

Họ và tên: Nguyễn Khánh Quy

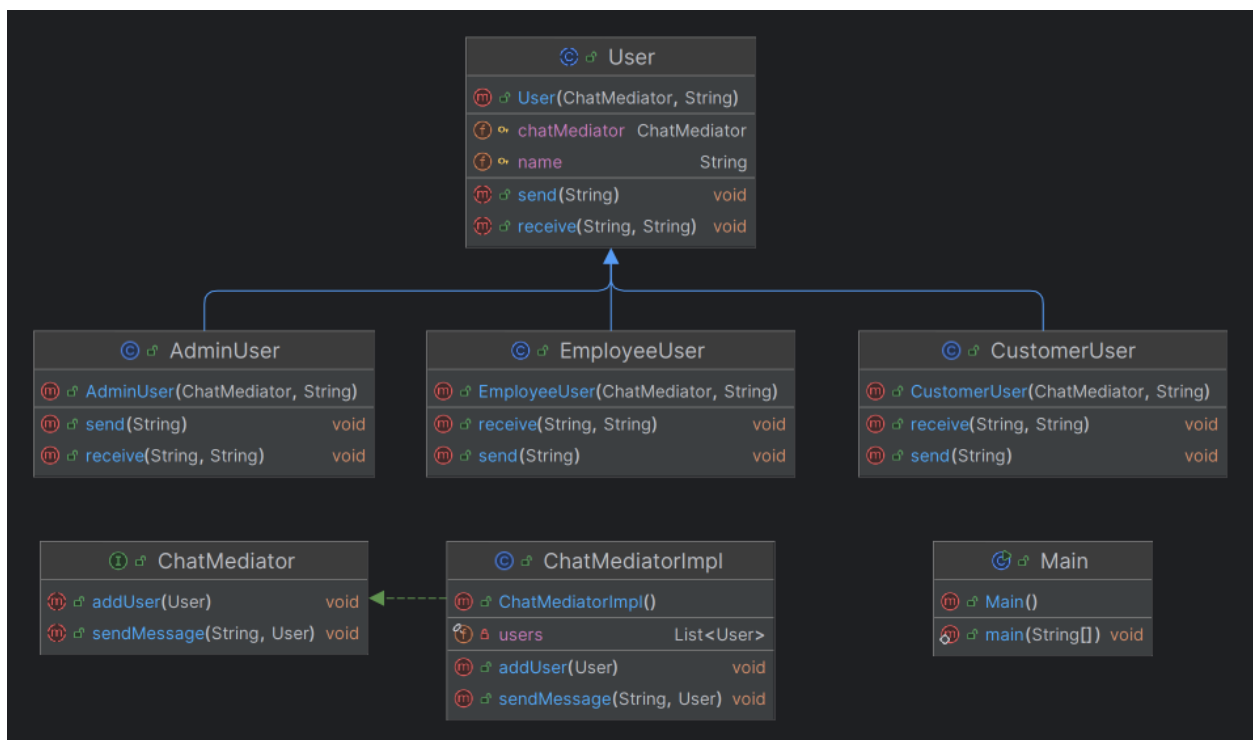
MSSV: 21110282

Mediator Pattern

Mẫu **Mediator** trong lập trình hướng đối tượng là một mẫu thiết kế hành vi được sử dụng để giảm sự phức tạp trong giao tiếp giữa các đối tượng hoặc các lớp trong một hệ thống. **Mediator** làm giảm sự phụ thuộc trực tiếp giữa các đối tượng, giúp chúng dễ dàng được tái sử dụng và bảo trì. Dưới đây là một thí dụ về mẫu

Mediator bằng ngôn ngữ lập trình Java. Trong thí dụ này, chúng ta sẽ thiết kế một hệ thống chat đơn giản mô tả quá trình chat giữa các User trong một hệ thống.

Mediator sẽ đóng vai trò là trung gian điều khiển chat này.



Bước 1: Xác định Interface (ChatMediator):

Đầu tiên, chúng ta cần tạo một interface **ChatMediator**. Đây là interface định nghĩa các phương thức cần thiết cho trung gian trò chuyện. Có hai phương thức

chính là **addUser(User user)** để thêm người dùng vào hệ thống và **sendMessage(String message, User user)** để gửi tin nhắn.

```
public interface ChatMediator {  
  
    void addUser(User user);  
  
    void sendMessage(String message, User user);  
  
}
```

Bước 2: Tạo lớp trừu tượng (User):

Lớp này là lớp trừu tượng đại diện cho các người dùng trong hệ thống. Có các thuộc tính chính là **chatMediator** để lưu trữ trung gian trò chuyện và **name** để lưu trữ tên của người dùng.

