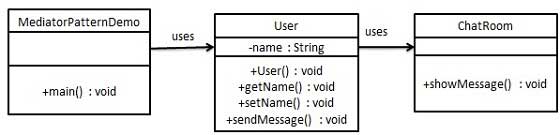
Mediator pattern is used to reduce communication complexity between multiple objects or classes. This pattern provides a mediator class which normally handles all the communications between different classes and supports easy maintenance of the code by loose coupling. Mediator pattern falls under behavioral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



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.time.LocalDateTime;

import java.time.LocalTime;

/\*

\* create Mediator class

\* ChatRoom.java

\*/

class ChatRoom{

public static void showMessag(User user, String message) {

LocalDateTime localDateTime = LocalDateTime.now();

System.out.println(localDateTime.toString() + " : message from "+user.getName());

}

}

/\*

\* create user class

\*/

class User{

private String name;

public User(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public void sendMessage(String message) {

ChatRoom.showMessag(this, message);

}

}

/\*

\* use the user object to show communications between them

\*/

public class TestMediator {

public static void main(String[] args) {

User robert = new User("Robert");

User john = new User("John");

robert.sendMessage("Hi , John");

john.sendMessage("Hi, Robert");

}

}