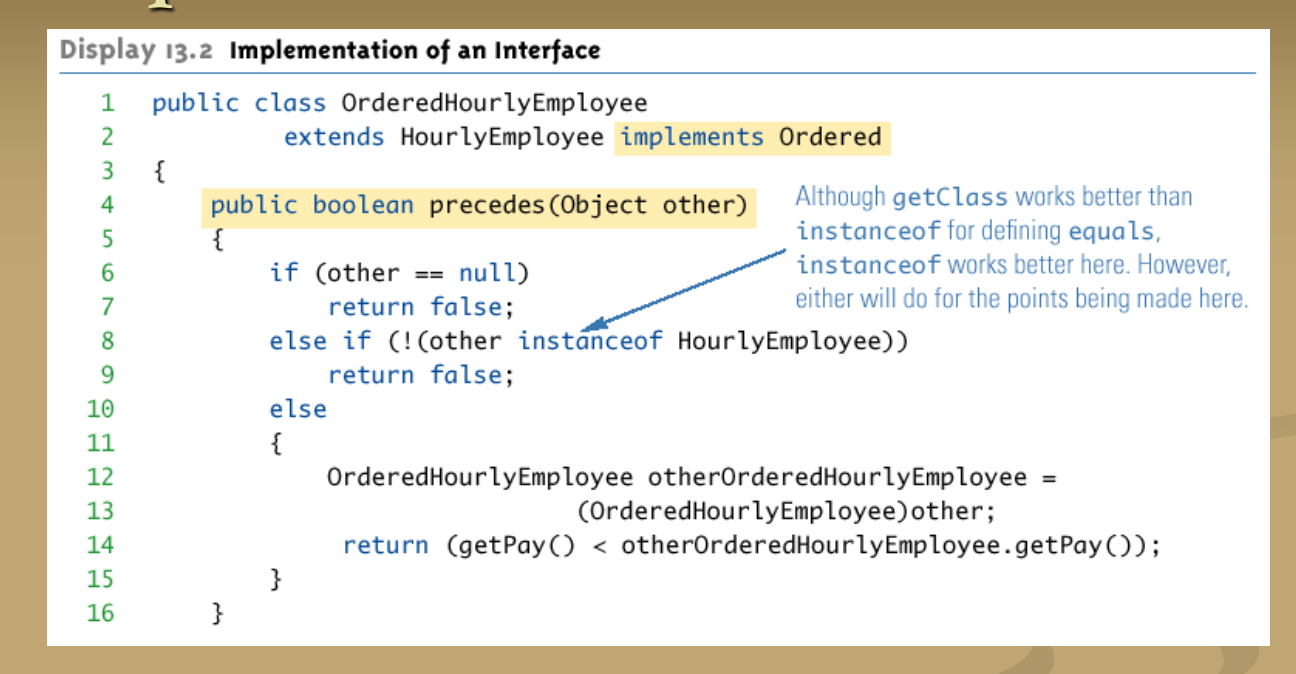
因为interface是一种type，所以一个method的参数可以是interface type。这个参数可以接受任意implement这个interface的class

