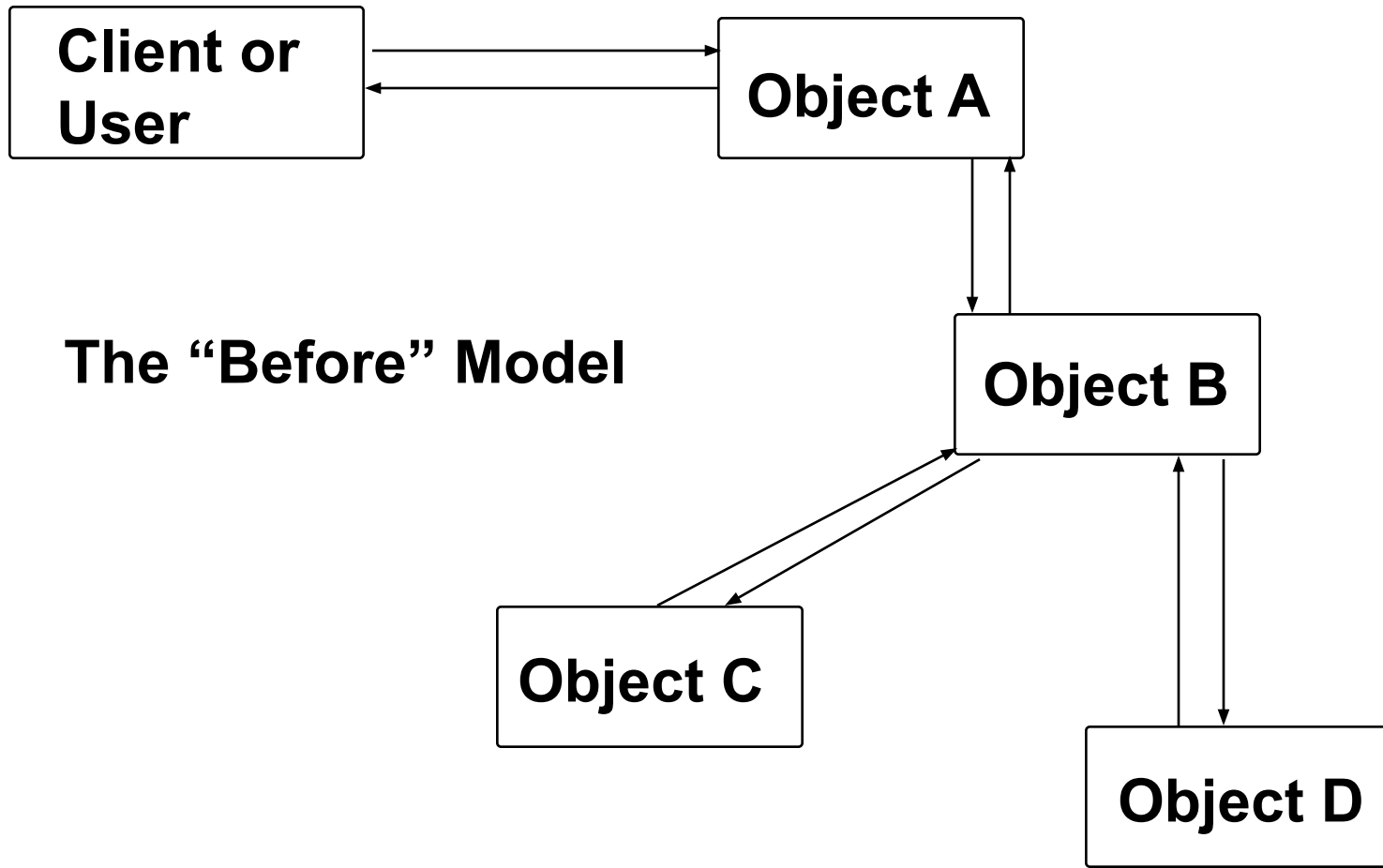


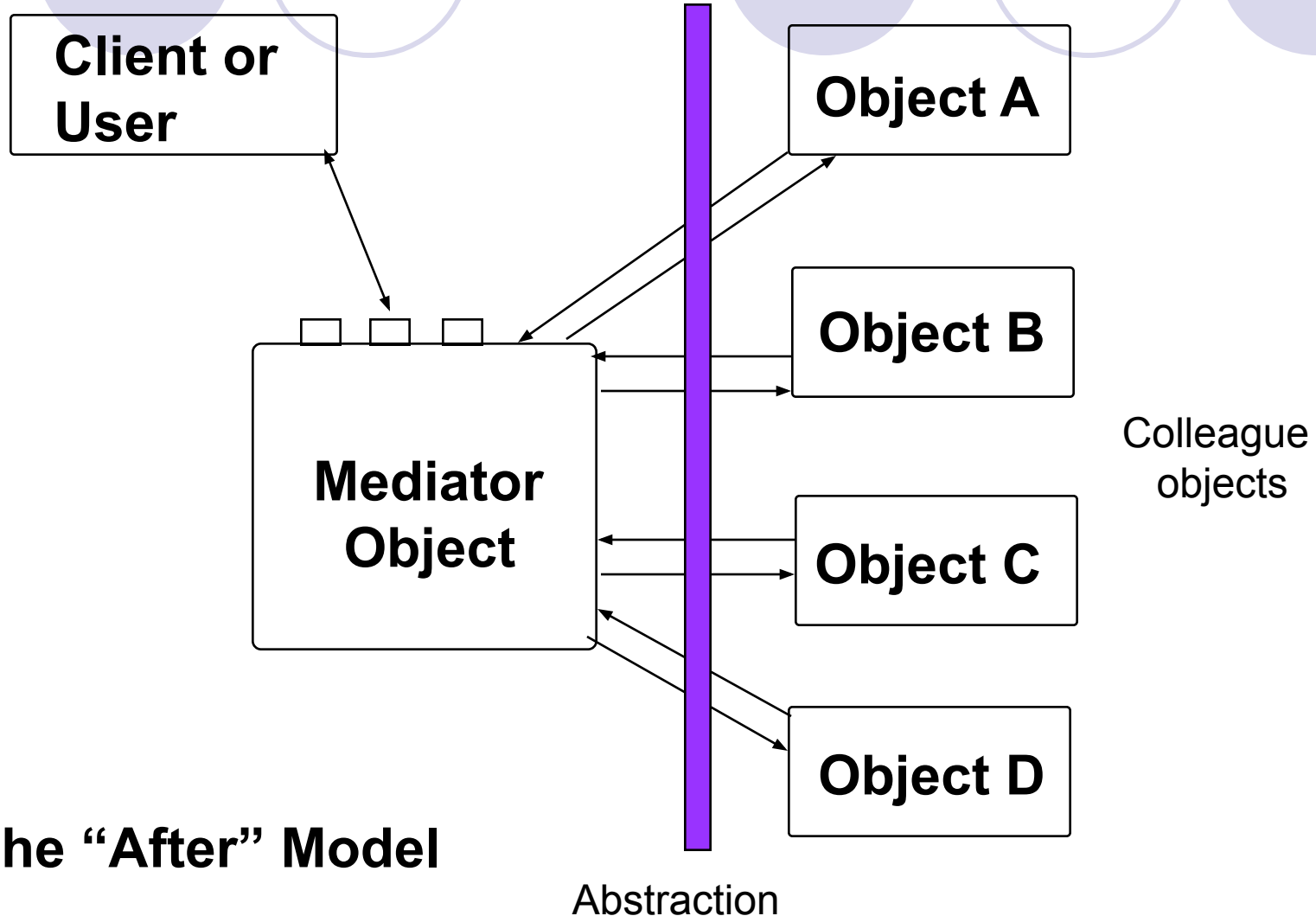
Mediator Pattern: Definition

- Mediator object coordinates communication between interacting objects.
- Interacting objects work with and through a Mediator rather than each other directly.
- The Mediator pattern is also known as a “glue” pattern.
- The Mediator pattern decouples the individual tasks.
 - Each object only knows its own input and output.
 - This way, each object does not depend on the other objects around it.

