**Mediator（中介者）**

意图

用一个中介对象来封装一系列的对象交互。中介者使各个对象不需要显示地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。

动机

面向对象设计鼓励将行为分布到各个对象中。这种分布可能会导致对象间有很多连接。在最坏的情况下，每一个对象都知道其他所有对象。

虽然将一个系统分割成许多对象通常可以增强可复用性，但是对象间互相连接的激增又会降低其可复用性。大量的相互连接使得一个对象似乎不太可能在没有其他对象的支持下工作—系统表现为一个不可分割的整体。而且，对系统的行为进行任何较大的改动都十分困难，因为行为被分布在许多对象中。结果是，你可能不得不定义很多子类以定制系统的行为。

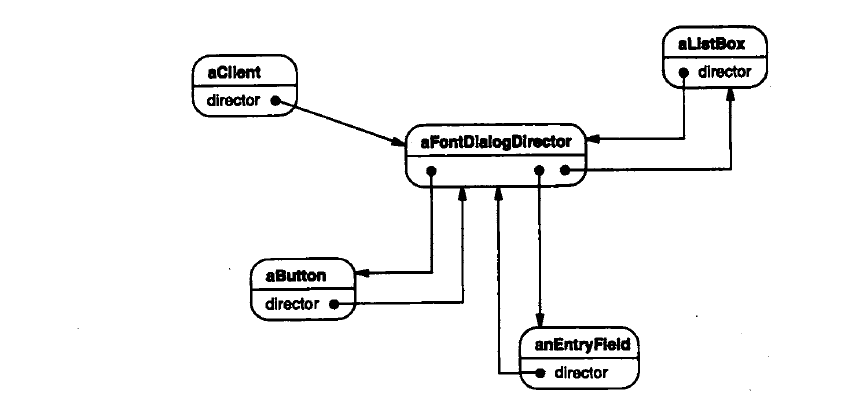
例如，考虑一个图形用户界面中对话框的实现。对话框使用一个窗口来展现一系列的窗口组件，如按钮、菜单和输入域等。

通常对话框中的窗口组件间存在依赖关系。例如，当一个特定的输入域为空时，某个按钮不能使用；在称为列表框的一列选项中选择一个表目可能会改变一个输入域的内容；反过来，在输入域中输入正文可能会自动的选择一个或多个列表框中的相应的表目；一旦正文出现在输入域中，其他一些按钮可能就变得能够使用了，这些按钮允许用户做一些操作，比如改变或删除这些正文所指的东西。

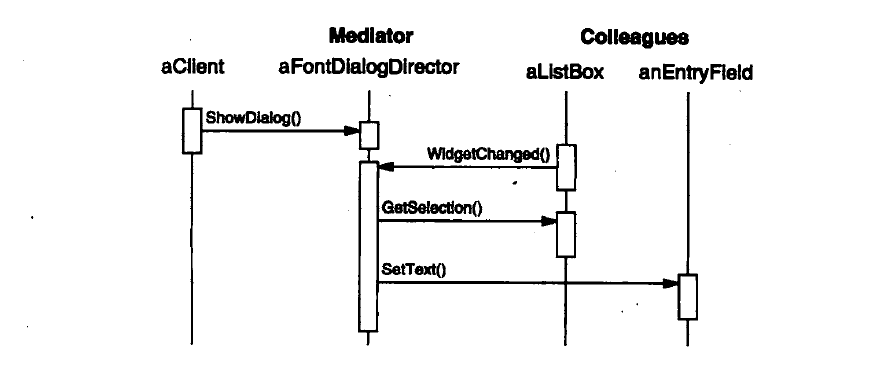
不同的对话框会有不同的窗口组件间的依赖关系。因此即使对话框显示相同类型的窗口组件，也不能简单地直接重用已有的窗口组件类；而必须定制它们以反映特定对话框的依赖关系。由于涉及很多个类，用逐个生成子类的办法来定制它们会很冗长。

可以通过将集体行为封装在一个单独的中介者对象中以避免这个问题。中介者负责控制和协调一组对象间的交互。中介者充当一个中介以使组中的对象不再相互显示引用。这些对象仅知道中介者，从而减少了相互连接的数目。

例如，FontDialogDirector可作为一个对话框中的窗口组件间的中介者。FontDialogDirector对象知道对话框中的各窗口组件，并协调它们之间的交互。它充当窗口组件间通信的中转中心，如下图所示。



下面的交互图说明了各对象如何协作处理一个列表框中选项的变化。

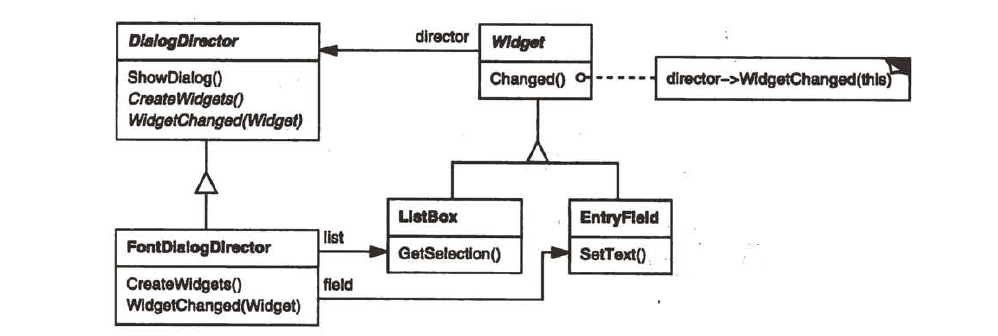


下面一系列事件使一个列表框的选择被传送给一个输入域：

1. 列表框告诉它的操作者它被改变了。
2. 导控者从列表框中得到选中的选择项。
3. 导控者将该选择项传递给入口域。
4. 现在入口域已有正文，导控者使得用于发起一个动作的某些按钮可用。

注意导控者是如何在对话框和入口域间进行中介的。窗口组件间的通信都通过导控者间接地进行。它们不必互相知道；它们仅需知道导控者。而且，由于所有这些行为都局部于一个类中，只要扩展或替换这个类，就可以改变和替换这些行为。

这里展示的是FontDialogDirector抽象怎样被集成到一个类库中，如下图所示。



DialogDirector是一个抽象类，它定义了一个对话框的总体行为。客户调用ShowDialog操作将对话框显示在屏幕上。CreateWidgets是创建一个对话框的窗口组件的抽象操作。WidgetChanged是另一个抽象操作；窗口组件调用它来通知它的导控者它们被改变了。DialogDirector的子类将重定义CreateWidgets以创建正确的窗口组件，并重定义WidgetChanged以处理其变化。

结构图

