Homework : 06

Github: https://github.com/chaitanyanalage/CS5800/tree/main

Code:

```java
package org.example;

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

public class ChatHistory implements Iterable<Message> {
    private List<Message> sentMessages;
    private List<Message> receivedMessages;

    public ChatHistory() {
        sentMessages = new ArrayList<>();
        receivedMessages = new ArrayList<>();
    }

    public void addReceivedMessage(Message message) {
        receivedMessages.add(message);
    }

    public void addSentMessage(Message message) {
        sentMessages.add(message);
    }

    public Message getLastSentMessages() {
        if (!sentMessages.isEmpty()) {
            return sentMessages.get(sentMessages.size() - 1);
        } else {
            return null;
        }
    }

    public Message getLastReceivedMessage() {
        if (receivedMessages.size() > 0) {
            return receivedMessages.get(receivedMessages.size() - 1);
        } else {
            return null;
        }
    }

    public void removeLastSentMessage(Message message) {
        sentMessages.remove(message);
    }

    public List<Message> getSentMessages() {
        return sentMessages;
    }

    public void removeLastReceivedMessage(Message message) {
        receivedMessages.remove(message);
    }

    public List<Message> combineMessages() {
```

```
        List<Message> combinedMessages = new ArrayList<>(sentMessages);
        combinedMessages.addAll(receivedMessages);
        return combinedMessages;
    }

    @Override
    public Iterator<Message> iterator() {
        return combineMessages().iterator();
    }

    public Iterator<Message> iterator(User userToSearchWith) {
        return new SearchMessagesByUser(combineMessages().iterator(),
userToSearchWith);
    }
}
```

```
package org.example;

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

public class ChatServer {
    private List<User> users;

    public ChatServer(){
        users = new ArrayList<>();
    }


    public void sendMessage(Message message){
        User sender = message.getSender();
        List<User> receivers = new ArrayList<>(message.getReceivers());
        if(!users.contains(sender)){
            System.out.printf("Cannot send message as user %s is not
registered.\n", sender.getUsername());
            return;
        }
        List<User> validReceivers = new ArrayList<>();
        for (User user : receivers){
            if (!users.contains(user)){
                System.out.printf("Cannot send message from %s to %s as user
%s is not registered.\n", sender.getUsername(), user.getUsername(),
user.getUsername());
            }
            else {
                validReceivers.add(user);
            }
        }
        for (User receiver : validReceivers){
            List<User> blockedAccounts = receiver.getBlockedUsers();
            if (blockedAccounts != null && blockedAccounts.contains(sender)){
                System.out.println("Cannot send message from " +
sender.getUsername() + " to " + receiver.getUsername() +
                        " because " + sender.getUsername() + " is blocked by
" + receiver.getUsername() + ".");
            } else{
```

```
                sender.sendMessage(message);
                System.out.printf("Successfully sent message from %s to
%s\n", sender.getUsername(), receiver.getUsername());
                receiver.receiveMessage(message);
                System.out.printf("%s: %s received message from %s: '%s'\n",
message.getTimestamp(),
                        receiver.getUsername(), sender.getUsername(),
message.getTextMessage());
            }
        }
    }

    public void registerUser(User user) {
        users.add(user);
        System.out.printf("Successfully registered user %s!\n",
user.getUsername());
    }

    public void unregisterUser(User user) {
        users.remove(user);
        System.out.printf("Unregistered user %s from system!!!\n",
user.getUsername());
    }

    public void undoLastMessage(User user){
        List<Message> sentMessages = user.getChatHistory().getSentMessages();
        if (sentMessages.size() == 0){
            System.out.printf("Cannot un-send last message as user %s has not
sent any messages.\n", user.getUsername());
            return;
        }
        Message message = user.getChatHistory().getLastSentMessages();
        user.undoLastSentMessage();
        List<User> receivers = message.getReceivers();
        for (User receiver : receivers){
            receiver.getChatHistory().removeLastReceivedMessage(message);
        }
    }

    public List<User> getUsers(){
        return users;
    }
}
```

```
package org.example;

import java.util.Iterator;

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

