CS5800 Homework 06
Subham Panda
017314921

[GitHub link]

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;
```

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

```java
package org.example;

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);
    }
}
```

```
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 projectLead = new User("Subham Panda", chatServer);
        User backendDev = new User("Amrit Nandan", chatServer);
        User frontendDev = new User("Rajat Pattnaik", chatServer);
        User qaEngineer = new User("Animish Choudhury", chatServer);

        System.out.println("\n===== Team Members Created =====");

        System.out.println("---------------------------------");
        chatServer.sendMessage(new Message(projectLead, List.of(backendDev),
"Amrit, could you update the status of the backend API integration?"));
        chatServer.sendMessage(new Message(qaEngineer, List.of(backendDev),
"Amrit, have you fixed the bugs I reported yesterday?"));
        chatServer.sendMessage(new Message(backendDev, List.of(qaEngineer),
"Hey Animish, I've addressed the critical bugs. Please verify on your
end."));
        chatServer.sendMessage(new Message(qaEngineer, List.of(projectLead),
"Subham, the test results look promising. We should discuss the next
steps."));
        chatServer.sendMessage(new Message(projectLead, List.of(frontendDev),
"Rajat, do we need additional resources for the UI/UX phase?"));
        chatServer.sendMessage(new Message(projectLead, List.of(qaEngineer),
"Animish, can you prioritize the load testing for tomorrow?"));
        chatServer.sendMessage(new Message(frontendDev, List.of(projectLead),
"Hello, Subham, I think we're on track. However, I'll need the final assets
by Friday."));
        System.out.println("---------------------------------");

        System.out.println("\n===== Demonstrating block function =====");
        backendDev.blockerUsers(projectLead);
        System.out.println("---------------------------------");
        chatServer.sendMessage(new Message(projectLead, List.of(backendDev,
frontendDev), "Please check your emails for the updated project timeline."));
        System.out.println("---------------------------------");
        chatServer.sendMessage(new Message(frontendDev, List.of(projectLead),
"Subham, could we discuss the timeline adjustment in tomorrow's meeting?"));
        System.out.println("---------------------------------");

        System.out.println("\n===== Demonstrating unsent function =====");
        System.out.println("Rajat Pattnaik unsent last message");
        chatServer.undoLastMessage(frontendDev);
        System.out.printf("Now, Rajat Pattnaik's last message is '%s'\n",
frontendDev.getChatHistory().getLastSentMessages());
        System.out.println("---------------------------------");

        System.out.println("\n===== Demonstrating unsent function =====");
        System.out.println("Amrit Nandan unsent last message:");
        chatServer.undoLastMessage(backendDev);
        System.out.println("---------------------------------");
        // Iterating over all messages in user3's chat history
        System.out.println("Iterating over all messages in Rajat Pattnaik's
chat history:");
        Iterator<Message> allMessagesIterator = frontendDev.iterator();
        while (allMessagesIterator.hasNext()) {
            System.out.println(allMessagesIterator.next());
        }
        System.out.println("---------------------------------\n");
        // Iterating over all messages in user1's chat history
```

```
        System.out.println("Iterating over all messages in Subham Panda's
chat history:");
        allMessagesIterator = projectLead.iterator();
        while (allMessagesIterator.hasNext()) {
            System.out.println(allMessagesIterator.next());
        }
        System.out.println("-------------------------------------\n");
        // Iterating over all messages in user4's chat history
        System.out.println("Iterating over all messages in Animish
Choudhury's chat history:");
        allMessagesIterator = qaEngineer.iterator();
        while (allMessagesIterator.hasNext()) {
            System.out.println(allMessagesIterator.next());
        }
        System.out.println("-------------------------------------\n");

        System.out.println("\n===== Demonstrating unregister function
=====");
        chatServer.unregisterUser(qaEngineer);
        chatServer.sendMessage(new Message(projectLead, List.of(qaEngineer),
"Lets catch up!"));
        System.out.println("-------------------------------------\n");
    }
}
```

```
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("Subham Panda", chatServer);
        user2 = new User("Amrit Nandan", chatServer);
        user3 = new User("Rajat Pattanaik", chatServer);
        user4 = new User("Animish Choudhury", chatServer);
    }

