|  |
| --- |
| Chain of Responsibility |
| * Gives more than one object an opportunity to handle a request by linking receiving objects together. * Example: JDK implementation of exception handling. |
| Mediator Pattern |
| * **Too many relationships exist and common point of control or communication is needed.** * Example, group chat program. All users need to have a reference of mediator. Mediator maintains a list of users. When one of the users sends a message, the user actually uses the reference of the mediator to loop through all the users and send out the message to everybody. ([Detail](http://www.journaldev.com/1730/mediator-design-pattern-in-java-example-tutorial))           ChatMediator mediator = new ChatMediatorImpl();          User user1 = new UserImpl(mediator, "Pankaj");          User user2 = new UserImpl(mediator, "Lisa");          mediator.addUser(user1);          mediator.addUser(user2);            user1.send("Hi All"); |
| Observer Pattern |
| * **Other objects will be notified when one object’s state is changed.** * Example: can be found in almost every GUI env. When buttons or other components are placed in application, the application typically registers as a listener for those controls. When a user triggers an event, such as clicking a button, the control iterates through its registered observers and sends a notification to each. (one button can have multiple listeners. ) |
|  |
|  |