Inheritance: 继承 object-oriented programming：面向对象的编程



继承是 面向对象的 编程的一种main techniques技巧

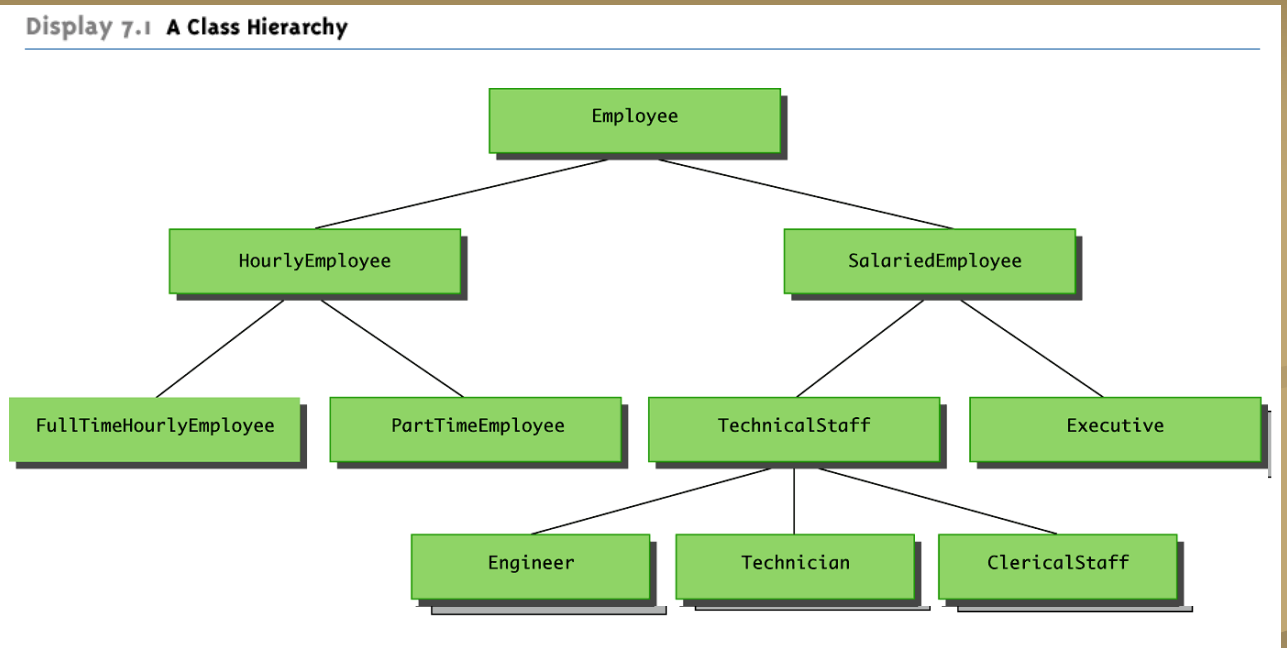
通过使用继承，还未创造的class可以通过已存在的CLASS创造，还未创造的class会继承已存在的class的实例变量和方法

新的class被称作为derived class 衍生类

原来的class被称作为base class 基本类

使用继承的好处就是 可以重复使用已存在的代码

当我们设计class时，要把他们分类，分层



extend baseclass（具体哪个基本类）必须要在衍生类的definition中写出来



derived class继承了所有base class的实例变量和method

derived class所创建的object也可以激活base class的method

一个子类只能有一个父类，如果没有extend那么就是继承了java.lang.object

也有另外一种叫法，derived class是child class子类，base class是parent class父类，父类以上是ancestor class祖类，如果A是B的祖类，那么B就是A的孙类descendent class