```
package org.example;

import java.util.Iterator;
import java.util.List;
```

```
public class MainSystem {
    private static final ChatServer chatServer = new ChatServer();

    public static void main(String[] args) {
        //creating 4 users and adding them to system
        User roomfrnd1 = new User("Chaitanya Nalage", chatServer);
        User roomfrnd2 = new User("Rituraj Gharat", chatServer);
        User roomfrnd3 = new User("Aryan Dive", chatServer);
        User roomfrnd4 = new User("Sanika Kubal", chatServer);
        System.out.println("\n------------------------ User's Created -------
-----------------");


        System.out.println("-----------------------------------");
        System.out.println("-----------------------------------");
        chatServer.sendMessage(new Message(roomfrnd1, List.of(roomfrnd2),
"Rituraj, hello! What's for supper this evening?"));
        chatServer.sendMessage(new Message(roomfrnd4, List.of(roomfrnd2),
"Hello, Rituraj How do you feel right now?"));
        chatServer.sendMessage(new Message(roomfrnd2, List.of(roomfrnd4),
"Hi, Sanika Thank you for your inquiry, unfortunately things have become
worse."));
        chatServer.sendMessage(new Message(roomfrnd4, List.of(roomfrnd1),
"Chaitanya, I appreciate you not asking me what to eat."));
        chatServer.sendMessage(new Message(roomfrnd1, List.of(roomfrnd3),
"Hello, Aryan Is chicken on the menu for today, or are you still sticking to
vegetarian food?"));
        chatServer.sendMessage(new Message(roomfrnd1, List.of(roomfrnd4), "I
apologize so much, Sanika. What would you want to eat?"));
        chatServer.sendMessage(new Message(roomfrnd3, List.of(roomfrnd1),
"Greetings, Chaitanya I'm not eating meat, so I can't have chicken."));
        System.out.println("-----------------------------------");
        System.out.println("-----------------------------------");
        System.out.println("\n------------------------ Demonstrating block
function ------------------------");
        roomfrnd2.blockerUsers(roomfrnd1);
        System.out.println("-----------------------------------");
        chatServer.sendMessage(new Message(roomfrnd1, List.of(roomfrnd2,
roomfrnd3), "Aryan, take me to the Cheesecake Factory, please!"));
        System.out.println("-----------------------------------");
        chatServer.sendMessage(new Message(roomfrnd3, List.of(roomfrnd1),
"So, Chaitanya, why cheesecake factory?"));
        System.out.println("-----------------------------------");
        System.out.println("\n------------------------ Demonstrating block
function ------------------------");
        System.out.println("Aryan Dive unsent last message");
        chatServer.undoLastMessage(roomfrnd3);
        System.out.printf("Now, Aryan Dive's last message is '%s'\n",
roomfrnd3.getChatHistory().getLastSentMessages());
        System.out.println("-----------------------------------");
        System.out.println("\n===== Demonstrating unsent function =====");
        System.out.println("Rituraj Gharat unsent last message:");
        chatServer.undoLastMessage(roomfrnd2);
        System.out.println("-----------------------------------");
```

```
        // Trying iterating over all messages in user3's chat history
        System.out.println("Iterating over all messages in Aryan Dive's chat
history:");
        Iterator<Message> allMessagesIterator = roomfrnd3.iterator();
        while (allMessagesIterator.hasNext()) {
            System.out.println(allMessagesIterator.next());
        }
        System.out.printf("-------------------------------------\n");
        // Trying iterating over all messages in user1's chat history
        System.out.println("Iterating over all messages in Chaitanya Nalage's
chat history:");
        allMessagesIterator = roomfrnd1.iterator();
        while (allMessagesIterator.hasNext()) {
            System.out.println(allMessagesIterator.next());
        }
        System.out.printf("-------------------------------------\n");
        // Trying iterating over all messages in user4's chat history
        System.out.println("Iterating over all messages in Sanika Kubal chat
history:");
        allMessagesIterator = roomfrnd4.iterator();
        while (allMessagesIterator.hasNext()) {
            System.out.println(allMessagesIterator.next());
        }
        System.out.printf("-------------------------------------\n");
        chatServer.unregisterUser(roomfrnd1);
        chatServer.sendMessage(new Message(roomfrnd3, List.of(roomfrnd1),
"Since I'd like to have the cheesecakes from there."));
    }
}
```