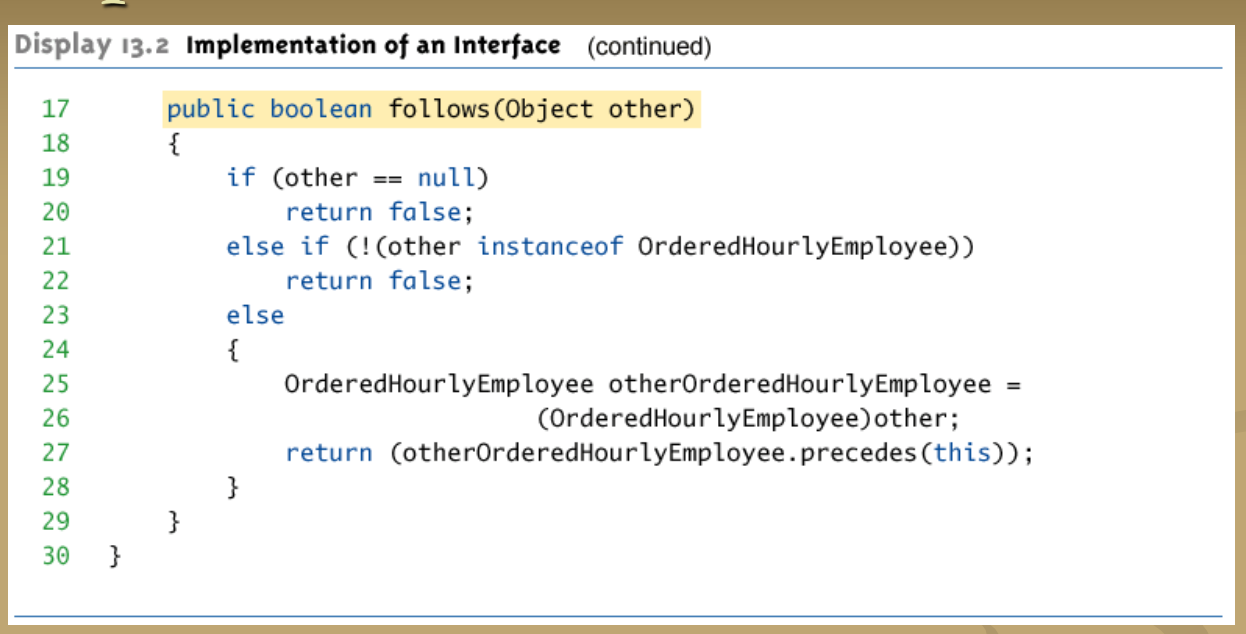
注意了，不加花括号，直接分号结束



为了implement一个interface,一个具体的class必须做两件事：

1. 他必须有语句，在class描述的开头， 如果要implement 多个，就用逗号隔开
2. 这个class必须implement所有interface definition里的method heading



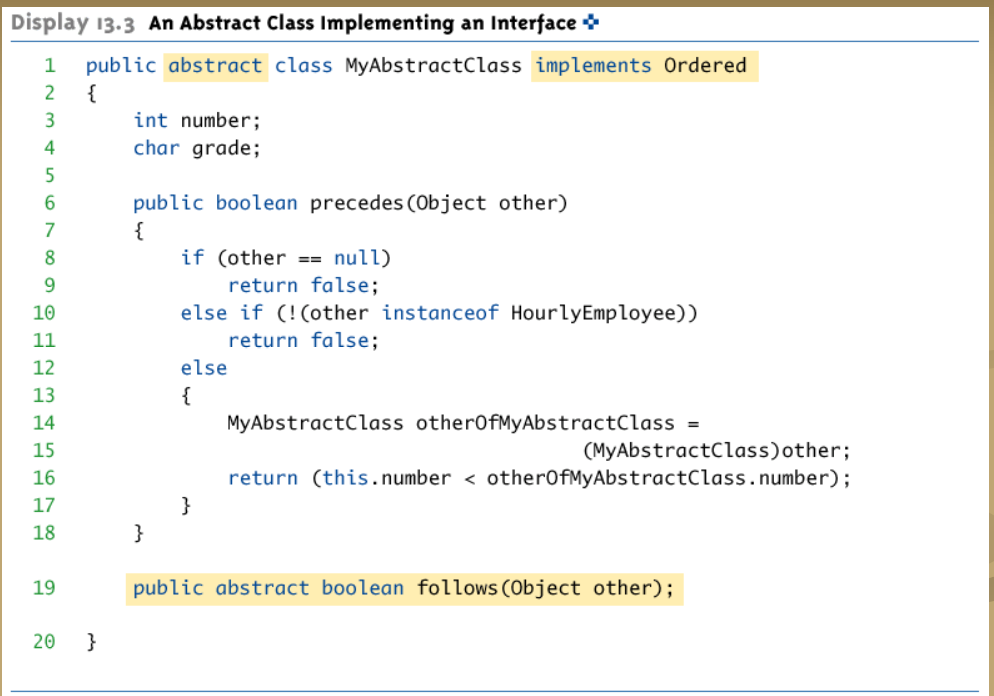


Abstract class implementing interface

一个abstract也可以Implement 一个或多个interface

所有的interface里的method heading都要被做成abstract method（如果有完整描述，就不用加abstract，没有完整描述，仅有一个heading要加abstract）

一个标准（非abstract）class必须描述所有的abstract class里的Method heading和interface的method heading