override(重写)如果子类从父类那里继承了一个Method,那么它可以change或override这个Method

当在子类override父类的一个method的时候，return的type(int string啥的)是不能改变的

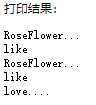


covariant return type:协变返回类型

如果是class type，那么return的就可以是returned type的子类

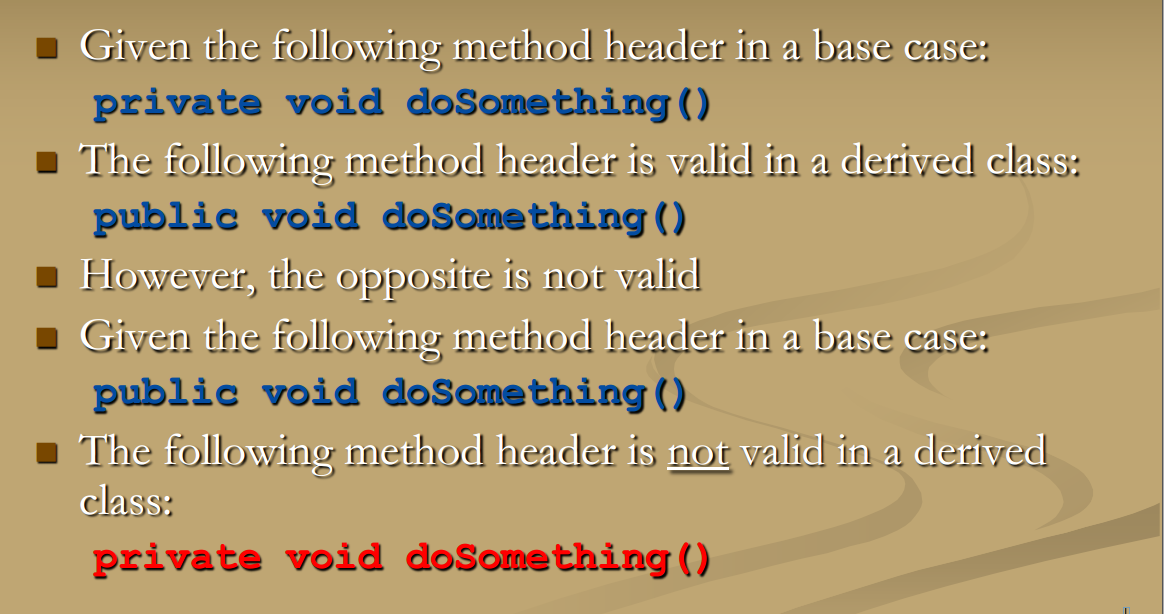


**导出类**（子类）覆盖（即重写）**基类**（父类）方法时，返回的类型可以是基类方法返回类型的子类。



access permission进入许可: overriden的method的进入许可可以从父类的private method改成子类的public method(或更加自由的其他形式)

但是不可以从父类的Public改成子类的private或其他更严格的形式

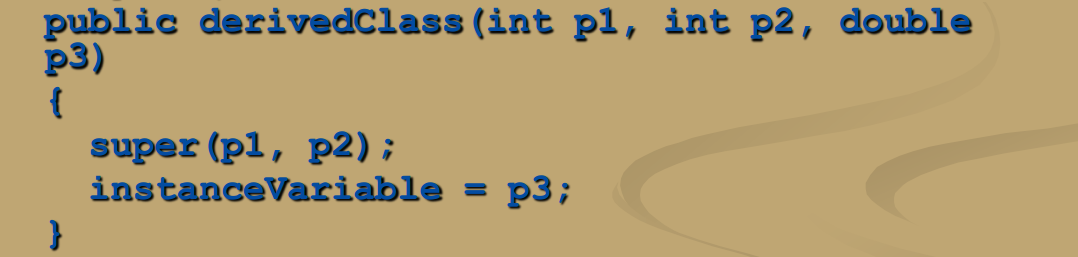


Overload是重载，是有相同的方法名，但参数类型或个数彼此不同  
Override是重写，是在子类与父类中，子类中的方法的方法名，参数个数、类型都与父类中的完全一样，在子类中覆盖掉了父类的改方法。

