Behavioral Design Patterns

Chat App

Memento and Mediator

You are developing a chat application that allows multiple users to communicate with each other. The application has a central server that acts as a mediator between the users. Each user can send messages to one or more users through the server(mediator) which in turn forwards these messages to the appropriate recipients.

Additionally, you want to allow users to undo the last message they sent. You decide to use the Memento design pattern to implement this feature.

You also want to implement a feature that allows users to block messages from specific users. You decide to use the Mediator design pattern to implement this feature.

Write a Java program that implements the **Mediator** and **Memento** design patterns. Your program should include the following **classes**:

Message: A class representing a message sent by a user. It should have properties for the sender, recipient(s), timestamp, and message content.

User: A class representing a user of the chat application. It should have methods to send messages to other users, receive messages from other users, and undo the last message sent.

NOTE: You will NOT communicate with other Users directly you will use Mediator!

ChatServer: The mediator class that manages communication between users. It should have methods to register and unregister users, send messages from one user to one or more other users, and block messages from specific users.

ChatHistory: A class that stores the chat history for a user. It should have methods to add a new message to the history and get the last message sent.

MessageMemento: A class that represents a snapshot of a message sent by a user. It should have properties for the message content and timestamp.

Your program should demonstrate the following features:

Users can send messages to one or more other users through the chat server.

Users can undo the last message they sent using the Memento design pattern.

Users can block messages from specific users using the Mediator design pattern.

Users can receive messages from other users and view the chat history for a specific user.

Your program should include a driver that demonstrates these features with 3 users.

Iterator

Complete the chat application above and then modify it to allow Users to iterate over their message history by a specific user. Meaning they can iterate through their messages by a specific user name.

ChatHistory: Make this class an Iterable and create an Iterator class that will input a User and iterate over ChatHistory for messages sent or received by the user.

> HINT: Instead of using Java.lang Interface you should use an interface that allows you to pass an argument into the iterator. Thus, ChatHistory should implement interface below instead of Iterable.

```
import java.util.Iterator;

public interface IterableByUser {
    Iterator iterator(User userToSearchWith);
}
```

User: Make this class an Iterable and create a wrapper to allow the iterator method from ChatHistory to be called from the User iterator method. User class should implement the interface above.

searchMessagesByUser: a class that will implement Iterator.

> NOTE: There is a potential bug that could arise between the time **hasNext()** and **next()** is called as the collection you are iterating through can be modified. In this case it can have messages removed from the collection. Disregard this bug and assume that the collection is immutable, meaning that it can NOT be modified, after hasNext() is called.