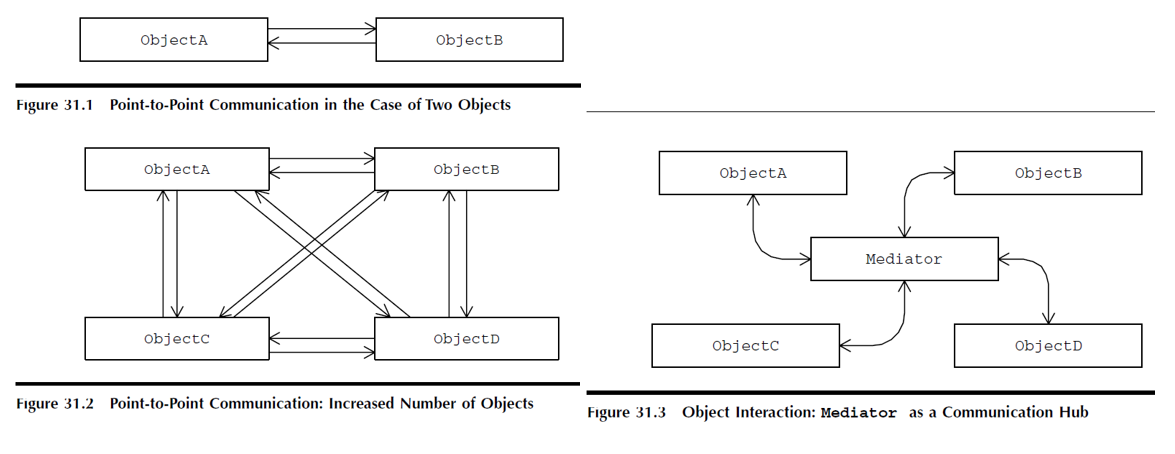
# Patrón Mediador

## Introducción

En general, las aplicaciones orientadas a objetos consisten en un conjunto de objetos que interactúan entre sí con el fin de proporcionar un servicio. Esta interacción puede dirigirse directamente (punto a punto) siempre que el número de objetos que se refieren directamente sea muy bajo. La Figura 31.1 muestra este tipo de interacción directa donde ObjectA y ObjectB se relacionan directamente entre sí. A medida que aumenta el número de objetos, este tipo de interacción directa puede conducir a un complejo laberinto de referencias entre objetos (Figura 31.2), lo que afecta la mantenibilidad de la aplicación. Además, tener un objeto que se refiera directamente a otros objetos reduce en gran medida la posibilidad de reutilizar estos objetos debido a un acoplamiento más alto.

En tales casos, el patrón de Mediador puede usarse para diseñar un modelo de comunicación controlado y coordinado para un grupo de objetos, eliminando la necesidad de que los objetos se referencien directamente entre sí (Figura 31.3).



## Objetivo del Patrón Mediador

El patrón Mediador sugiere abstraer todos los detalles de interacción de objetos en una clase separada, conocida como **Mediador**, con conocimiento sobre el grupo de objetos que interactúan. Todos los objetos del grupo siguen siendo responsables de ofrecer el servicio para el que están diseñados, pero los objetos no interactúan entre sí directamente con este propósito. La interacción entre dos objetos diferentes se enruta a través de la clase Mediator. Todos los objetos envían sus mensajes al mediador. Mediator luego envía mensajes a los objetos apropiados según los requisitos de la aplicación.

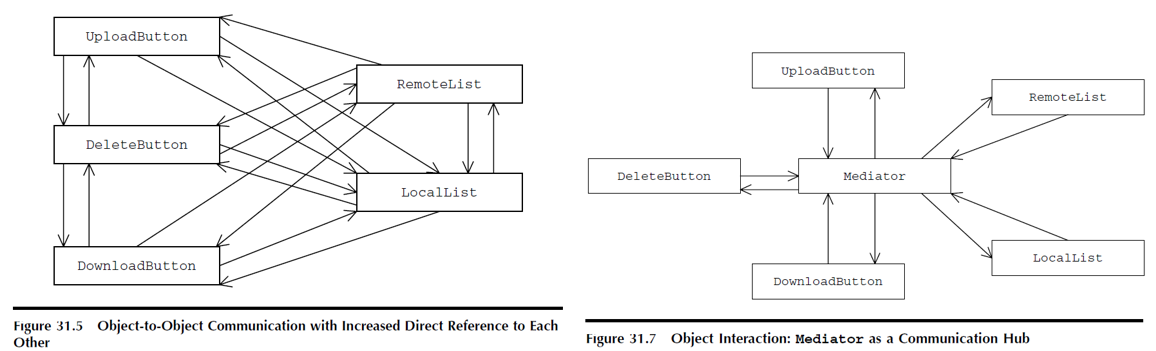
## ventajas principales:

* Con todas las interacciones movidas a un objeto separado (mediador), se hace más fácil alterar el comportamiento de las interrelaciones de objetos, al reemplazar el mediador con una de sus subclases con funcionalidad extendida o alterada.
* Mover dependencias interobjeto fuera de objetos individuales resulta en reutilización de objetos mejorada.
* Debido a que los objetos no necesitan referirse directamente entre sí, los objetos pueden ser probados en unidades más fácilmente.
* El bajo grado de acoplamiento resultante permite que las clases individuales sean modificadas sin afectar a otras clases.