如果父类method之前有个final，那么子类就不能将他重写

如果父类class之前有个final，那么这个class就不能作为父类存在，不能有继承

super constructor:子类用父类的一个constructor来初始化所有从父类继承的所有数据



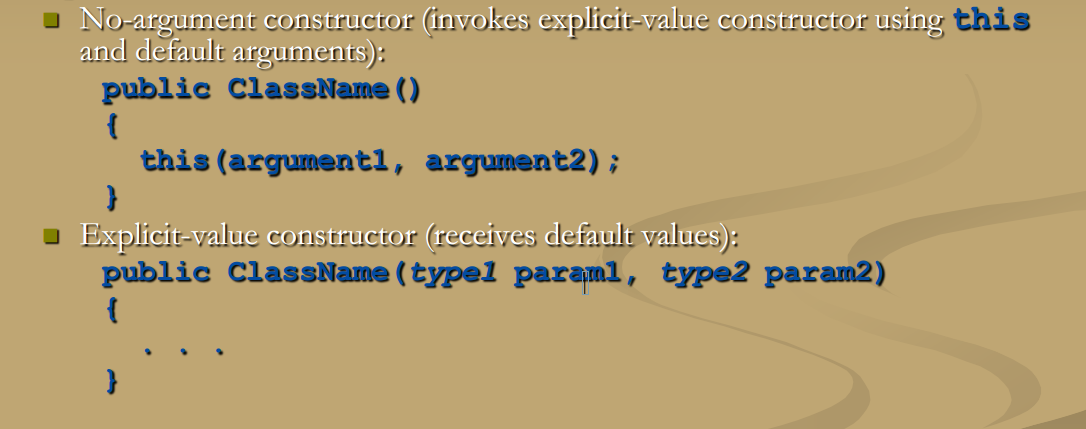
叫做base class constructor

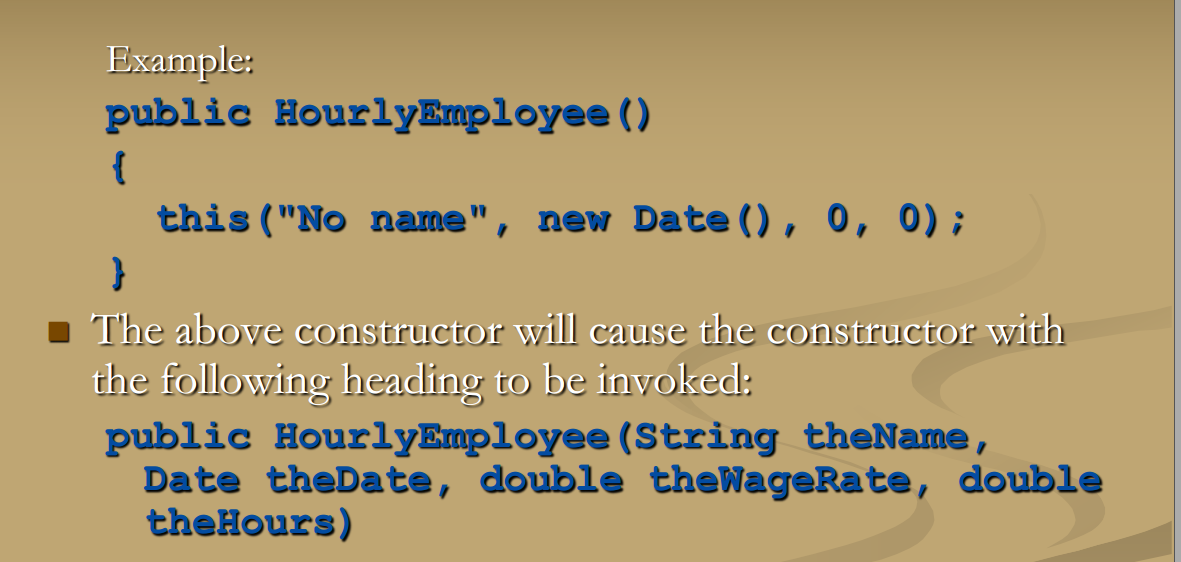
呼唤base class的constructor一定不能用父类的名字，而要用super代替