```
package org.example;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;


public class Message {
    private User sender;
    private List<User> receivers;
    private String textMessage;
    private Date timestamp;

    public Message(User sender, List<User> receivers, String textMessage) {
        this.sender = sender;
        this.receivers = receivers;
        this.textMessage = textMessage;
        this.timestamp = new Date();
    }

    public MessageMememto saveToMememto(){
        return new MessageMememto(this);
    }
```

```
    public void restoreFromMememto(MessageMememto messageMememto){
        Message previousMessage = messageMememto.getPreviousMessage();
        this.sender = previousMessage.getSender();
        this.receivers = previousMessage.getReceivers();
        this.textMessage = previousMessage.getTextMessage();
        this.timestamp = previousMessage.getTimestamp();
    }

    public String getTextMessage(){
        return textMessage;
    }

    public List<User> getReceivers(){
        return receivers;
    }


    public User getSender(){
        return sender;
    }



    public String toString(){
        return String.format("%s: Message content: '%s'",
                timestamp.toString(), textMessage);
    }

    public Date getTimestamp(){
        return timestamp;
    }
}
```

```
package org.example;

public class MessageMememto {
    private Message message;
    // A class that represents a snapshot of a message sent by a user. It
should have
    //properties for the message content and timestamp.

    public Message getPreviousMessage(){
        return message;
    }

    public MessageMememto(Message message){
        this.message = message;
    }


}
```

```
package org.example;
```

```
import java.util.Iterator;

public class SearchMessagesByUser implements Iterator<Message> {
    private Iterator<Message> messageIterator;
    private User userToSearchWith;

    public SearchMessagesByUser(Iterator<Message> messageIterator, User
userToSearchWith) {
        this.messageIterator = messageIterator;
        this.userToSearchWith = userToSearchWith;
    }

    @Override
    public boolean hasNext() {
        while (messageIterator.hasNext()) {
            Message message = messageIterator.next();
            if (message.getSender().equals(userToSearchWith) ||
                    message.getReceivers().contains(userToSearchWith)) {
                messageIterator = userToSearchWith.iterator();
                return true;
            }
        }
        return false;
    }

    @Override
    public Message next() {
        while (messageIterator.hasNext()) {
            Message message = messageIterator.next();
            if (message.getSender().equals(userToSearchWith) ||
                    message.getReceivers().contains(userToSearchWith)) {
                return message;
            }
        }
        return null;
    }

    @Override
    public void remove() {
        throw new UnsupportedOperationException();
    }
}
```

```
package org.example;

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

public class User implements Iterable<Message>, IterableByUser{
    private String username;
    private ChatServer chatServer;
    private ChatHistory chatHistory;
    private List<MessageMememto> messageMememtos;
    private List<User> blockedUsers;
```

```
    public User(String username, ChatServer chatServer){
        this.username = username;
        this.chatServer = chatServer;
        chatServer.registerUser(this);
        this.chatHistory = new ChatHistory();
        this.messageMememtos = new ArrayList<>();
        this.blockedUsers = new ArrayList<>();
    }

    public void sendMessage(Message message){
        this.chatHistory.addSentMessage(message);
    }


    public void receiveMessage(Message message){
        this.chatHistory.addReceivedMessage(message);
    }

    public String getUsername(){
        return username;
    }

    public void undoLastSentMessage(){
        List<Message> sentMessages = chatHistory.getSentMessages();
        Message lastMessage = sentMessages.get(sentMessages.size() - 1);
        chatHistory.removeLastSentMessage(lastMessage);
        MessageMememto messageMememto = lastMessage.saveToMememto();
        lastMessage.restoreFromMememto(messageMememto);
        sentMessages.remove(lastMessage);
    }

