LEARN JAVA DESIGN PATTERNS
problem solving approches

Design Patterns Tutorial

- Design Patterns - Home
- Design Patterns - Overview
- Design Patterns - Factory Pattern
- Abstract Factory Pattern
- Design Patterns - Singleton Pattern
- Design Patterns - Builder Pattern
- Design Patterns - Prototype Pattern
- Design Patterns - Adapter Pattern
- Design Patterns - Bridge Pattern
- Design Patterns - Filter Pattern
- Design Patterns - Composite Pattern
- Design Patterns - Decorator Pattern
- Design Patterns - Facade Pattern
- Design Patterns - Flyweight Pattern
- Design Patterns - Proxy Pattern
- Chain of Responsibility Pattern
- Design Patterns - Command Pattern
- Design Patterns - Interpreter Pattern
- Design Patterns - Iterator Pattern
- Design Patterns - Mediator Pattern
- Design Patterns - Memento Pattern
- Design Patterns - Observer Pattern
- Design Patterns - State Pattern
- Design Patterns - Null Object Pattern
- Design Patterns - Strategy Pattern
- Design Patterns - Template Pattern
- Design Patterns - Visitor Pattern
- Design Patterns - MVC Pattern
- Business Delegate Pattern
- Composite Entity Pattern
- Data Access Object Pattern
- Front Controller Pattern
- Intercepting Filter Pattern
- Service Locator Pattern
- Transfer Object Pattern

Design Patterns Resources

- Design Patterns - Questions/Answers

Design Patterns - Mediator Pattern

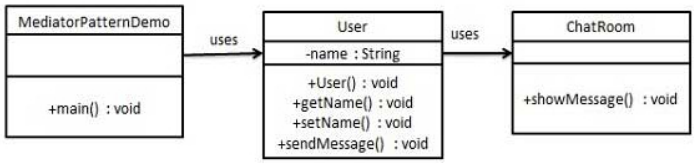
Previous Page Next Page

Mediator pattern is used to reduce communication complexity between multiple objects or classes. This pattern provides a mediator class which normally handles all the communications between different classes and supports easy maintenance of the code by loose coupling. Mediator pattern falls under behavioral pattern category.

Implementation

We are demonstrating mediator pattern by example of a chat room where multiple users can send message to chat room and it is the responsibility of chat room to show the messages to all users. We have created two classes *ChatRoom* and *User*. *User* objects will use *ChatRoom* method to share their messages.

MediatorPatternDemo, our demo class, will use *User* objects to show communication between them.



Step 1

Create mediator class.

ChatRoom.java

```
import java.util.Date;

public class ChatRoom {
   public static void showMessage(User user, String message){
      System.out.println(new Date().toString() + " [" + user.getName() + "] : " + message);
   }
}
```

Step 2

Create user class

User.java

```
public class User {
   private String name;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public User(String name){
      this.name  = name;
   }

   public void sendMessage(String message){
      ChatRoom.showMessage(this,message);
   }
}
```

Step 3

Use the *User* object to show communications between them.

MediatorPatternDemo.java

```
public class MediatorPatternDemo {
```

```
   public static void main(String[] args) {
      User robert = new User("Robert");
      User john = new User("John");

      robert.sendMessage("Hi! John!");
      john.sendMessage("Hello! Robert!");
   }
}
```

Step 4

Verify the output.

```
Thu Jan 31 16:05:46 IST 2013 [Robert] : Hi! John!
Thu Jan 31 16:05:46 IST 2013 [John] : Hello! Robert!
```


Previous Page Print Page Next Page