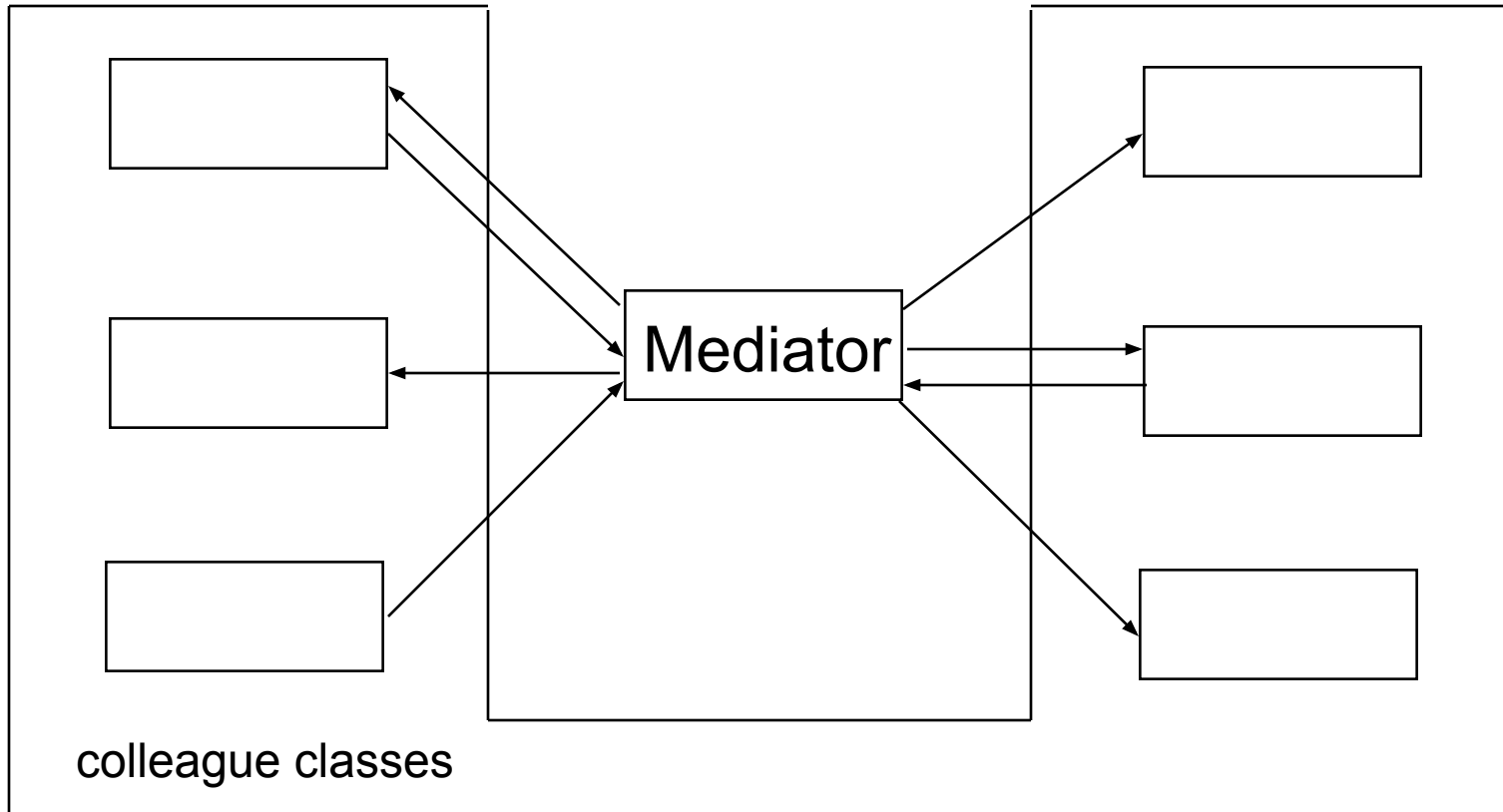
Mediator Pattern: Illustration



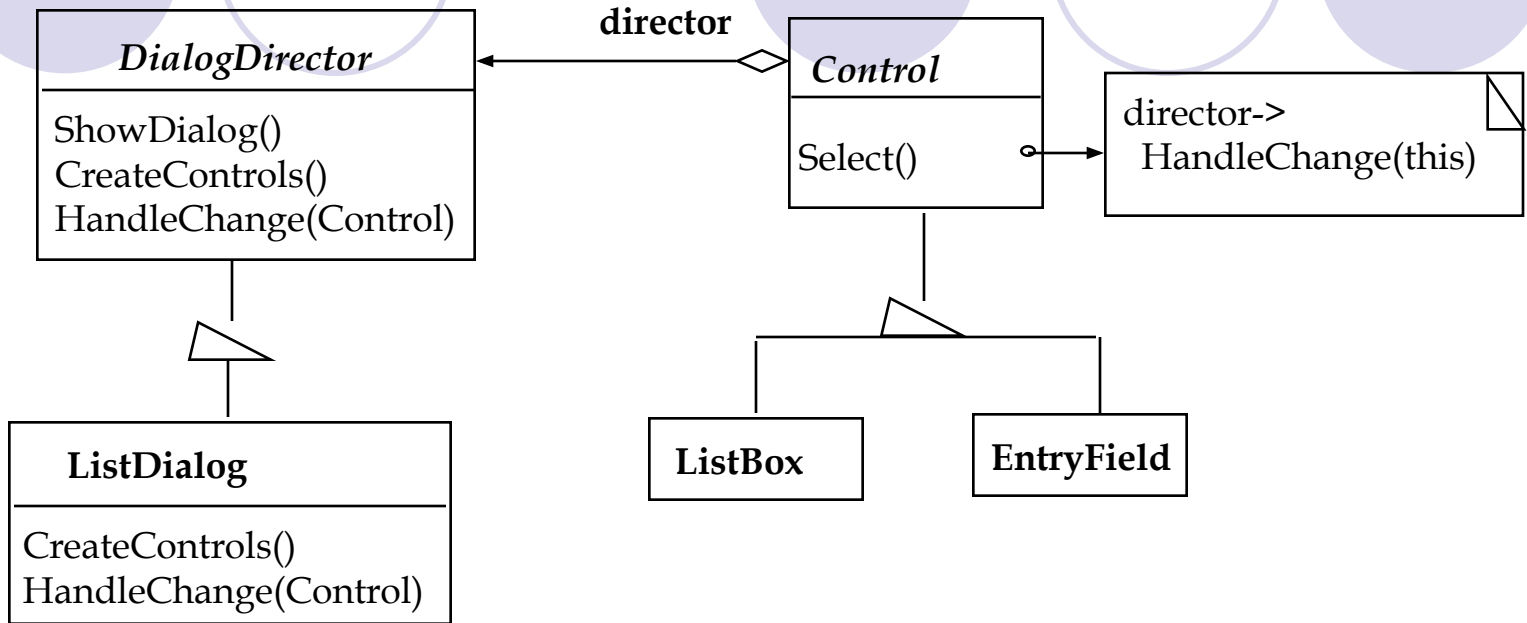
Mediator Pattern: More Illustration



Mediator Pattern: Structure

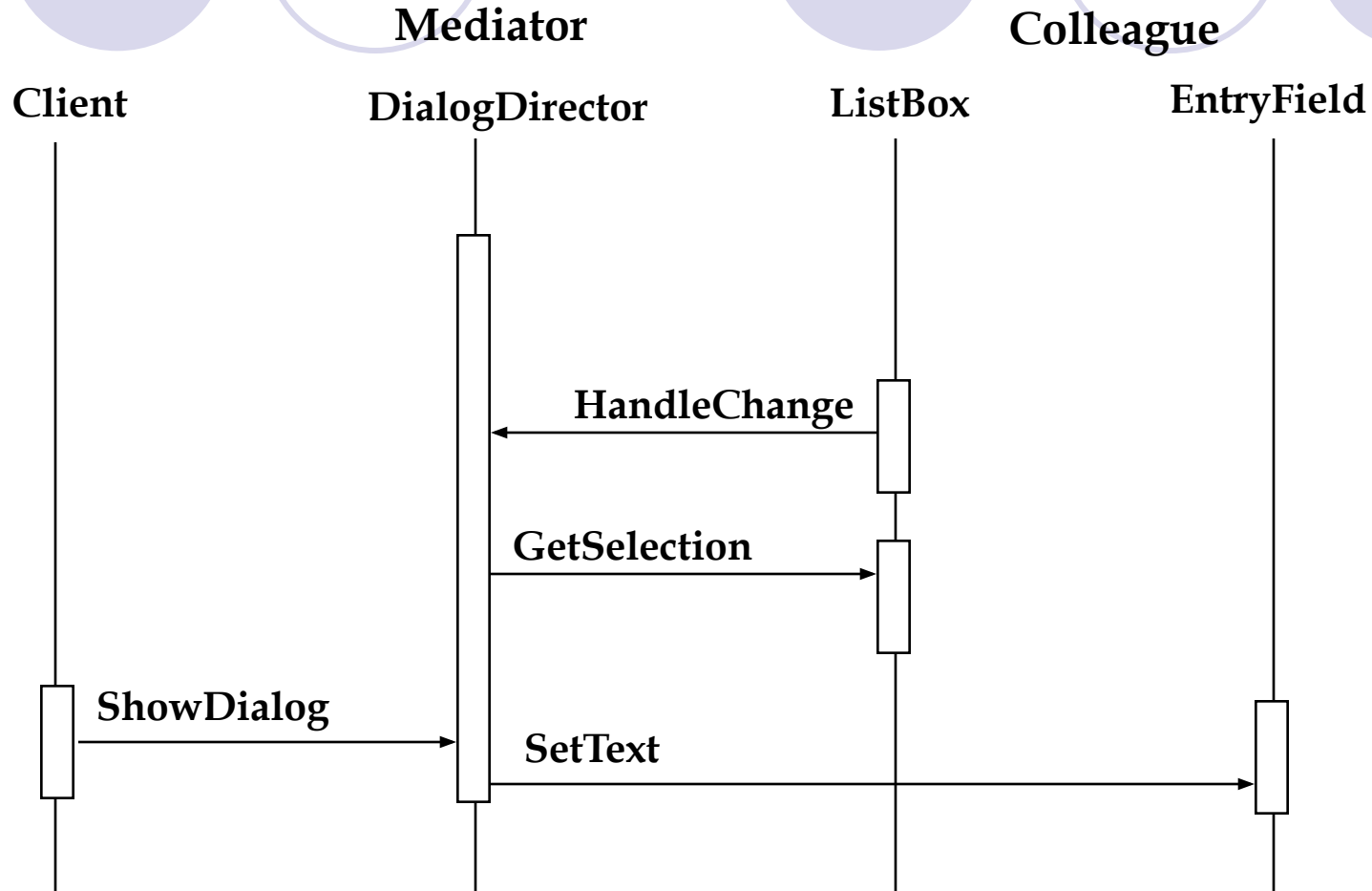


Example: Graphic UI



- **DialogDirector** is an abstract class that defines the overall infrastructure of a dialog. It keeps a reference to the window that presents the dialog.
- Clients call the `ShowDialog` operation to display the dialog on the screen.
- `CreateControls` and `HandleChange` are abstract operations.
- **DialogDirector** subclasses override `CreateControls` to create the proper controls and `HandleChange` to handle the changes.

Graphic UI Example: Interaction Diagram



Participants



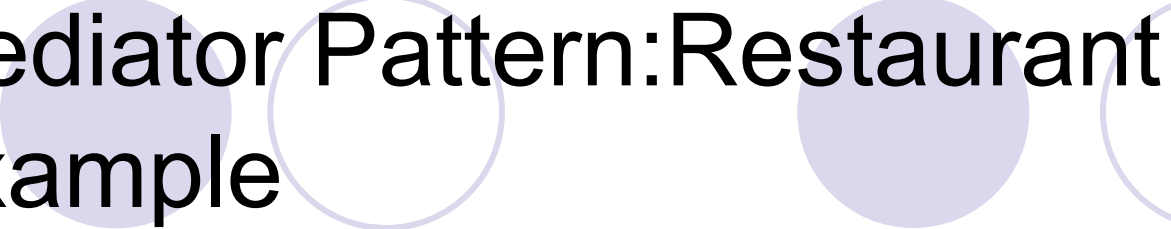
- Mediator (DialogDirector)