子类的constructor第一步一定要是call super父类，不然就会默认call default constructor

this constructor:有时候在一个类中写了多个constructor，想在这个constructor中call另外一个constructor，就要用this，不可以同时使用this 或super ,想要同时使用this或super，就要先用this呼唤另一个constructor，然后在另一个constructor内，第一步就呼唤super

通常一个没有参数的constructor用this来呼唤explict value constructor(有明确值的constructor)





父类的private instance variable不可以在其他类（包括子类）中直接使用，一定要通过父类定义的accessor 和Mutator获得进入权限access

如果说子类有一个method通过method描述已经获得了对于父类Private instance variable 的进入权限，那么其他人可以通过这个method来进入这个instance variable

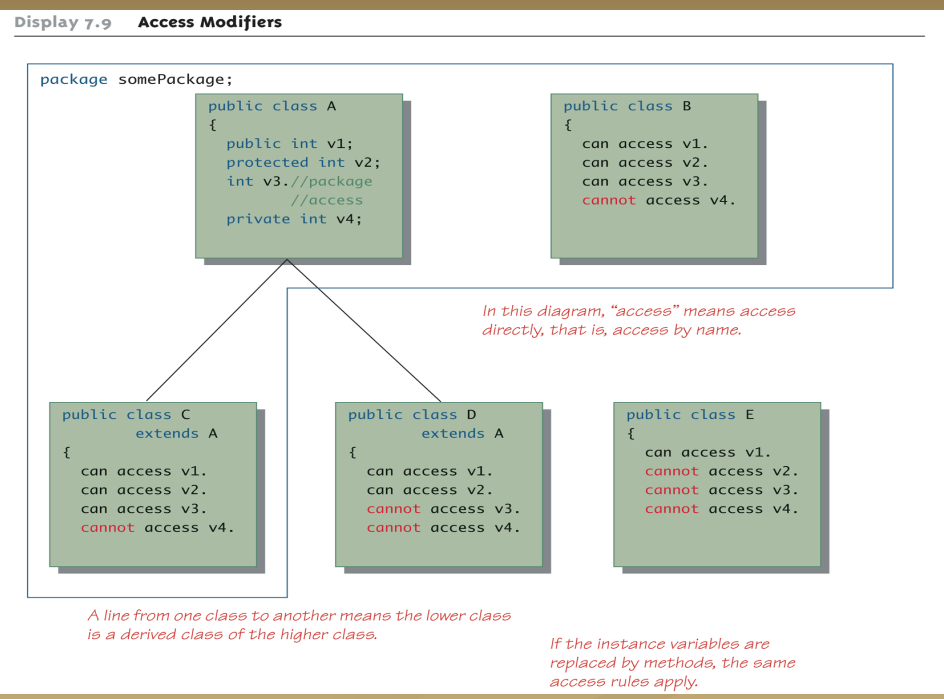
父类的private method 也是无法直接使用的，他只是作为父类public method的辅助Method，如果你要使用，就应该用Public

protected前缀：在自己的class，子类的class，同一package的class，都可以直接通过名字access。

因此和private比起来保护性相当低，所以instance variable应该避免加protected前缀

什么前缀都没有 的ACCESS权限是：package access/default access/friendly access

在package内任意class可以通过Name access

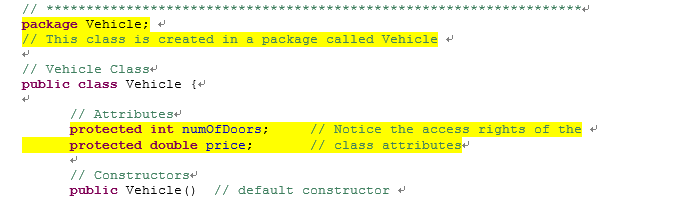


所有在当前目录current directory（不属于其他package）属于一个没有名字的package——default package,这时候如果一个Instance variable或者method有package access，那么就可以直接进入