    public void blockerUsers(User blockedUser){
        List<User> users = chatServer.getUsers();
        if (!users.contains(this)){
            System.out.printf("User %s is not registered\n", username);
            return;
        } else if (!users.contains(blockedUser)){
            System.out.printf("User %s is not registered\n",
blockedUser.getUsername());
            return;
        }
        setBlockUsers(blockedUser);
    }

    public void setBlockUsers(User blockedUser){
        if (blockedUsers != null && blockedUsers.contains(blockedUser)){
            System.out.println("User " + blockedUser.getUsername() + " has
already blocked user " + blockedUser.getUsername());
        } else {
            blockedUsers.add(blockedUser);
            System.out.println("User " + username + " has blocked user " +
blockedUser.getUsername());
        }
    }

    public List<User> getBlockedUsers(){
```

```
        return blockedUsers;
    }

    public ChatHistory getChatHistory() {
        return chatHistory;
    }

    @Override
    public Iterator<Message> iterator() {
        return chatHistory.iterator();
    }

    @Override
    public Iterator<Message> iterator(User userToSearchWith) {
        return chatHistory.iterator(userToSearchWith);
    }
}
```

```
import java.util.List;
import org.example.ChatServer;
import org.example.User;
import org.example.Message;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class MainSysTest {
    private ChatServer chatServer;
    private User user1, user2, user3, user4;

    @BeforeEach
    void setUp() {
        chatServer = new ChatServer();
        user1 = new User("Chaitanya Nalage", chatServer);
        user2 = new User("Rituraj Gharat", chatServer);
        user3 = new User("Aryan Dive", chatServer);
        user4 = new User("Sanika Kubal", chatServer);
    }

//    @Test
//    void testSendMessage() {
//        chatServer.sendMessage(new Message(user1, List.of(user2), "Hey!"));
//        assertEquals("Hey!",
user2.getChatHistory().getLastSentMessages().getTextMessage());
//    }
//
//    @Test
//    void testReceiveMessage() {
//        chatServer.sendMessage(new Message(user1, List.of(user2), "Hey!"));
//        Message lastReceivedMessage =
user2.getChatHistory().getLastSentMessages();
//        assertNotNull(lastReceivedMessage);
//        assertEquals("Hey!", lastReceivedMessage.getTextMessage());
//    }
```

```
    @Test
    void testUserBlocking() {
        user1.blockerUsers(user2);
        assertTrue(user1.getBlockedUsers().contains(user2));
    }

    @Test
    void testMessageUndo() {
        chatServer.sendMessage(new Message(user3, List.of(user1), "Message to
be undone"));
        int originalNumOfMessages =
user3.getChatHistory().getSentMessages().size();
        chatServer.undoLastMessage(user3);
        assertEquals(originalNumOfMessages - 1,
user3.getChatHistory().getSentMessages().size());
    }

    @Test
    void testUserRegistration() {
        assertTrue(chatServer.getUsers().contains(user1));
        assertTrue(chatServer.getUsers().contains(user2));
        assertTrue(chatServer.getUsers().contains(user3));
        assertTrue(chatServer.getUsers().contains(user4));
    }

    @Test
    void testUnregisterUser() {
        chatServer.unregisterUser(user4);
        assertFalse(chatServer.getUsers().contains(user4));
    }