 - implements cooperative behavior by coordinating Colleague objects.
 - stores state.
- Colleague classes (ListBox, EntryField)
 - implements pieces of the cooperative behavior.

Collaborations

The title 'Collaborations' is positioned on the left. To its right, there are two groups of three circles each. The first group consists of a solid light purple circle, a hollow light purple circle, and another solid light purple circle. The second group, further to the right, also consists of a solid light purple circle, a hollow light purple circle, and another solid light purple circle.

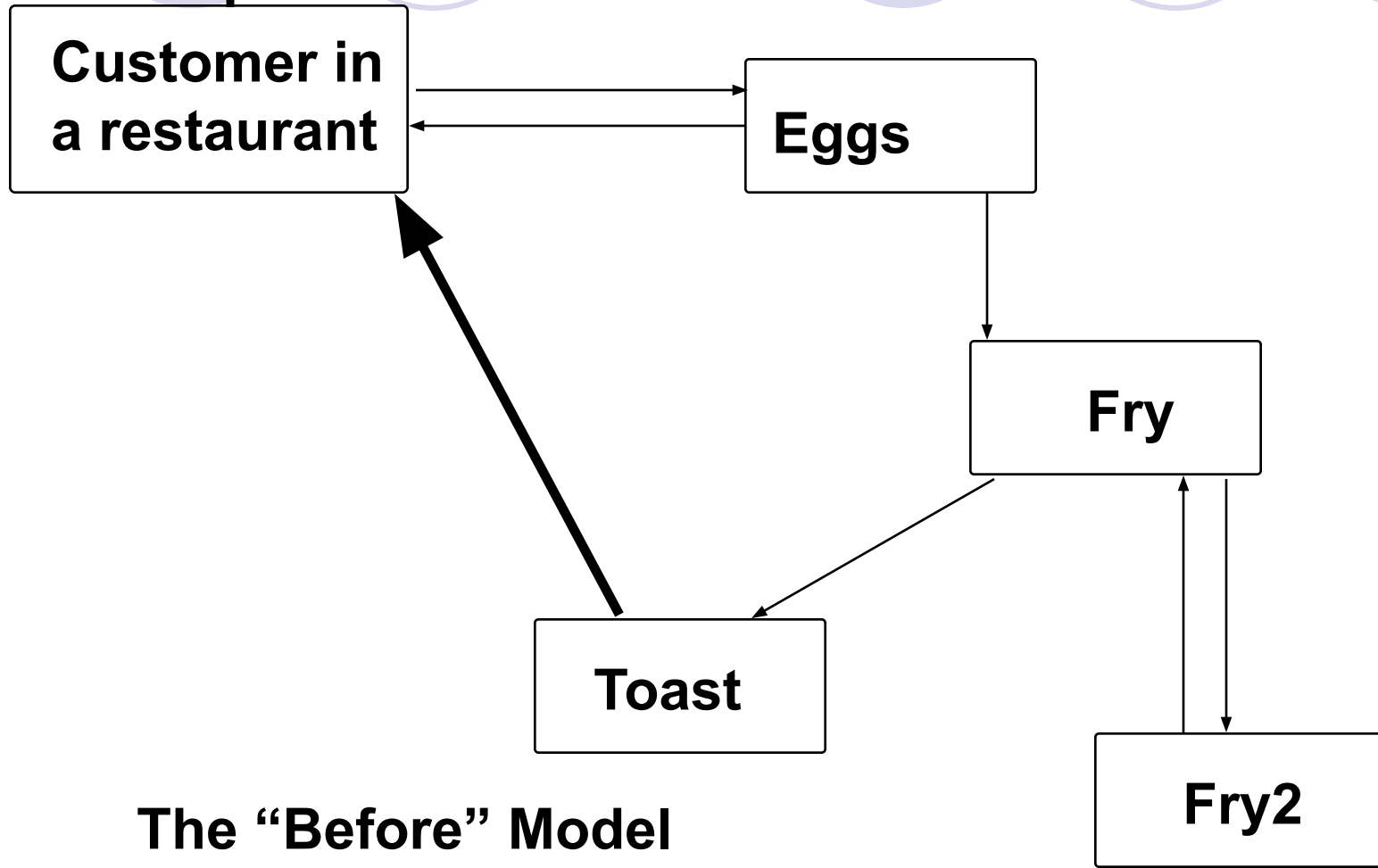
- Colleagues send/receive requests to/from a Mediator object.
- The Mediator implements the cooperative behavior by routing requests between the appropriate Colleague(s).