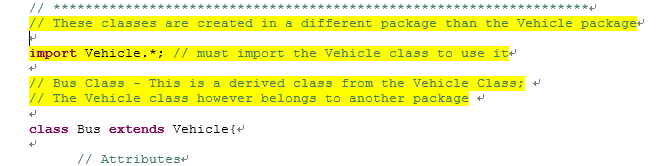
如果class B继承自class A,而A有一个protected instance variable n，但是CLASS A 与B在different package,那么：

1. B中任意一个method可以通过name直接激活n
2. B中的method可以创造一个local object，这个object可以通过name 激活n

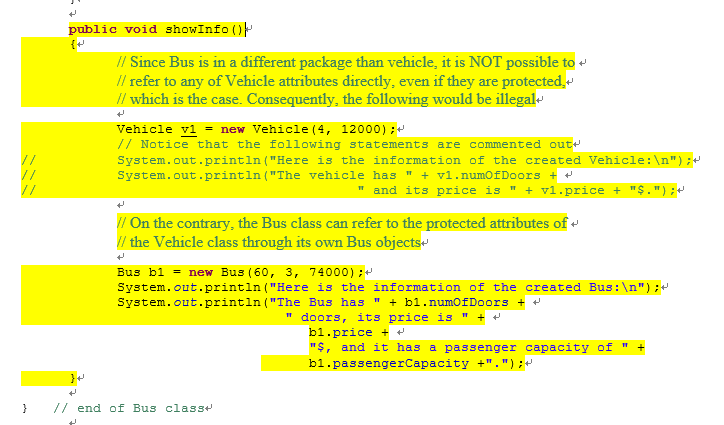
例子  
Vehicle package里由两个protected变量



这时候在另外一个不同的package里创造一个子class，如果想要继承Vhicle，就要先import整个package



因为bus和Vehicle不在一个package里，所以不能直接提取Vehicle变量，即便是protected，虽然可以在bus里Vehicle一个v1出来，但是这个vehicle出来的v1不能通过name激活变量，但是bus可以bus 一个b1出来，而这个b1 object是可以直接通过name激活变量的，b1.numofdoors啥的都行



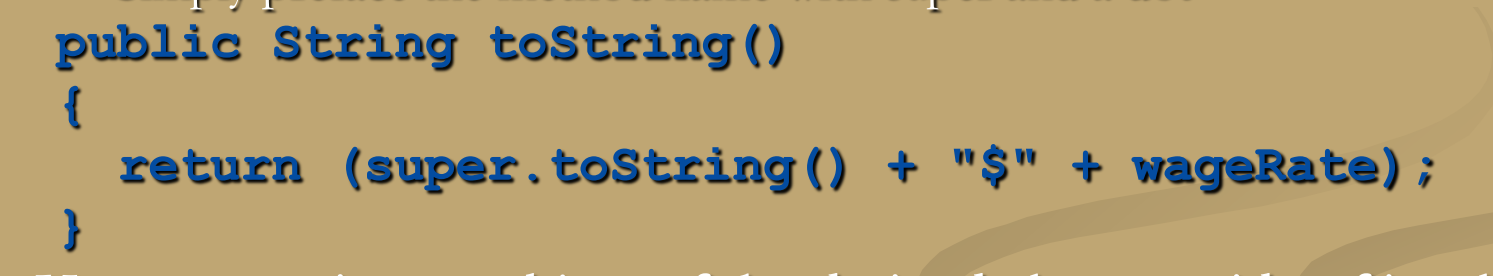
背后的原理：

一个class是继承了他父类的method和instance variable的

对于非public instance n B可以access n当他作为B的instantce variable，但是作为A的Instance variableB就没有ACCESS N的权利

Static variable也是会被所有子类继承的，public private等前缀作用保持不变（和普通variable一个效果）

父类一个被overriden的method还是可以被使用



但是如果在子类class difinition以外创造了一个子类的object，就不可能激活父类那个被overriden的method

不可以连环使用super,比如super.super.xxxxx是不行的

在java中，每个class都是Object class的孙类

所以每个class的每个object实际上都是Object type，同时也是他自己class的type

即便一个class没有明确说明他是另一class的子类，但它仍是object的子类

Object type也可以作为一个method的参数，object type的参数可以是任意class的Object(不一定非要在一个class)