}
```

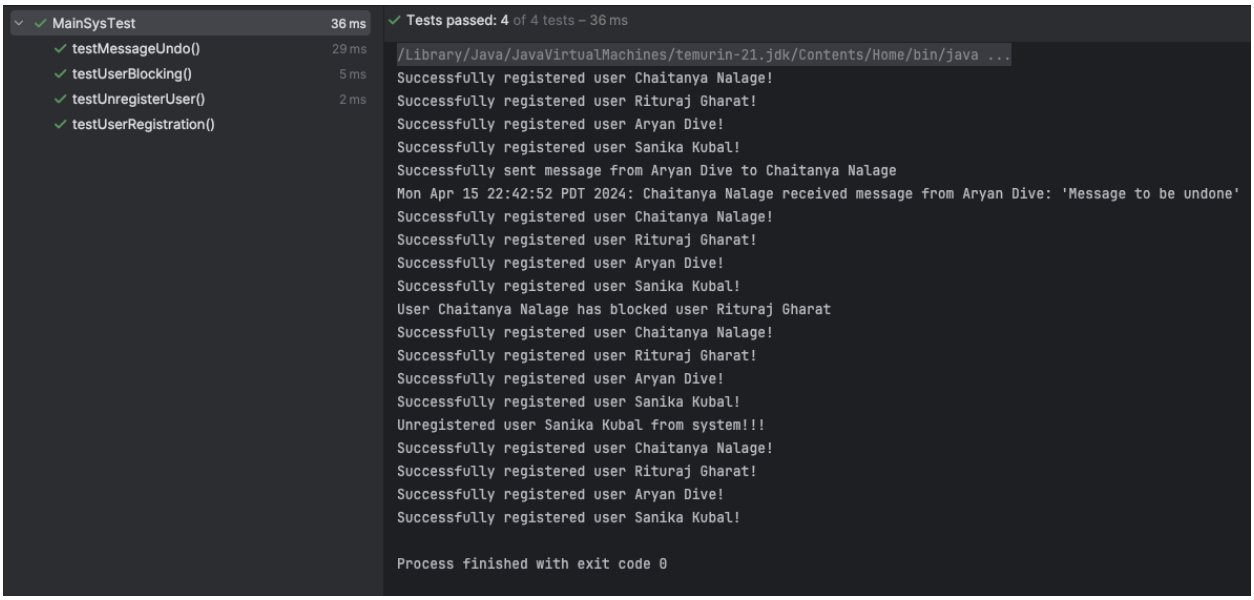
Output:

Driver:

```
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59346:/Applications/IntelliJ IDEA.a
Successfully registered user Chaitanya Nalage!
Successfully registered user Rituraj Gharat!
Successfully registered user Aryan Dive!
Successfully registered user Sanika Kubal!

---------- User's Created ----------
---------------------------------
---------------------------------
Successfully sent message from Chaitanya Nalage to Rituraj Gharat
Mon Apr 15 22:40:48 PDT 2024: Rituraj Gharat received message from Chaitanya Nalage: 'Rituraj, hello! What's for supper this evening?'
Successfully sent message from Sanika Kubal to Rituraj Gharat
Mon Apr 15 22:40:48 PDT 2024: Rituraj Gharat received message from Sanika Kubal: 'Hello, Rituraj How do you feel right now?'
Successfully sent message from Rituraj Gharat to Sanika Kubal
Mon Apr 15 22:40:48 PDT 2024: Sanika Kubal received message from Rituraj Gharat: 'Hi, Sanika Thank you for your inquiry, unfortunately things have become worse.'
Successfully sent message from Sanika Kubal to Chaitanya Nalage
Mon Apr 15 22:40:48 PDT 2024: Chaitanya Nalage received message from Sanika Kubal: 'Chaitanya, I appreciate you not asking me what to eat.'
Successfully sent message from Chaitanya Nalage to Aryan Dive
Mon Apr 15 22:40:48 PDT 2024: Aryan Dive received message from Chaitanya Nalage: 'Hello, Aryan Is chicken on the menu for today, or are you still sticking to vegetarian food?'
Successfully sent message from Chaitanya Nalage to Sanika Kubal
Mon Apr 15 22:40:48 PDT 2024: Sanika Kubal received message from Chaitanya Nalage: 'I apologize so much, Sanika. What would you want to eat?'
Successfully sent message from Aryan Dive to Chaitanya Nalage
Mon Apr 15 22:40:48 PDT 2024: Chaitanya Nalage received message from Aryan Dive: 'Greetings, Chaitanya I'm not eating meat, so I can't have chicken.'
---------------------------------
---------------------------------