    @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"));
```

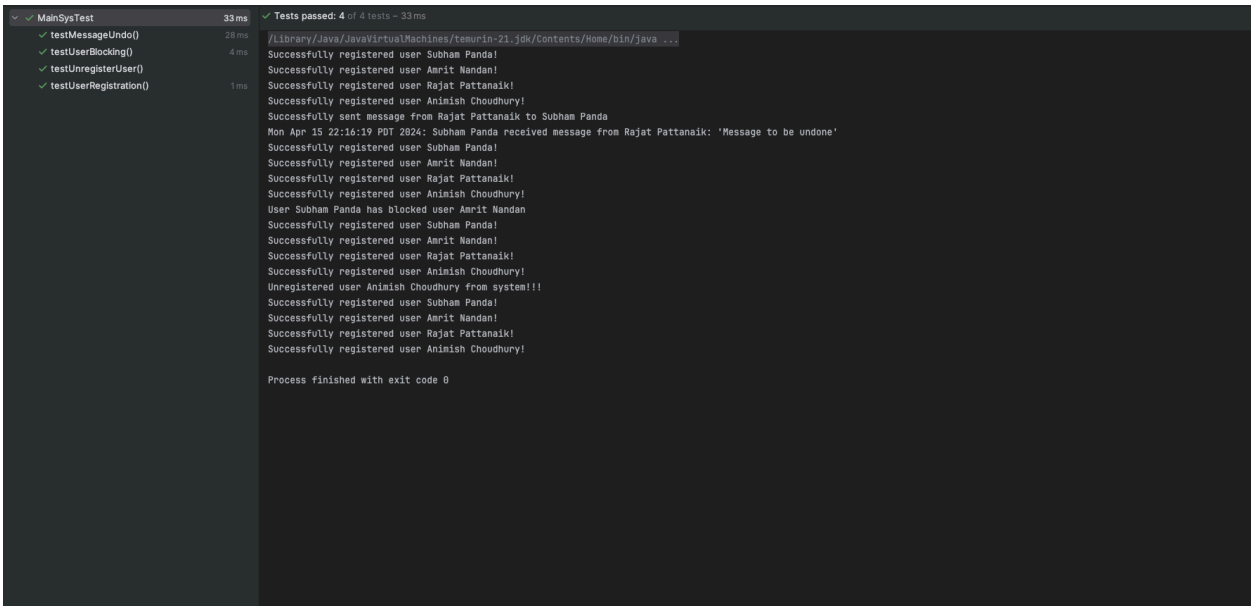
```
        int originalNumOfMessages =
user3.getChatHistory().getSentMessages().size();
        chatServer.undoLastMessage(user3);
        assertEquals(originalNumOfMessages - 1,
user3.getChatHistory().getSentMessages().size());
    }

    @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=49389:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.e
Successfully registered user Subham Panda!
Successfully registered user Amrit Nandan!
Successfully registered user Rajat Pattnaik!
Successfully registered user Animish Choudhury!

===== Team Members Created =====
---------------------------------
Successfully sent message from Subham Panda to Amrit Nandan
Mon Apr 15 22:13:48 PDT 2024: Amrit Nandan received message from Subham Panda: 'Amrit, could you update the status of the backend API integration?'
Successfully sent message from Animish Choudhury to Amrit Nandan
Mon Apr 15 22:13:48 PDT 2024: Amrit Nandan received message from Animish Choudhury: 'Amrit, have you fixed the bugs I reported yesterday?'
Successfully sent message from Amrit Nandan to Animish Choudhury
Mon Apr 15 22:13:48 PDT 2024: Animish Choudhury received message from Amrit Nandan: 'Hey Animish, I've addressed the critical bugs. Please verify on your end.'
Successfully sent message from Animish Choudhury to Subham Panda
Mon Apr 15 22:13:48 PDT 2024: Subham Panda received message from Animish Choudhury: 'Subham, the test results look promising. We should discuss the next steps.'
Successfully sent message from Subham Panda to Rajat Pattnaik
Mon Apr 15 22:13:48 PDT 2024: Rajat Pattnaik received message from Subham Panda: 'Rajat, do we need additional resources for the UI/UX phase?'
Successfully sent message from Subham Panda to Animish Choudhury
Mon Apr 15 22:13:48 PDT 2024: Animish Choudhury received message from Subham Panda: 'Animish, can you prioritize the load testing for tomorrow?'
Successfully sent message from Rajat Pattnaik to Subham Panda
Mon Apr 15 22:13:48 PDT 2024: Subham Panda received message from Rajat Pattnaik: 'Hello, Subham, I think we're on track. However, I'll need the final assets by Friday.'
---------------------------------