判断左x是否是Vehicle的实例，return一个boolean值

**public** **static** **void** showVehicleInfo(Object x)

{

// Knowing that all Vehicle classes have overridden the toString() method

// if the passed object is one of these classes, then it is safe to use

// the toString() method to display its information

**if**(x **instanceof** Vehicle)

{

System.*out*.println("Here is the information of this vehicle");

System.*out*.println(x);

System.*out*.println();

}

**else**

System.*out*.println("The type of the passed object is unknown; will not display any information ");

}

这里直接把Object x作为参数，如果x是Vehicle那么….

不然就告诉你不是Vehicle

Object class自带一些Method比如toString equals

所有class都会继承这些method

但是我们应该在我们的class中重新override一遍让他更适合我们的需求



**public** **boolean** equals(SportCar sc)

{

System.*out*.println("Executing equals() from the SportCar Class");

**return** (**this**.getNumOfDoors() == sc.getNumOfDoors() &&

**this**.getPrice() == sc.getPrice() &&

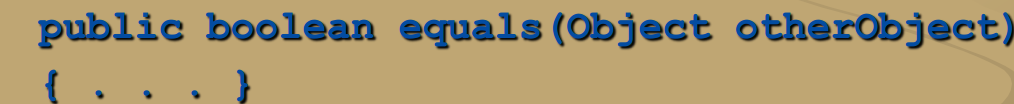
**this**.getNumOfSeats() == sc.getNumOfSeats() &&