Mediator Pattern: Restaurant Example

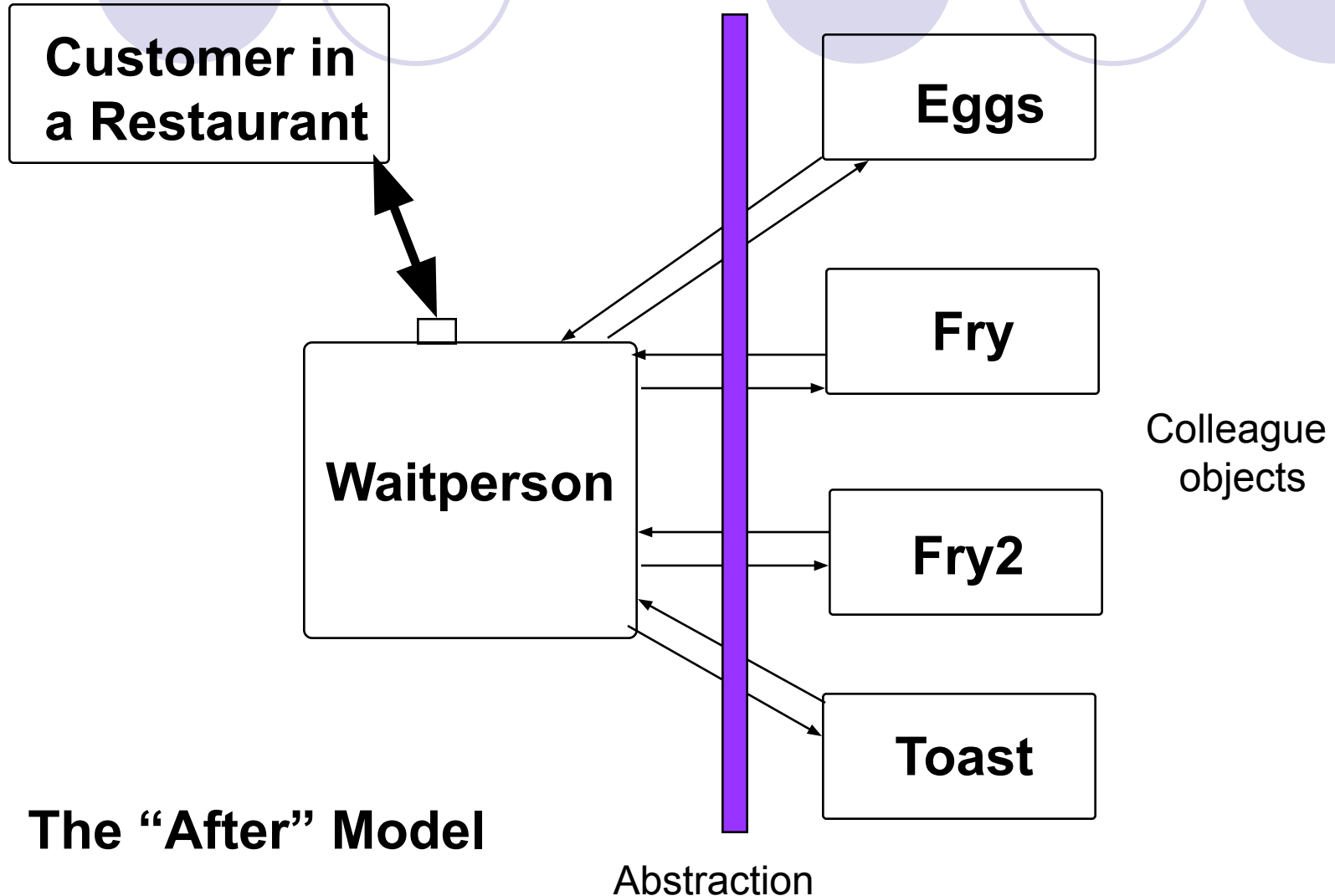


- It is time for breakfast. A customer places an order and gives it to the egg cooker.
- The egg cooker does his job and then moves the order to the fry cook for meat and hash browns. The fry cook sends the plate on, and so forth.
- When the order has finished processing in the kitchen the customer's name is called and the plate is given to the customer.
- This scheme is inflexible because you cannot change the structure of the kitchen and you cannot add any other steps to the cooking processes.

Mediator Pattern: Restaurant Example



Mediator Pattern: The Solution



Mediator Pattern: More on The Solution

- Each of the objects knows only of the Mediator existence . Each performs its task and returns its finished product to the Mediator.
- Flexibility is added because:
 - Each individual object is free to change its own methods without effecting the others and without the other's knowledge. This allows for dynamic binding and polymorphism.
 - An additional process may be added. This would require a change in the Mediator object.

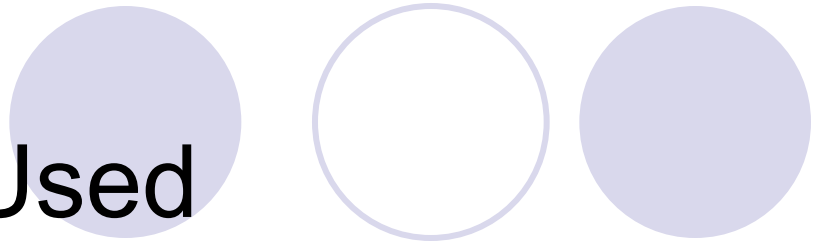
Mediator Pattern: Advantages

- Moves complexity out of the problem domain solution (mainline routines). Abstraction hides some non-essential implementation details.
- Changes or improvements to the underlying functions and objects do not require changes to client code.
- Frees up individual colleague objects by decoupling them.
- Enhances the chance for extensibility in the colleague objects since changes to individual colleagues do not ripple through to other colleagues and affect overall dependencies.
- Different Mediator objects can provide a different set of services to the client by reusing colleague objects.

Mediator Pattern: Disadvantages

- If the Mediator becomes too complex, extensibility of the Mediator may be limited without significant rework. Do not allow the Mediator to become a “special case” handler.
- If the degree of complexity that you are moving from the Colleague into the Mediator is small then extra overhead of Mediator development may not be justified.
- Decision made by the Mediator have to “jive” with what the Mainline routines expect. In other words, “Can you trust your Mediator’s decisions?”