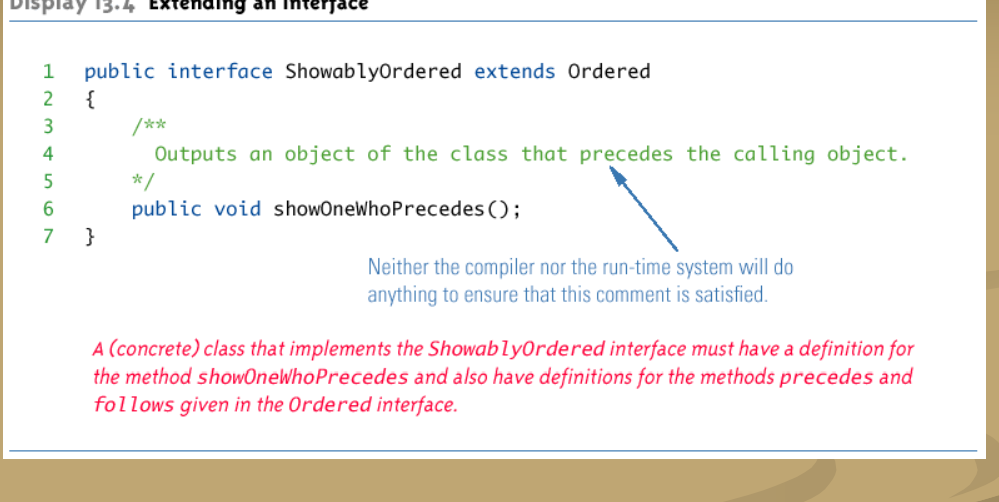


Derived interface

如同class一样，Interface可以继承base interface

这叫做extending the interface，格式如下

如果一个标准class implement了一个derived interface，那么它不仅必须有derived interface的method，还要有base interface的method



当一个class implement一个interface时，compiler和run-time system会检查Interface的语法和接口

然而，compiler与run-time system都不会检查interface的内容是否与他的潜在意思相一致

所以通常需要给interface加上semantics（注释）

这就可以让每个编程人员都能Implement这个Interface,并且按照他的潜在意思进行

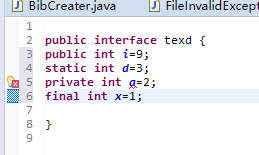
如果method body不满足潜在意思，那么写出来的软件可能无法正常运行

Defined Constant in interface

Constant:常量，一种特殊的变量，变量都是可以改动的，常量无法改动

前缀必须为public/static/final

Interface中可以有常量·，任意implement这个interface的class都可以使用这些常量



只能extend一个class：因为不同的class可能对同样的method heading有不同的描述，你不知道继承那个

可以implement 多个Interface:没有实际的method definition，不会出现此问题

但会出现其他的问题

1. 继承常量的时候常量名字相等，值不等
2. Method 有相同的名字，但有不同的return type

如果class definition同时implement了这样两个Interface,就会有一个error, class definition也会Illegal

一个极端但相当常用的Interface就是Serializable interface

他既没有method headling又没有defined constants，他就是纯粹的什么都没有

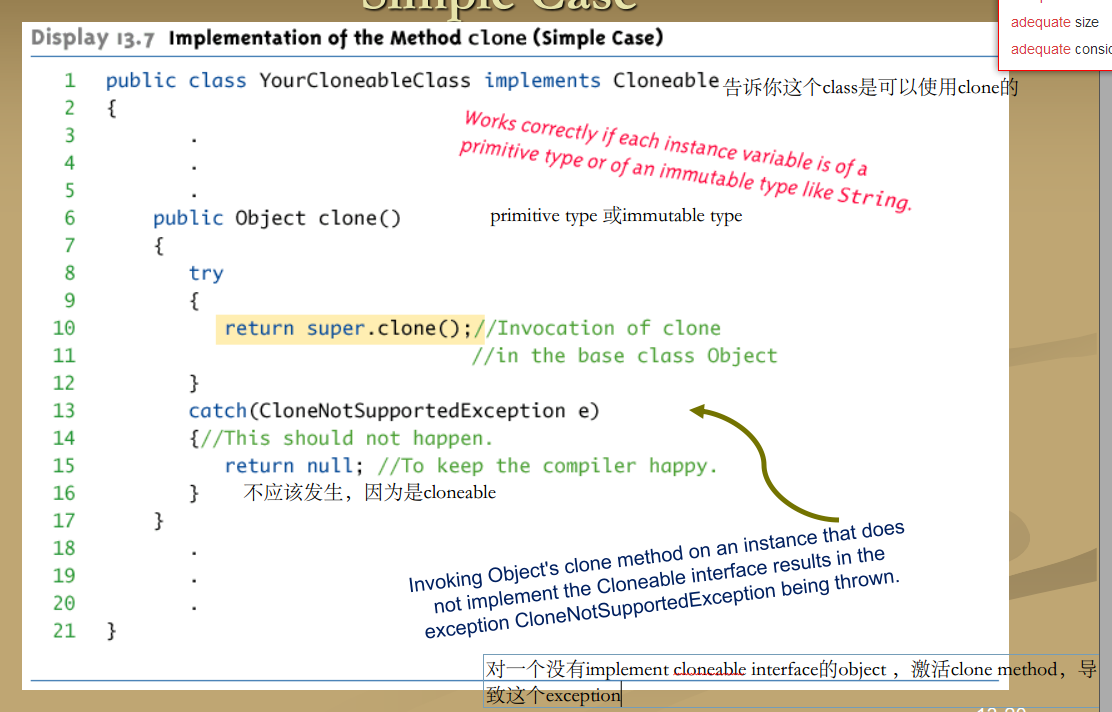
它仅仅用来作为一个tag向系统指出它有可能implement file I/O以一种特殊的形式