**this**.gasConsumption == sc.gasConsumption);

}

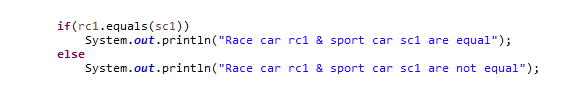
上面这种只是OVERLOAD

我们应该用override



为什么?因为这个EQUAL无法比较不同的class,会报错

例子



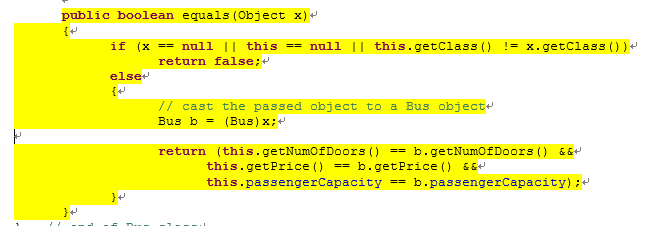


Equals必须满足以下条件

确认x和this不等于null

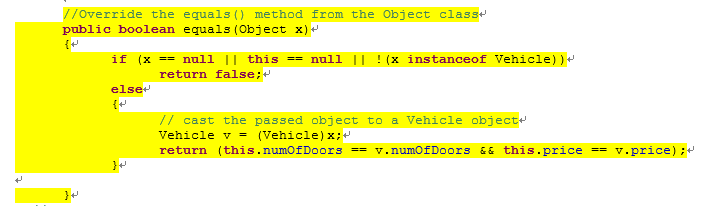
改变x的type，改成当前class的type,创造一个新bus

Return……全部相等

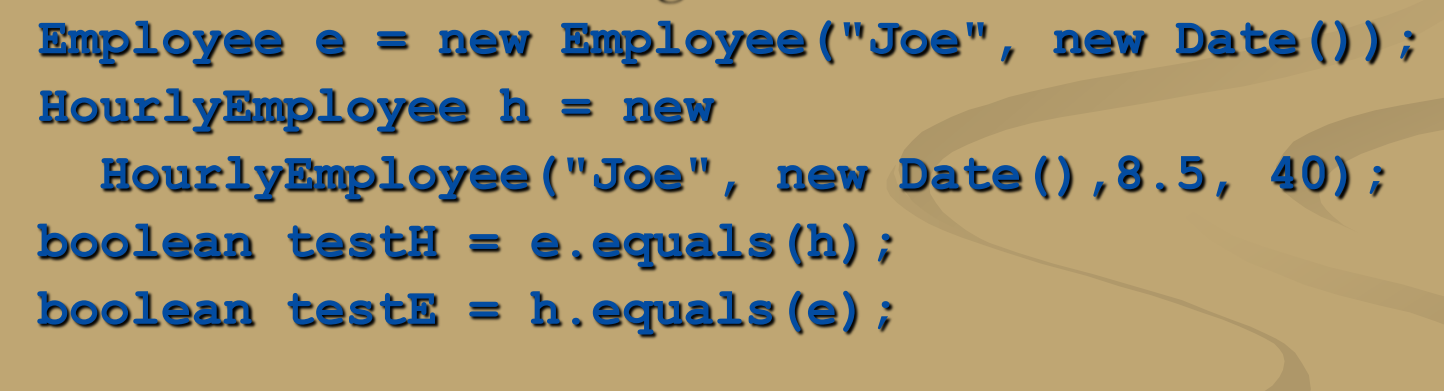


还有另外一种写法：用instanceof来替代getclass

X instanceof XXCLASS会return一个boolean值，是就return true，不是就return false



但是Instanceof也会return true当他是XXCLASS的子类或孙类



第一个是true因为HourlyEmployee是Employee的子类

第二个是false因为e不是HourlyEmployee

所以用getclass最好啦，因为他更精确，可以确切的指定一个值

但是不能用A.GETCLASS=XXCLASS

要A.GETCLASS==B.GETCLASS

所有的Object的getclass()这个method都是继承自Object Class

Getclass()在Object Class里已经有了final的词缀，所以不可以override

Getclass()只对那些用new产生的object有效

子类与父类的关系实际上是is a

HourlyEmployee实际上is a Employee

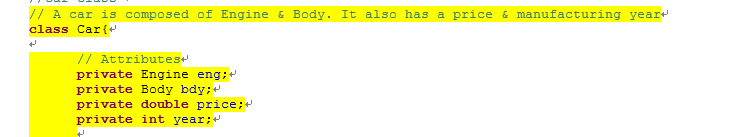
目的是从一个简单的base class创造一个复杂的derived class

从一个简单的base class创造一个复杂的class还有另一种方法 has a

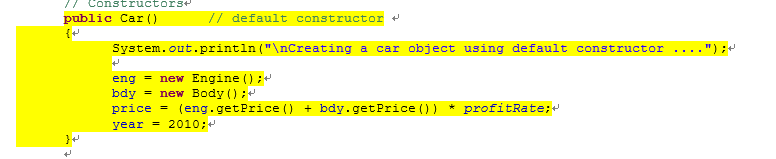
即composition

例如一个Car class,有实例变量tires,seats，而tires seats实际上是Tire Class和Seat Class的Object form

变量部分，注意没有extends

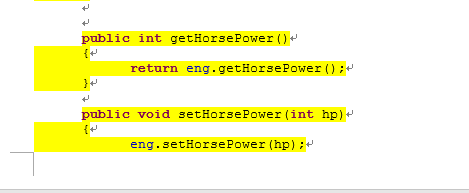


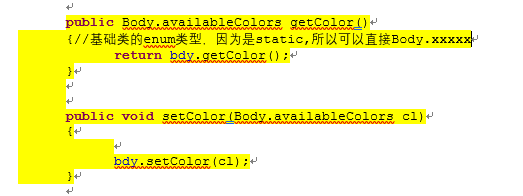
Constructor部分,给这些变量new 一个真正的object。



相当于Engine eng=new Engine();

GET SET实际上是overload，因为不是extends





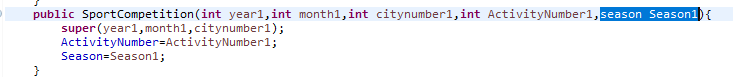
和继承的区别，要把父类的getset再写一遍，因为实际上没继承这些方法

继承constructor第一步Super,COMPOSITION第一步new object

Composition1.docx

Enum:定义一个 enum型，然后用这个enum就可以创造变量，变量的值只能从enum中取





赋值的时候

