**Adapter（适配器）**

意图

将一个类的接口转换成客户希望的另外一个接口。Adapter模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

动机

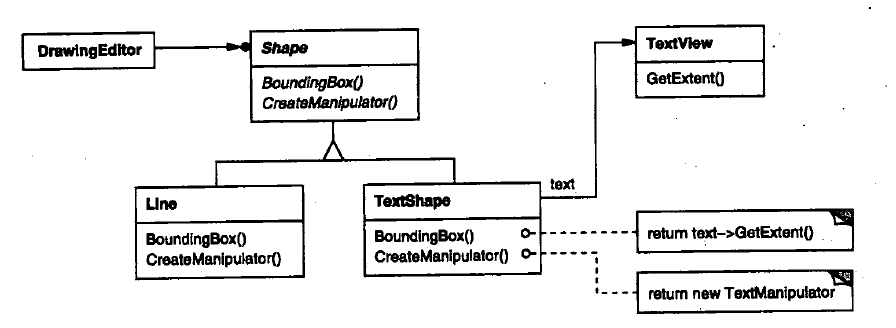
有时，为复用而设计的工具箱不能够被复用的原因仅仅是因为它的接口与专业应用领域所需要的接口不匹配。

例如，有一个绘图编辑器，这个编辑器允许用户绘制和排列基本图元生成图片和图表。这个绘图编辑器的关键抽象是图形对象。图新对象有一个可编辑的形状，并可以绘制自身。图形对象的接口由一个称为Shape的抽象类定义。绘图编辑器为每一种图形对象定义了一个Shape的子类：LineShape类对应于直线，PolygonShape类对应于多边形，等等。

像LineShape和PolygonShape这样的基本几何图形的类比较容易实现，这是由于它们的绘图和编辑功能本来就很有限。但是对于可以显示和编辑正文的TextShape子类来说，实现相当困难，因为即使是基本的正文编辑也要涉及到复杂的屏幕刷新和缓冲区管理。同时，成品的用户界面工具箱可能已经提供了一个复杂的TextView类用于显示和编辑正文。理想的情况是我们可以复用这个TextView类以实现TextShape类，但是工具箱的设计者当时并没有考虑Shape的存在，因此TextView和Shape对象不能互换。

一个应用可能会有一些类具有不同的接口并且这些互不兼容，在这样的应用中象TextView这样已经存在并且不相干的类如何协同工作呢？我们可以改变TextView类使它兼容Shape类的接口，但前提是必须有这个工具箱的源代码。然而即使我们得到了这些源代码，修改TextView也没有什么意义的；因为不应该仅仅为实现一个应用，工具箱就不得不采用一些与特定领域相关的接口。

我们可以不用上面的方法，而定义一个TextShape类，由它来适配TextView的接口和Shape的接口。我们可以用两种方法做这件事：1，继承Shape类的接口和TextView的实现；2，将一个TextView实例作为TextShape的组成部分，并且使用TextView的接口实现TextShape。这两种方法恰恰为对应于Adapter模式的类和对象版本。我们将TextShape称之为适配器Adapter。



上面的类图说明了对象适配器实例。它说明了在Shape类中声明的BoundingBox请求如何被转换成在TextView类中定义的GetExtent请求。由于TextShape将TextView接口与Shape的接口进行了匹配，因此绘图编辑器就可以复用原先并不兼容的TextView类。

Adapter时常还要负责提供那些被匹配的类所没有提供的功能，上面的类图中说明了适配器如何实现这些职责。由于绘图编辑器允许用户交互的将每一个Shape对象“拖动”到一个新的位置，而TextView设计中没有这种功能。我们可以实现TextShape类的CreateManipulator操作，从而增加这个缺少的功能，这个操作返回相应的Manipulator子类的一个实例。

Manipulator是一个抽象类，它所描述的对象知道如何驱动Shape类响应相应的用户输入，例如将图形拖动到一个新的位置。对应于不同形状的图形，Manipulator有不同的子类；例如子类TextManipulator对应于TextShape。TextShape通过返回一个TextManipulator实例，增加了TextView中缺少而Shape需要的功能。

结构图