 interface是另外一个特殊interface

他没有method heading或defined constant，他只用来指出method 应该怎样被使用或者重新描述

Method 对存储内的object信息进行了bit-by-bit copy  
如果data都是primitive type data 或者immutable class type 的data（比如String）,那么就是adequate（胜任的）

接下来的例子就是一个简单的没有mutable class type（可修改类型的）的Instance variable的class，也没有特殊的base class（所以base class是Object）



如果一个被clone的object里有mutable class的Instance variable ，那么clone就会导致privacy leak

如果对于这样的一个class implement 一个cloneable interface

那么：

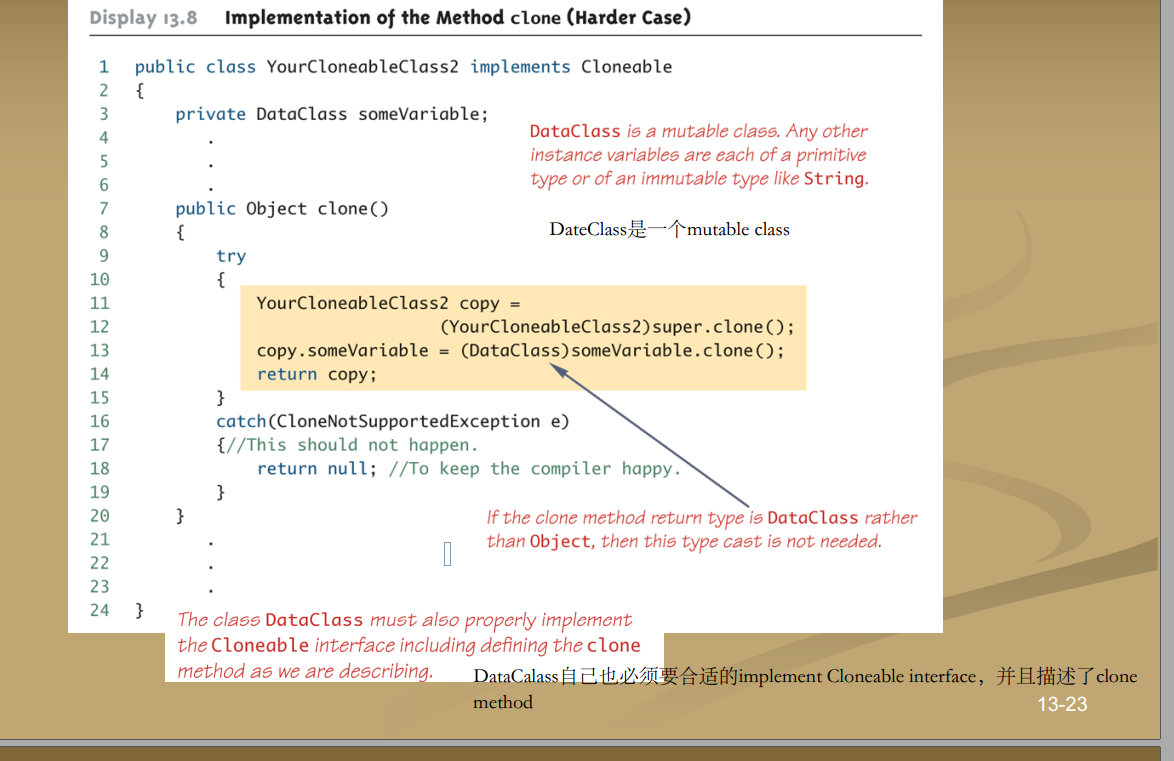
首先激活base class 里的clone method，



然后把mutable class type重新设置一个instance variable的值。然后再用这个Instance variable的clone(这个clone是instance variable所属的class中的method，这个class也implement 了 clonable interface)