Mediator Pattern: When It Should be Used

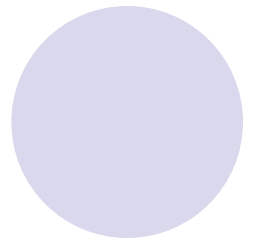
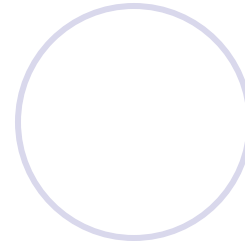
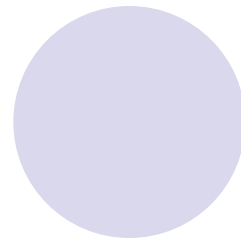
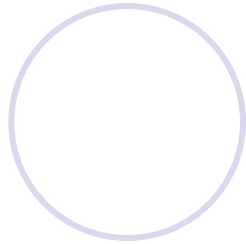
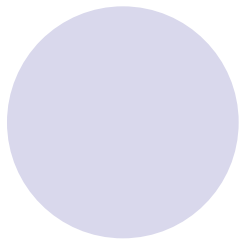


- When we have a group of colleagues that interact in complex but well-defined ways.
- When the order of individual colleague operations might change.
- Relieve colleague objects from the additional burden of mediation in addition to their regular services.



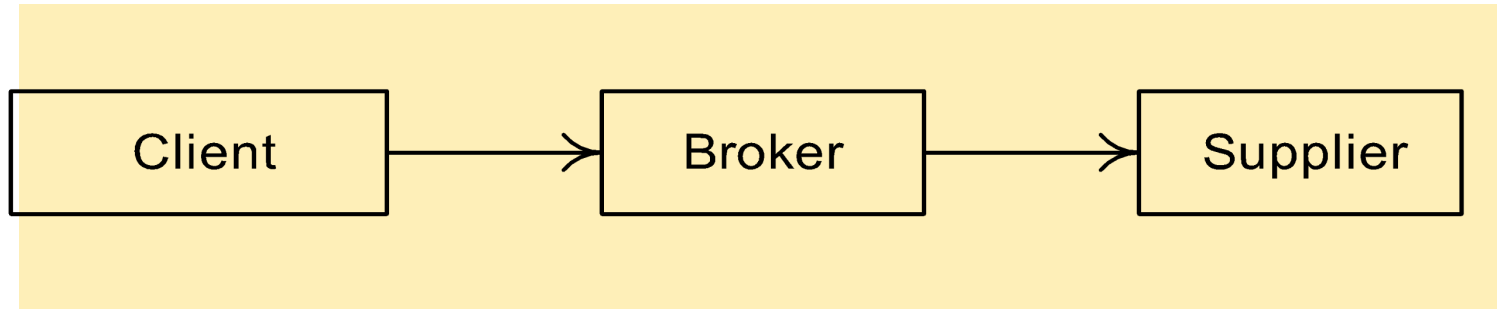
Discussion Questions

- (T) for true or (F) for false
 - () 1. The Mediator pattern replaces many-to-many interactions with one-to-many ones between mediator and colleagues, which are easier to understand, maintain, and extend.
 - () 2. A common use of the Mediator pattern is to keep track of instances of a certain class.
 - () 3. One of the application of the Mediator pattern is to coordinate complex updates.
 - () 4. The Mediator pattern coordinate communication between interacting objects.



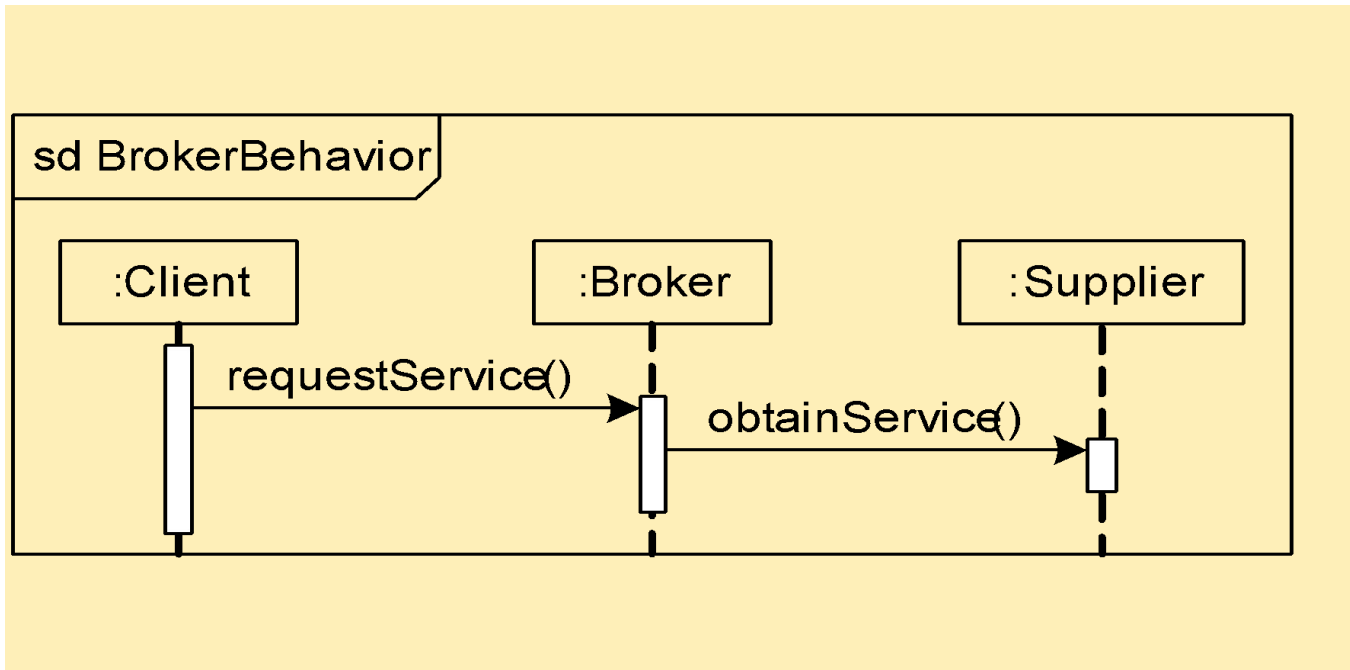
Broker Design Patterns: Façade and Mediator

Broker Pattern Structure



- The Client must access the Broker and the Broker must access the Supplier
- Most Broker patterns elaborate this basic structure