Phương thức trừu tượng **send(String message)** và **receive(String message, String nameSender)** được định nghĩa để gửi và nhận tin nhắn.

```
public abstract class User {  
  
    protected ChatMediator chatMediator;  
  
    protected String name;  
  
    public User(ChatMediator chatMediator, String name) {  
        this.chatMediator = chatMediator;  
        this.name = name;  
    }  
  
    public abstract void send(String message);  
  
    public abstract void receive(String message, String nameSender);  
  
}
```

Bước 3: Tạo các lớp con của (User):

Các lớp con như **AdminUser**, **EmployeeUser** và **CustomerUser** mô hình hóa các loại người dùng khác nhau trong hệ thống.

Mỗi lớp con triển khai phương thức **send()** và **receive()** của lớp **User**, đồng thời gọi phương thức tương ứng của **ChatMediator** để gửi tin nhắn.

```
public class AdminUser extends User {

    public AdminUser(ChatMediator chatMediator, String name) {
        super(chatMediator, name);
    }

    @Override
    public void send(String message) {
        System.out.println "[" + name + "]" + " đã gửi tin nhắn: " +
message);
        chatMediator.sendMessage(message, this);
    }

    @Override
    public void receive(String message, String nameSender) {
        System.out.println "[" + name + "]" + " nhận được tin nhắn từ " + "["
+ nameSender + "]" + ": " + message);
    }

}
```

```
public class EmployeeUser extends User {

    public EmployeeUser(ChatMediator chatMediator, String name) {
        super(chatMediator, name);
    }

    @Override
    public void send(String message) {
        System.out.println "[" + name + "]" + " đã gửi tin nhắn: " +
message);
        chatMediator.sendMessage(message, this);
    }

    @Override
    public void receive(String message, String nameSender) {
        System.out.println "[" + name + "]" + " nhận được tin nhắn từ " + "["
+ nameSender + "]" + ": " + message);
    }

}
```

```
public class CustomerUser extends User {

    public CustomerUser(ChatMediator chatMediator, String name) {
        super(chatMediator, name);
    }

}
```

```

        @Override
        public void send(String message) {
            System.out.println "[" + name + "]" + " đã gửi tin nhắn: " +
message);
            chatMediator.sendMessage(message, this);
        }

        @Override
        public void receive(String message, String nameSender) {
            System.out.println "[" + name + "]" + " nhận được tin nhắn từ " + "["
+ nameSender + "]" + ": " + message);
        }
    }
}

```

Bước 4: Tạo lớp ChatMediatorImpl:

Lớp này triển khai interface **ChatMediator**. Nó lưu trữ danh sách các người dùng trong hệ thống và cung cấp các phương thức để thêm người dùng và gửi tin nhắn.

```

import java.util.ArrayList;
import java.util.List;

public class ChatMediatorImpl implements ChatMediator {

    private final List<User> users;

    public ChatMediatorImpl() {
        this.users = new ArrayList<>();
    }

    @Override
    public void addUser(User user) {
        this.users.add(user);
    }

    @Override
    public void sendMessage(String message, User user) {
        for (User u : this.users) {
            // Người gửi sẽ không nhận lại tin nhắn của mình gửi đi
            if (u != user) {
                u.receive(message, user.name);
            }
        }
    }
}