注意这只有cloneable interface被合适的implemented的时候才能生效



Inner Class

Inner Class指的是class被其他class所描述

这个包括inner class的class叫做outer class

Inner class的描述必须要在outer class里面

一定要把inner class放在开头或结尾,确保它容易找到

Inner class的definition也是outer class 中的member，outer class的instance variable和Method也是outer class的member，这三者是并列关系

Inner class是local对于outer class definition来说

Inner class的名字也许会被Outer class definition 以外的class(比Outer class还Outside)使用

如果Inner class是private的，就不能被outside outer class使用

Inner class两个好处，可以让outer class更加独立因为他们被描述在同一个class里

Inner class和outer class的method 互相拥有进入private method 与instance variable的权限

把inner class作为一个辅助class是Inner class最有用的应用之一

如果作为辅助class，必须是private的

在inner class内的method definition:

可以使用Outer class的实例变量作为参数

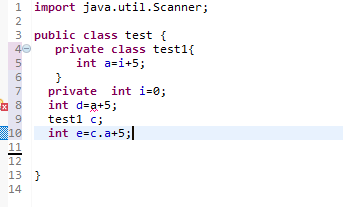
可以使用outer class的private method

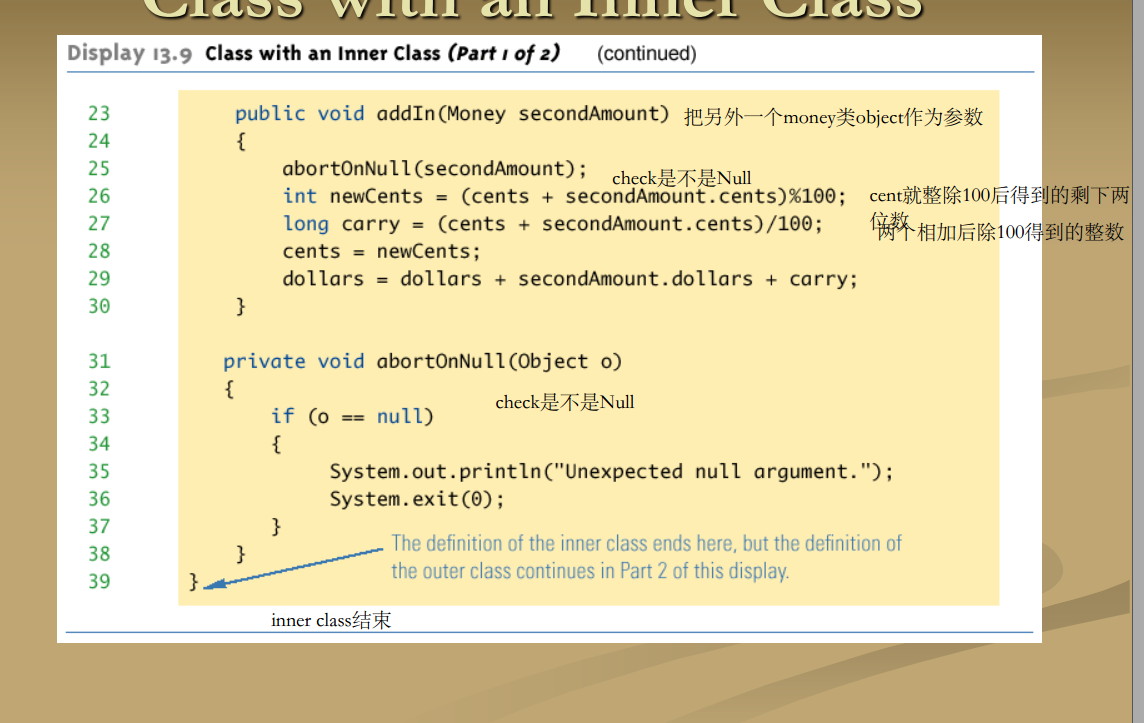
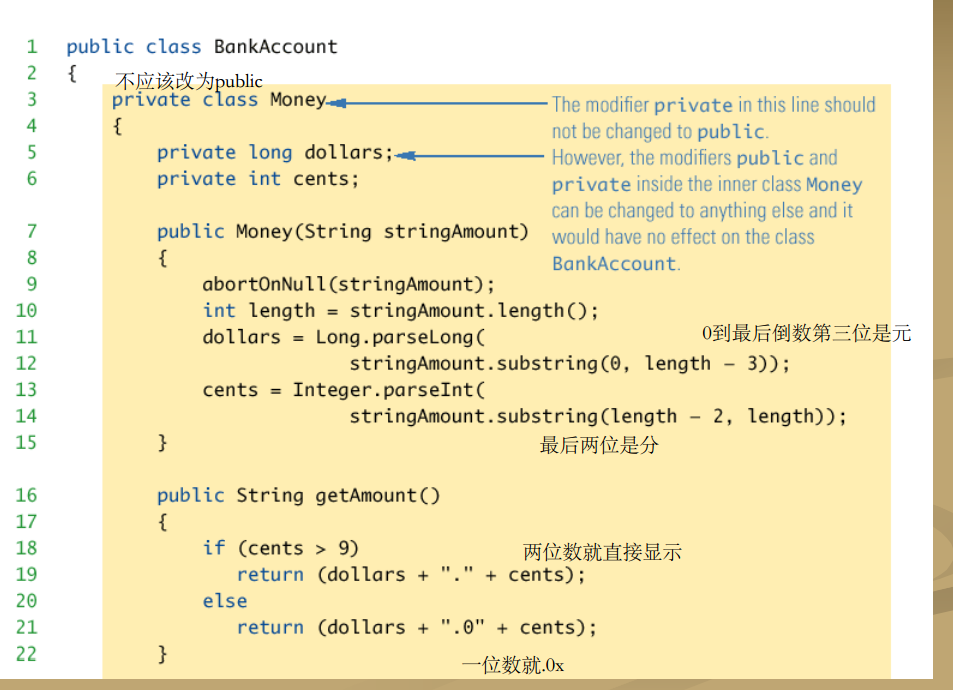
在outer class内的method definition：