Broker Pattern Behavior

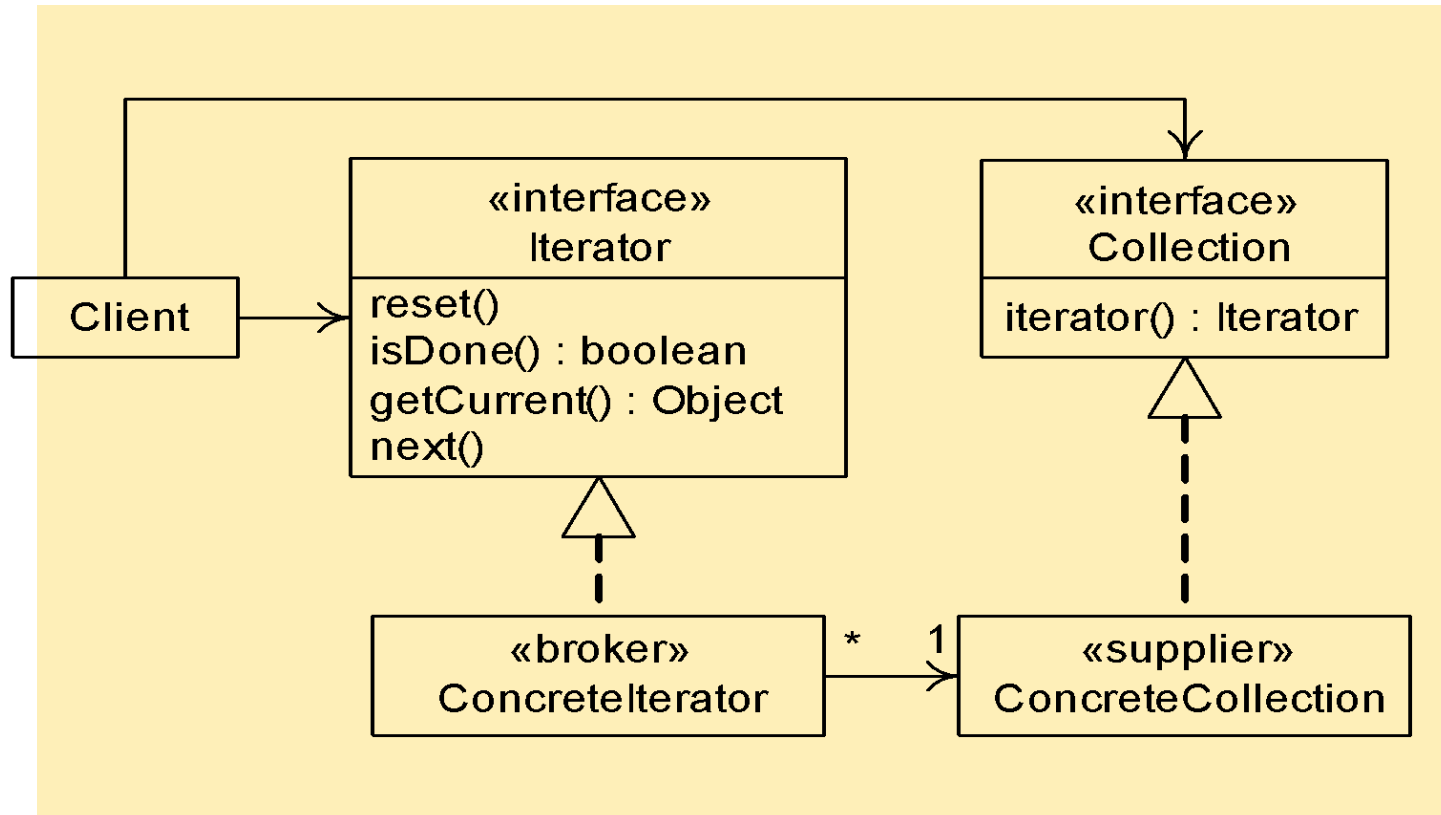




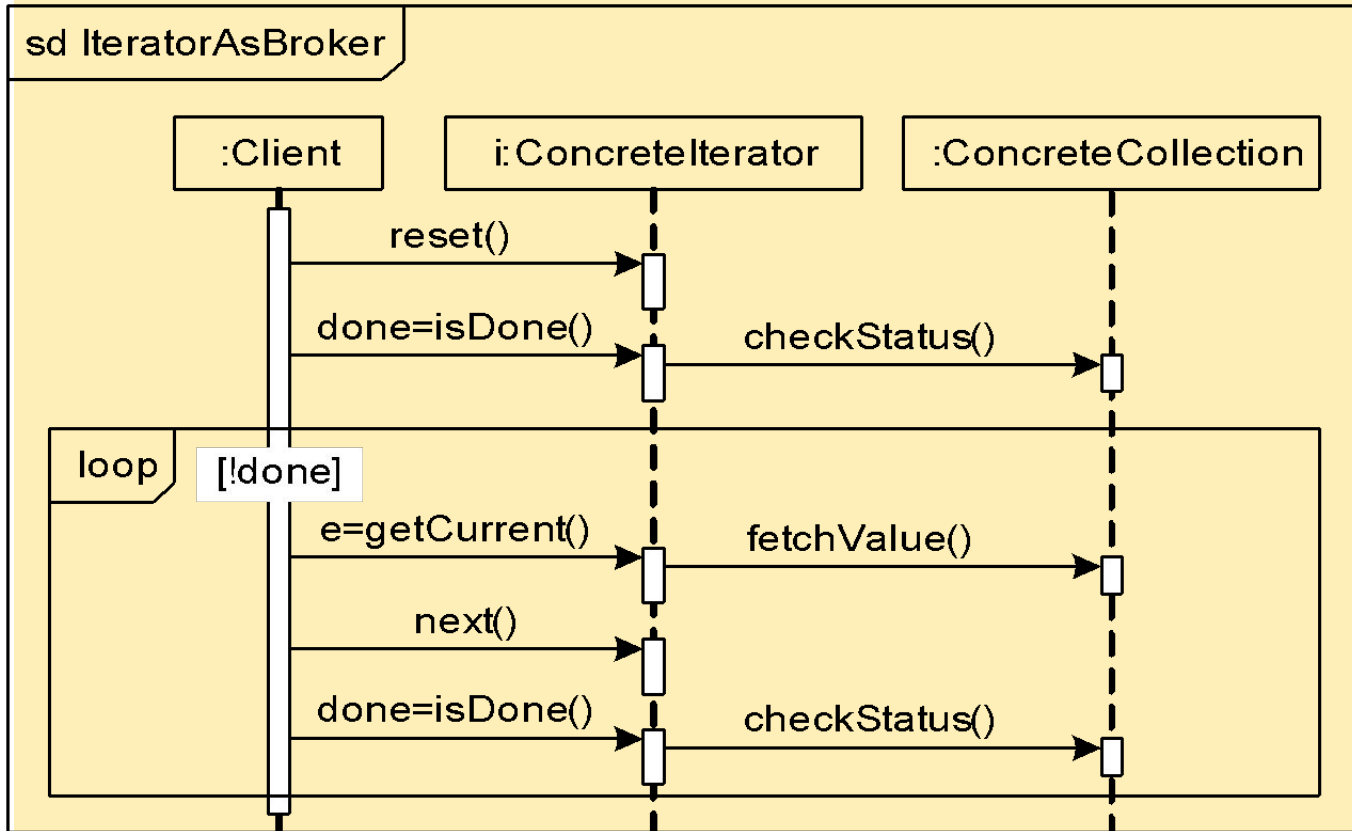
Broker Pattern Advantages

- *Simplify the Supplier*—A Broker can augment the Supplier's services.
- *Decompose the Supplier*—A complex Supplier can offload some of its responsibilities to a Broker.
- *Facilitate Client/Supplier Interaction*—A Broker may present a different interface, handle interaction details, etc.