```

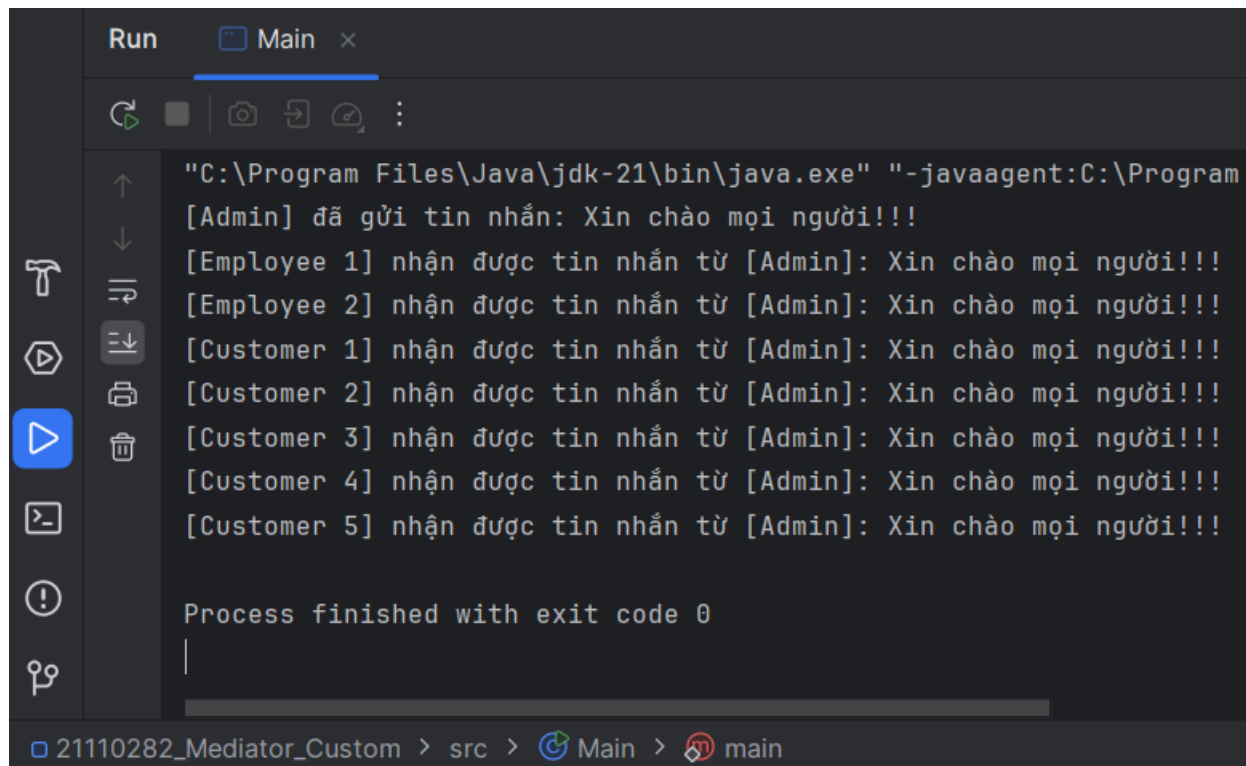
Bước 5: Thử nghiệm

Chúng ta tạo ra một trung gian trò chuyện mới và thêm các người dùng vào đó.

Sau đó, chúng ta thực hiện gửi một tin nhắn từ người dùng quản trị để kiểm tra xem tin nhắn có được chuyển đến các người dùng khác không.

```
public class Main {  
  
    public static void main(String[] args) {  
        ChatMediator mediator = new ChatMediatorImpl();  
  
        User adminUser = new AdminUser(mediator, "Admin");  
  
        User employeeUser1 = new EmployeeUser(mediator, "Employee 1");  
        User employeeUser2 = new EmployeeUser(mediator, "Employee 2");  
  
        User customerUser1 = new CustomerUser(mediator, "Customer 1");  
        User customerUser2 = new CustomerUser(mediator, "Customer 2");  
        User customerUser3 = new CustomerUser(mediator, "Customer 3");  
        User customerUser4 = new CustomerUser(mediator, "Customer 4");  
        User customerUser5 = new CustomerUser(mediator, "Customer 5");  
  
        mediator.addUser(adminUser);  
        mediator.addUser(employeeUser1);  
        mediator.addUser(employeeUser2);  
        mediator.addUser(customerUser1);  
        mediator.addUser(customerUser2);  
        mediator.addUser(customerUser3);  
        mediator.addUser(customerUser4);  
        mediator.addUser(customerUser5);  
  
        adminUser.send("Xin chào mọi người!!!");  
    }  
}
```

Kết quả:



```
Run Main x
[C:\Program Files\Java\jdk-21\bin\java.exe] "-javaagent:C:\Program
[Admin] đã gửi tin nhắn: Xin chào mọi người!!!
[Employee 1] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!
[Employee 2] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!
[Customer 1] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!
[Customer 2] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!
[Customer 3] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!
[Customer 4] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!
[Customer 5] nhận được tin nhắn từ [Admin]: Xin chào mọi người!!!

Process finished with exit code 0
|
21110282_Mediator_Custom > src > Main > main
```

Trong ví dụ này, các đối tượng **AdminUser**, **EmployeeUser**, và **CustomerUser** gửi tin nhắn thông qua **ChatMediator**, giúp giảm sự phụ thuộc lẫn nhau và làm cho mã nguồn dễ quản lý và mở rộng hơn.