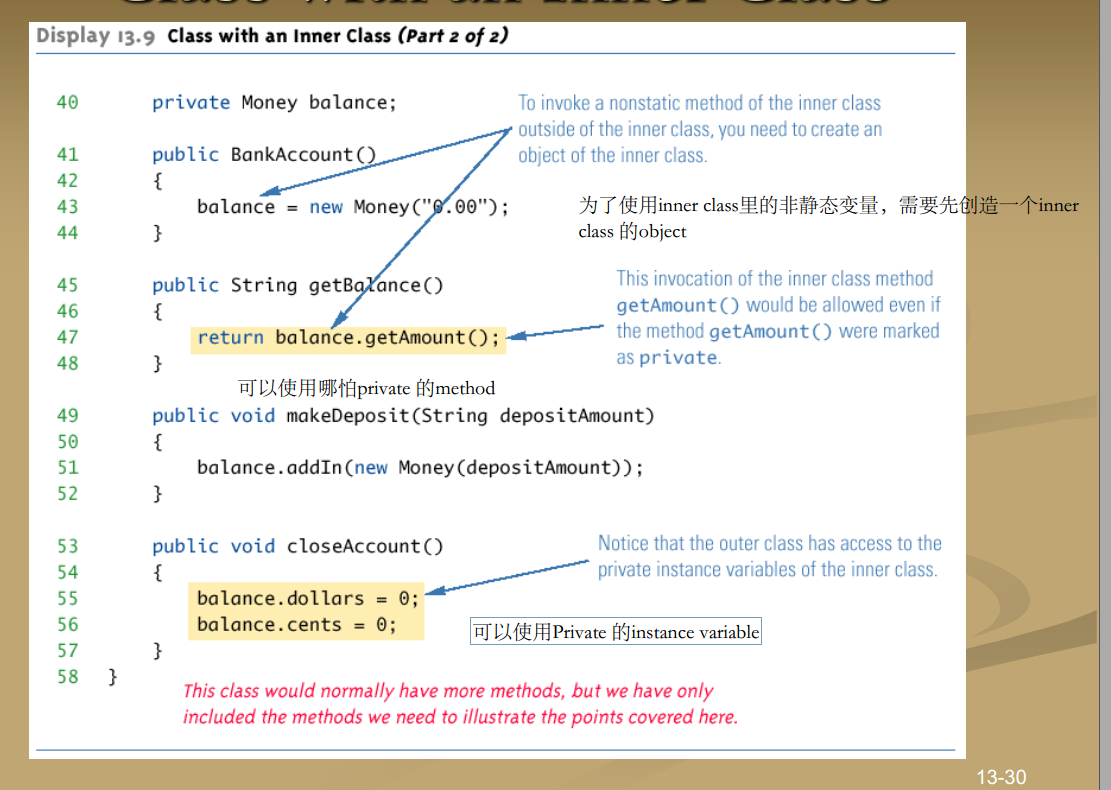
可以使用inner class的object中的实例变量作为参数

可以使用inner class的object的method

所以Inner class与outer class 的Public与Private是等价的



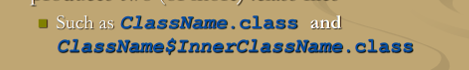




Inner class中的.class File

在JAVA中编译任何class都会产生一个.class file名字是

编译一个inner class会导致一个两个class都被编译，产生两个.classfile

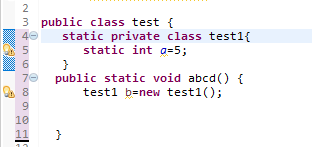


Static Inner Class

一个普通的inner class通过他的object与outer class链接

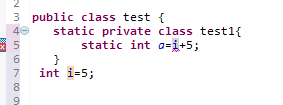
但以下情况，inner class必须是static的

1. 如果inner class的object是在outer class里的static method里创造出来的
2. 如果inner class必须有static member(variable或method)



因为static inner class与outer class的object没有联系，所以Inner class method不能使用outer class的instance variable

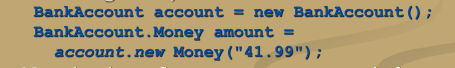
不能激活outer class的nonstatic method



为了使用static inner class的static method或者static variable，就直接用inner class的name.xx就行

如果inner class前缀是Public,那么outer class之外也可以使用

