**Decorator（装饰）**

意图

动态地给一个对象添加一些额外的职责。就增加功能来说，Decorator模式相比生成子类更加灵活。

动机

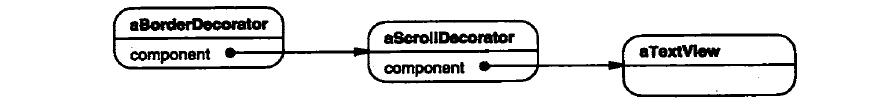
有时我们希望给某个对象而不是整个类添加一些功能。例如，一个图形用户界面工具箱允许你对任意一个用户界面组件添加一些特性，例如边框，或是一些行为，例如窗口滚动。

使用继承机制是添加功能的一种有效途径，从其他类继承过来的边框特性可以被多个子类的实例所使用。但这种方法不够灵活，因为边框的选择是静态的，用户不能控制对组件加边框的方式和时机。

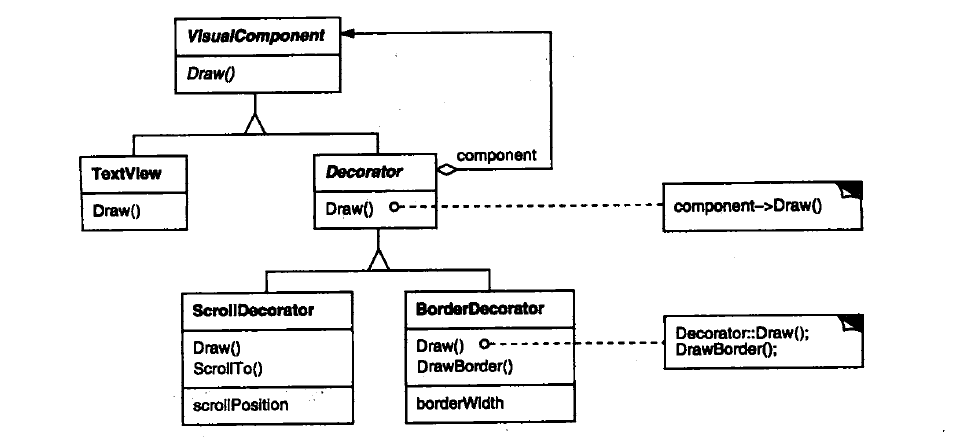
一种较为灵活的方式是将组件签入另一个对象中，由这个对象添加边框。我们称这个嵌入的对象为装饰。这个装饰与它所装饰的组件接口一致，因此它对使用该组件的客户透明。它将客户请求转发给该组件，并且可能在转发前后执行一些额外的动作，例如画一个边框。透明性使得你可以递归的嵌套多个装饰，从而可以添加任意多的功能。

例如，假定有一个对象TextView，它可以在窗口中显示文本。缺省的TextView没有滚动条，因为我们可能有时并不需要滚动条。当需要滚动条时，我们可以用ScrollDecorator添加滚动条。如果我们还想在TextView周围添加一个粗黑边框，可以使用BorderDecorator添加。因此只要简单地将这些装饰和TextView进行组合，就可以达到预期的效果。

下面的对象图展示了如何将一个TextView对象与BorderDecorator以及ScrollDecorator对象组装起来产生一个具有边框和滚动条的文本显示窗口。



ScrollDecorator和BorderDecorator类都是Decorator类的子类。Decorator类是一个可视组件的抽象类，用于装饰其他可视组件。如下图所示



VisualComponent是一个描述可视对象的抽象类，它定义了绘制和事件处理的接口。注意Decorator类怎样将绘制请求简单地发送给它的组件，以及Decorator的子类如何扩展这个操作。

Decorator的子类为特定功能可以自由添加一些操作。例如，如果其他对象知道界面中恰好有一个ScrollDecorator对象，这些对象就可以用ScrollDecorator对象的ScrollTo操作滚动这个界面。这个模式有一点很重要，它使得在VisualComponent可以出现的任何地方都可以有装饰。因此，客户通常不会感觉到装饰过的组件与未装饰组件之间的差异，也不会与装饰产生任何依赖。

结构图

