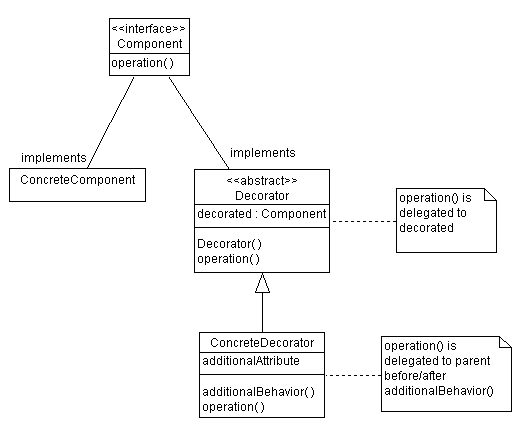
     学习AOP时，教材上面都说使用的是动态代理，可是在印象中代理模式一直都是控制访问什么的，怎么又动态增加行为了，动态增加行为不是装饰器模式吗？于是 找了很多资料，想弄清楚这两者之间到底有什么区别。结果发现这一篇英文文章讲的很清晰，就翻译一下，供参考。

        首先，让我们先看一下下面的这两个UML类图，他们分别描述了装饰器模式和代理模式的基本实现。



        这两个图可能使我们产生困惑。这两个设计模式看起来很像。对装饰器模式来说，装饰者（decorator）和被装饰者（decoratee）都实现同一个 接口。对代理模式来说，代理类（proxy class）和真实处理的类（real class）都实现同一个接口。此外，不论我们使用哪一个模式，都可以很容易地在真实对象的方法前面或者后面加上自定义的方法。

        然而，实际上，在装饰器模式和代理模式之间还是有很多差别的。装饰器模式关注于在一个对象上动态的添加方法，然而代理模式关注于控制对对象的访问。换句话 说，用代理模式，代理类（proxy class）可以对它的客户隐藏一个对象的具体信息。因此，当使用代理模式的时候，我们常常在一个代理类中创建一个对象的实例。并且，当我们使用装饰器模 式的时候，我们通常的做法是将原始对象作为一个参数传给装饰者的构造器。

        我们可以用另外一句话来总结这些差别：使用代理模式，代理和真实对象之间的的关系通常在编译时就已经确定了，而装饰者能够在运行时递归地被构造。

代理模式：

//代理模式  
public class Proxy implements Subject{  
  
 private Subject subject;  
 public Proxy(){  
 //关系在编译时确定  
 subject = new RealSubject();  
 }  
 public void doAction(){  
 ….  
 subject.doAction();  
 ….  
 }  
}

//代理的客户  
public class Client{  
 public static void main(String[] args){  
 //客户不知道代理委托了另一个对象  
 Subject subject = new Proxy();  
 …  
 }  
}

装饰模式：

//装饰器模式  
public class Decorator implements Component{  
 private Component component;  
 public Decorator(Component component){  
 this.component = component  
 }  
 public void operation(){  
 ….  
 component.operation();  
 ….  
 }  
}

//装饰器的客户  
public class Client{  
 public static void main(String[] args){  
 //客户指定了装饰者需要装饰的是哪一个类  
 Component component = new Decorator(new ConcreteComponent());  
 …  
 }  
}