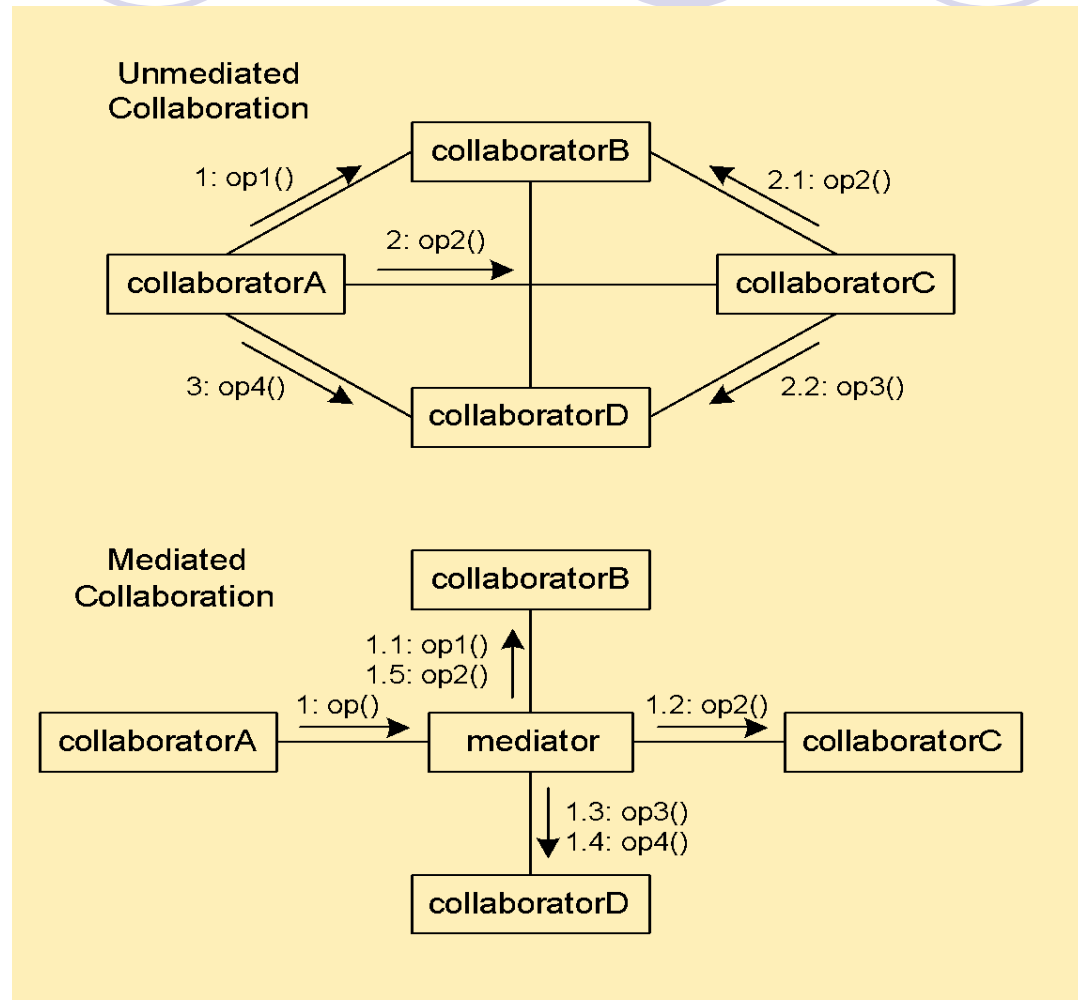
Broker Example: Iterator Form



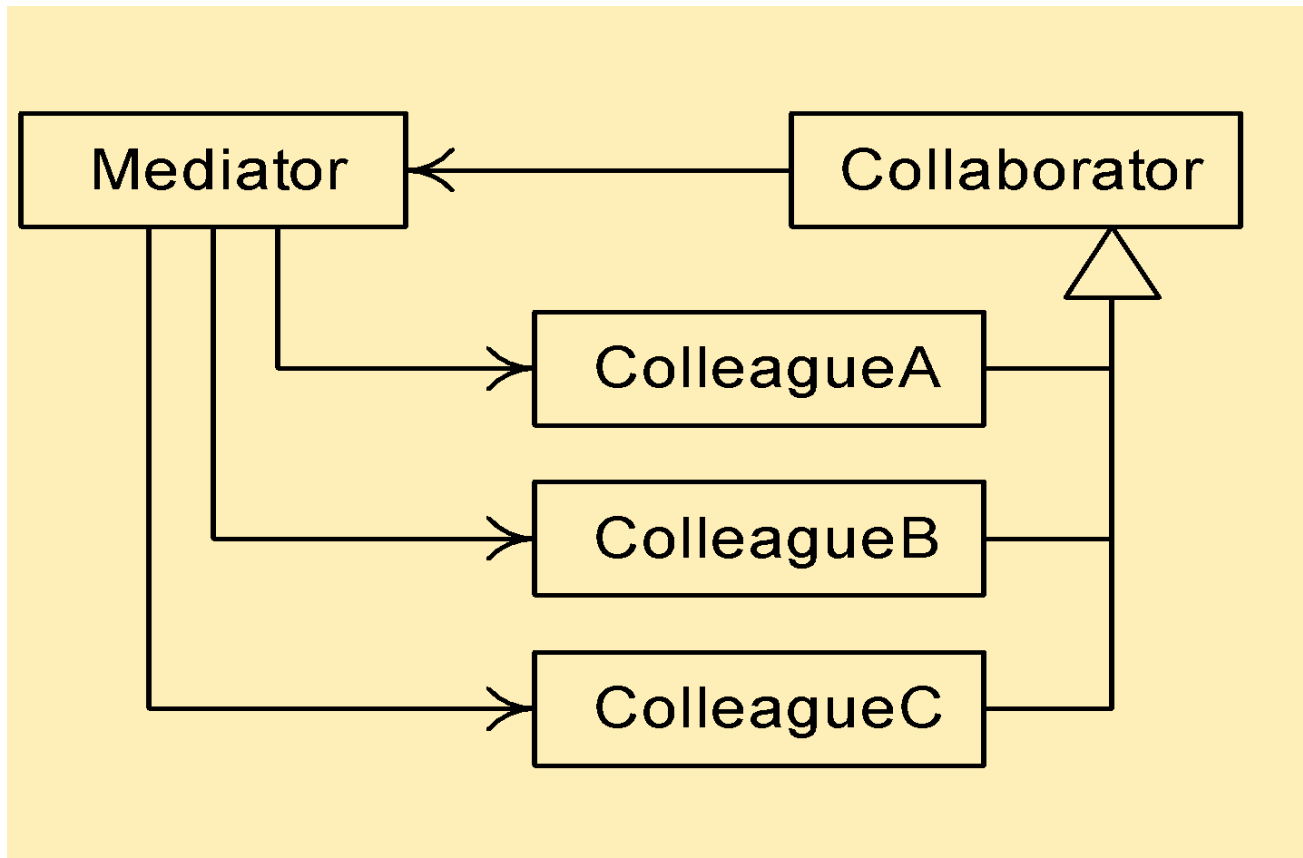
Broker Example: Iterator Behavior



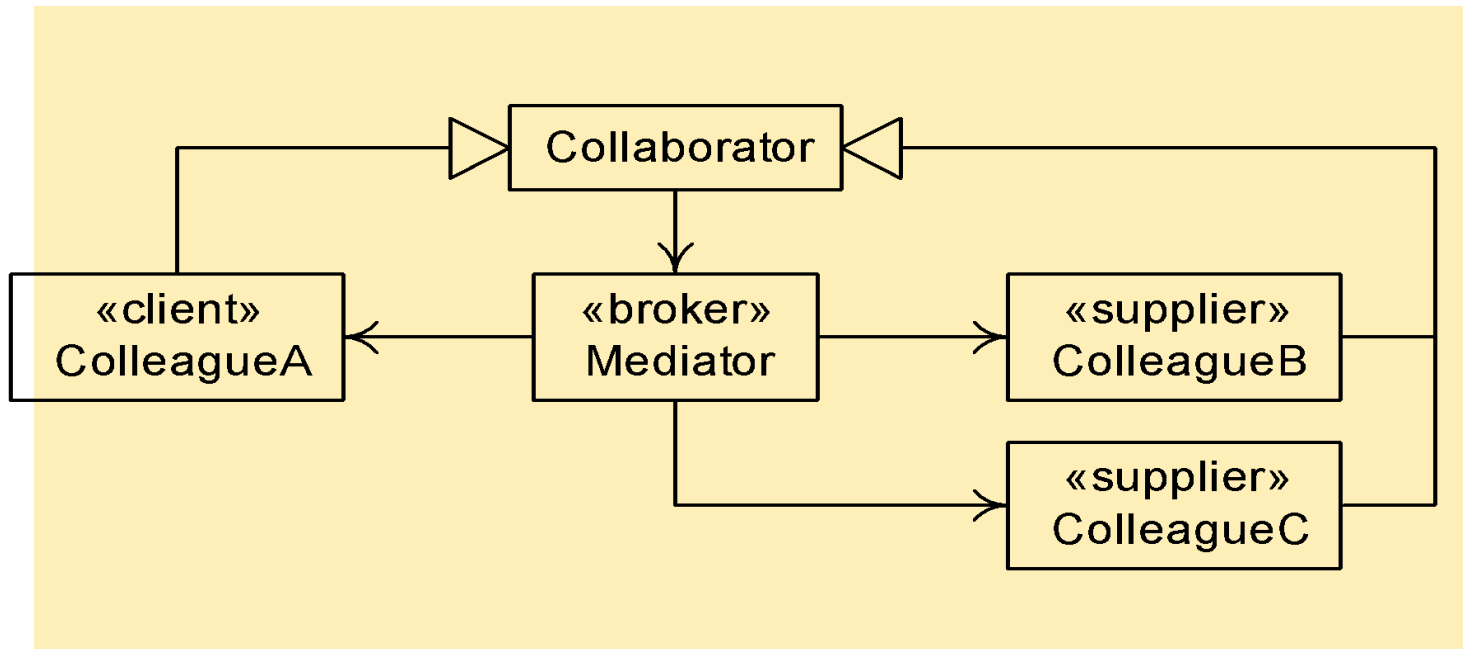
Using a Mediator



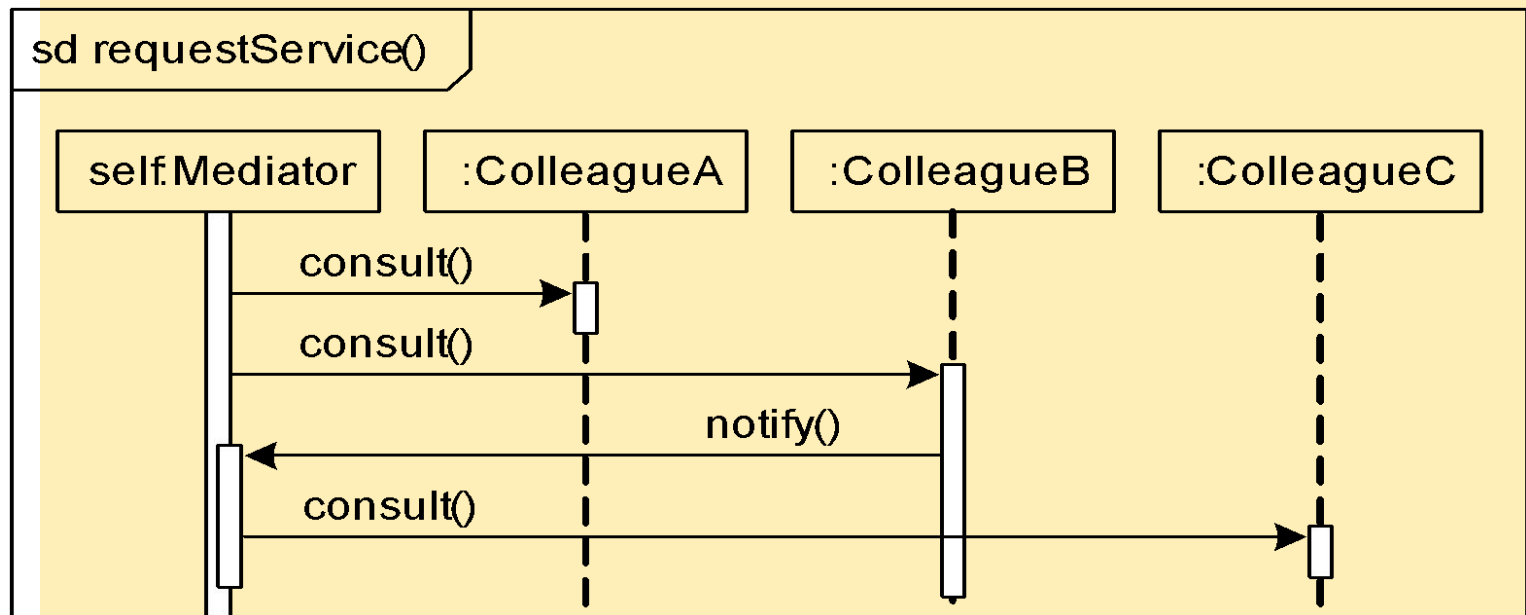
Mediator Pattern Structure



Mediator as a Broker



Mediator Behavior





When to Use a Mediator

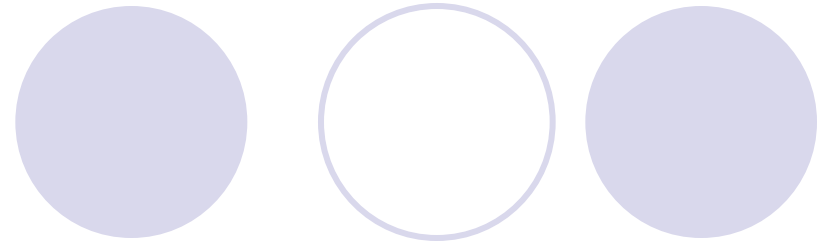
- Use the Mediator pattern when a complex interaction between collaborators must be encapsulated to
 - Decouple collaborators,
 - Centralize control of an interaction, and
 - Simplify the collaborators.
- Using a mediator may compromise performance.

Summary



- Broker patterns use a Broker class to facilitate the interaction between a Client and a Supplier.
- The Façade pattern uses a broker (the façade) to provide a simplified interface to a complex sub-system.
- The Mediator pattern uses a broker to encapsulate and control a complex interaction among several suppliers.

Related Patterns



- **Facade:** "Provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use." In more simplified terms, the facade pattern provides unidirectional communication with subsystem classes. Mediator is multidirectional.

Related Patterns



- **Observer:** Colleagues may communicate with a mediator using the observer pattern. In fact, this is very commonly used in conjunction with the Mediator, as it provides a facility by which the Colleagues may 'subscribe' to the Mediator, and thus alleviates the Mediator of the responsibility of having to know exactly what classes need to be informed of events.

Examples