---------- Demonstrating block function ----------
User Rituraj Gharat has blocked user Chaitanya Nalage
---------------------------------
Cannot send message from Chaitanya Nalage to Rituraj Gharat because Chaitanya Nalage is blocked by Rituraj Gharat.
Successfully sent message from Chaitanya Nalage to Aryan Dive
Mon Apr 15 22:40:48 PDT 2024: Aryan Dive received message from Chaitanya Nalage: 'Aryan, take me to the Cheesecake Factory, please!'
---------------------------------
Successfully sent message from Aryan Dive to Chaitanya Nalage
Mon Apr 15 22:40:48 PDT 2024: Chaitanya Nalage received message from Aryan Dive: 'So, Chaitanya, why cheesecake factory?'
---------------------------------

---------- Demonstrating block function ----------
Aryan Dive unsent last message
Now, Aryan Dive's last message is 'Mon Apr 15 22:40:48 PDT 2024: Message content: 'Greetings, Chaitanya I'm not eating meat, so I can't have chicken.''
```

```
---------- Demonstrating block function ----------
Aryan Dive unsent last message
Now, Aryan Dive's last message is 'Mon Apr 15 22:40:48 PDT 2024: Message content: 'Greetings, Chaitanya I'm not eating meat, so I can't have chicken.''
---------------------------------

===== Demonstrating unsent function =====
Rituraj Gharat unsent last message:
---------------------------------
Iterating over all messages in Aryan Dive's chat history:
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Greetings, Chaitanya I'm not eating meat, so I can't have chicken.'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Hello, Aryan Is chicken on the menu for today, or are you still sticking to vegetarian food?'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Aryan, take me to the Cheesecake Factory, please!'
---------------------------------
Iterating over all messages in Chaitanya Nalage's chat history:
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Rituraj, hello! What's for supper this evening?'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Hello, Aryan Is chicken on the menu for today, or are you still sticking to vegetarian food?'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'I apologize so much, Sanika. What would you want to eat?'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Aryan, take me to the Cheesecake Factory, please!'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Chaitanya, I appreciate you not asking me what to eat.'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Greetings, Chaitanya I'm not eating meat, so I can't have chicken.'
---------------------------------
Iterating over all messages in Sanika Kubal chat history:
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Hello, Rituraj How do you feel right now?'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'Chaitanya, I appreciate you not asking me what to eat.'
Mon Apr 15 22:40:48 PDT 2024: Message content: 'I apologize so much, Sanika. What would you want to eat?'
---------------------------------
Unregistered user Chaitanya Nalage from system!!!
Cannot send message from Aryan Dive to Chaitanya Nalage as user Chaitanya Nalage is not registered.

Process finished with exit code 0
```

Test Case:


MainSysTest 36 ms
testMessageUndo() 29 ms
testUserBlocking() 5 ms
testUnregisterUser() 2 ms
testUserRegistration()
Tests passed: 4 of 4 tests – 36 ms
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Contents/Home/bin/java ...
Successfully registered user Chaitanya Nalage!
Successfully registered user Rituraj Gharat!
Successfully registered user Aryan Dive!
Successfully registered user Sanika Kubal!
Successfully sent message from Aryan Dive to Chaitanya Nalage
Mon Apr 15 22:42:52 PDT 2024: Chaitanya Nalage received message from Aryan Dive: 'Message to be undone'
Successfully registered user Chaitanya Nalage!
Successfully registered user Rituraj Gharat!
Successfully registered user Aryan Dive!
Successfully registered user Sanika Kubal!
User Chaitanya Nalage has blocked user Rituraj Gharat
Successfully registered user Chaitanya Nalage!
Successfully registered user Rituraj Gharat!
Successfully registered user Aryan Dive!
Successfully registered user Sanika Kubal!
Unregistered user Sanika Kubal from system!!!
Successfully registered user Chaitanya Nalage!
Successfully registered user Rituraj Gharat!
Successfully registered user Aryan Dive!
Successfully registered user Sanika Kubal!
Process finished with exit code 0