如果这个inner class是nonstatic的，那么它必须通过使用outer class的object来创造

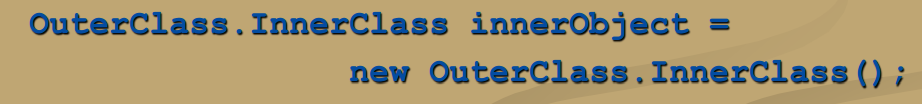


BankAccount 是outer class,.money是Inner class

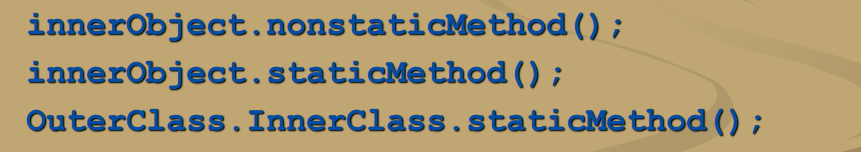
注意account.必须在new 之前

然后新object amount就可以Invoke inner class中的method了，但只能是inner class里的Method

如果是public 并且static



这三种都是可以接受的



如果inner class里面试图使用一个method

如果Inner class无此method，那就用outer class的此name的method

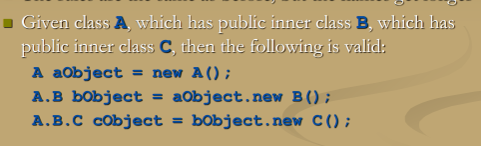
有，就用inner class的

如果有，还要用outer，必须使用以下invocation方法



Nesting inner classes

可以在Inner class里面在加Inner class，所有逻辑如上



对于一个有innerclass的outerclass来说

Outerclass的子类会自动继承innerclass作为一个innerclass，子类不能override这个innerclass

Outerclass可以作为子类

Innerclass也可以作为子类

Anonymous class匿名class

如果一个Object被创造了，但没必要给object的class命名，这时候就用了anonoymous class definition

这个class definition被嵌入在中

Anonymous classes有时被使用当他们被另外一个type的variable赋值的时候

另外的type通常是java interface

这个other type必须要做到，匿名类的object也是other type的object