## MEDIADOR VERSUS FACHADA

|  |  |
| --- | --- |
| MEDIADOR | FACHADA |
| Un mediador se utiliza para abstraer la funcionalidad necesaria de un grupo de objetos con el objetivo de simplificar la interacción del objeto. | Una fachada se usa para abstraer la funcionalidad requerida de un subsistema de componentes, con el objetivo de proporcionar una interfaz simplificada de nivel superior. |
| Todos los objetos interactúan entre sí a través del Mediador. El grupo de objetos conoce la existencia del Mediador. | Los clientes usan la fachada para interactuar con los componentes del subsistema. La existencia de la fachada no es conocida por los componentes del subsistema. |
| Debido a que el Mediador y todos los objetos que están registrados con él pueden comunicarse entre sí, la comunicación es bidireccional. | Los clientes pueden enviar mensajes (a través de la fachada) al subsistema pero no al revés, haciendo que la comunicación sea unidireccional. |
| Se puede suponer que un mediador permanece en el medio de un grupo de objetos que interactúan.  El uso de un Mediador permite que la implementación de cualquiera de los objetos interactivos se cambie sin ningún impacto en los otros objetos que interactúan con él solo a través del Mediador. | Una fachada se encuentra entre un objeto cliente y el subsistema.  El uso de una fachada permite que la implementación del subsistema se cambie por completo sin ningún impacto en sus clientes, siempre que los clientes no tengan acceso directo a las clases del subsistema. |
| Al subclasificar el Mediador, se puede ampliar el comportamiento de las interrelaciones de objetos. | Al subclasificar la fachada, se puede cambiar la implementación de la interfaz de nivel superior. |

Ejemplo: Servidor FTP

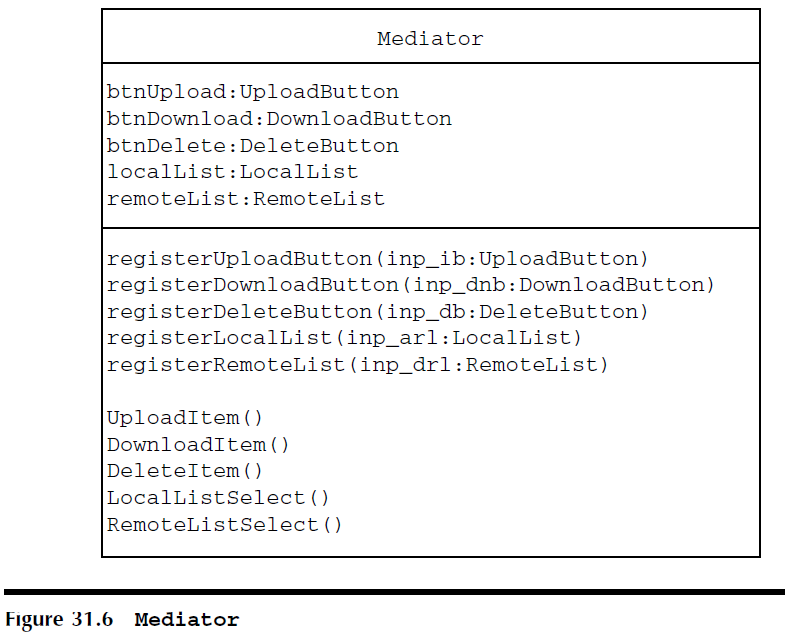


## Uso del mediador por parte del cliente

El cliente (Listado 31.2) crea una instancia del Mediador. cada vez que se crea un objeto de IU, el cliente le pasa la instancia de Mediator. El objeto UI se registra con esta instancia del Mediador.

## Objetos de interfaz de usuario: interacción del mediador

Debido a que todos los detalles de interacción del objeto se eliminan de los objetos individuales de Interfaz de Usuario al objeto Mediador, el método processEvent de cada uno de estos objetos UI se reduce a una simple llamada a un método Mediador apropiado (Listado 31.3). La Figura 31.7 muestra la interacción del objeto UI después de aplicar el patrón Mediator.



La interacción directa entre los diferentes objetos Item y Category se puede eliminar moviendo los detalles de interacción del objeto fuera de las clases Item y Category a una clase Mediator separada (Figura 31.10). El Mediator puede diseñarse con los siguientes dos conjuntos de métodos:

* Un conjunto de métodos para permitir que diferentes objetos Item y Category se registren con el Mediador.
* Un conjunto de métodos para agregar y eliminar elementos. El Mediador es responsable de implementar las interacciones entre diferentes objetos como parte de estos métodos.

Ejemplo 2

El Mediador puede mantener la asociación Item-Category en la variable de instancia item-CatAssoc. Los objetos objeto no necesitan referirse directamente a objetos de categoría. Por lo tanto, un objeto Item no necesita mantener la lista de Categorías a las que pertenece y viceversa. Del mismo modo, las operaciones Item y Delete no requieren la implementación de las operaciones de agregar y eliminar. El método de ejecución de AddCommand y DeleteCommand Commandobjects se reduce a una llamada a los métodos addItem y deleteItem Mediator, respectivamente.