The word "Examples" is positioned to the left of a series of six circles. The first circle is solid light purple. The second circle is white with a light purple outline. The third circle is solid light purple. The fourth circle is white with a light purple outline. The fifth circle is solid light purple. The sixth circle is white with a light purple outline.

Parliament

- The House of Commons in the Parliaments of Canada, England and many more countries are also excellent real-life examples of the mediator pattern. According to protocols established hundreds of years ago, Members of Parliament (Colleagues) may never address each other directly. All comments are instead directed to the Speaker of the House (the Mediator). Further, it is a breach of protocol to refer to a member by name. All members are referred to as "The Honourable Member from [district]" or "My good friend from [district]." This is analogous to addressing Colleague classes by their "interface" and not by their actual "implementation".

Examples

The word 'Examples' is positioned on the left. To its right are two overlapping circles, one solid light purple and one hollow light purple. Further to the right are three more circles: a solid light purple, a hollow light purple, and another solid light purple.

Chatroom

- To illustrate a possible use of the Mediator pattern, we have chosen to present an example chatroom application in Java. In this particular example, the Forum acts as the abstract Mediator, and the PoliticalForum is the concrete implementation of it. We then have a couple of Colleague implementations: MemberOfParliament, and PrimeMinister (PrimeMinister is actually a subclass of MemberOfParliament, so it too is a Colleague). By having each Colleague aware of the Forum they're participating in, they can send messages without actually referencing one another.