



**Example Use Cases:**

**Mediator Pattern:**

* **Chat Application:** Users communicate through a central chat room mediator.
* **Flight Control System:** Mediator coordinates communication between different control towers and planes.
* **GUI Components:** Mediator manages interactions between UI components (e.g., buttons, text fields).

**Observer Pattern:**

* **UI Components:** UI elements update based on changes in underlying data (e.g., MVC pattern).
* **Stock Market Updates:** Subscribers receive notifications when stock prices change.
* **Event Handling:** Handling user input or system events in graphical applications.

**Proxy Pattern:**

* **Virtual Proxy:** Loading large images or documents only when requested by the client.
* **Protection Proxy:** Controlling access rights to sensitive resources.
* **Remote Proxy:** Accessing objects over a network (e.g., RESTful services, remote method invocation).

**Summary:**

* **Mediator:** Manages communication between objects, promoting loose coupling.
* **Observer:** Establishes dependency relationships, notifying observers of changes.
* **Proxy:** Acts as a surrogate or placeholder, controlling access to the underlying object.

Each pattern serves different purposes: Mediator for centralizing communication, Observer for dependency management with notification, and Proxy for controlled access and resource management. Choosing the right pattern depends on the specific requirements of the system architecture and design goals.