===== Demonstrating block function =====
User Amrit Nandan has blocked user Subham Panda
---------------------------------
Cannot send message from Subham Panda to Amrit Nandan because Subham Panda is blocked by Amrit Nandan.
Successfully sent message from Subham Panda to Rajat Pattnaik
Mon Apr 15 22:13:48 PDT 2024: Rajat Pattnaik received message from Subham Panda: 'Please check your emails for the updated project timeline.'
---------------------------------
Successfully sent message from Rajat Pattnaik to Subham Panda
Mon Apr 15 22:13:48 PDT 2024: Subham Panda received message from Rajat Pattnaik: 'Subham, could we discuss the timeline adjustment in tomorrow's meeting?'
---------------------------------


===== Demonstrating unsent function =====
Rajat Pattnaik unsent last message
Now, Rajat Pattnaik's last message is 'Mon Apr 15 22:13:48 PDT 2024: Message content: 'Hello, Subham, I think we're on track. However, I'll need the final assets by Friday.''
---------------------------------

===== Demonstrating unsent function =====
Amrit Nandan unsent last message:
---------------------------------
Iterating over all messages in Rajat Pattnaik's chat history:
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Hello, Subham, I think we're on track. However, I'll need the final assets by Friday.'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Rajat, do we need additional resources for the UI/UX phase?'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Please check your emails for the updated project timeline.'
---------------------------------

Iterating over all messages in Subham Panda's chat history:
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Amrit, could you update the status of the backend API integration?'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Rajat, do we need additional resources for the UI/UX phase?'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Animish, can you prioritize the load testing for tomorrow?'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Please check your emails for the updated project timeline.'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Subham, the test results look promising. We should discuss the next steps.'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Hello, Subham, I think we're on track. However, I'll need the final assets by Friday.'
---------------------------------

Iterating over all messages in Animish Choudhury's chat history:
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Amrit, have you fixed the bugs I reported yesterday?'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Subham, the test results look promising. We should discuss the next steps.'
Mon Apr 15 22:13:48 PDT 2024: Message content: 'Animish, can you prioritize the load testing for tomorrow?'
---------------------------------


===== Demonstrating unregister function =====
Unregistered user Animish Choudhury from system!!!
Cannot send message from Subham Panda to Animish Choudhury as user Animish Choudhury is not registered.
-------------------------------------


Process finished with exit code 0
```

Test Cases


MainSysTest 33 ms
testMessageUndo() 28 ms
testUserBlocking() 4 ms
testUnregisterUser()
testUserRegistration() 1 ms
Tests passed: 4 of 4 tests – 33 ms
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Contents/Home/bin/java ...
Successfully registered user Subham Panda!
Successfully registered user Amrit Nandan!
Successfully registered user Rajat Pattanaik!
Successfully registered user Animish Choudhury!
Successfully sent message from Rajat Pattanaik to Subham Panda
Mon Apr 15 22:16:19 PDT 2024: Subham Panda received message from Rajat Pattanaik: 'Message to be undone'
Successfully registered user Subham Panda!
Successfully registered user Amrit Nandan!
Successfully registered user Rajat Pattanaik!
Successfully registered user Animish Choudhury!
User Subham Panda has blocked user Amrit Nandan
Successfully registered user Subham Panda!
Successfully registered user Amrit Nandan!
Successfully registered user Rajat Pattanaik!
Successfully registered user Animish Choudhury!
Unregistered user Animish Choudhury from system!!!
Successfully registered user Subham Panda!
Successfully registered user Amrit Nandan!
Successfully registered user Rajat Pattanaik!
Successfully registered user Animish Choudhury!
Process finished with exit code 0