```java
package edu.cpp.cs5800.chatapp;

import java.util.Iterator;
import java.util.List;
import java.util.Objects;

public class User {
    private static int idIncrement = 0;
    private final int id;
    private final String username;
    private final ChatServer chatServer;
    private Message.Memento memento;

    public User(String username, ChatServer chatServer) {
        this.id = ++idIncrement;
        this.username = username;
        this.chatServer = chatServer;
    }

    public String getUsername() {
        return username;
    }

    public int sendMessage(String content, List<User> recipients) {
        Message message = new Message(content, recipients, this);
        this.memento = message.takeSnapshot();
        return this.chatServer.sendMessage(message);
    }

    public void receiveMessage(Message message) {
        System.out.println(getUsername() + " received message: `" + message.getContent() + "` from " + message.getSender()
                .getUsername());
    }

    public boolean undoLastMessage() {
        if (memento == null) {
            System.out.println(getUsername() + " has no message to undo.");
            return false;
        } else {
            Message message = Message.restoreSnapshot(memento);
            return this.chatServer.unsendMessage(message);
        }
    }

    public String getLastMessage() {
        Message message = this.chatServer.getLastMessage(this);
        return String.format("%s sent to %s: %s", getUsername(), message.getRecipients(), message.getContent());
    }

    public Iterator<Message> getChatIterator() {
        return this.chatServer.getChatIterator(this);
```

```
52      }
53
54      public boolean blockUser(User user) {
55          return this.chatServer.blockUser(this, user);
56      }
57
58      public boolean unblockUser(User user) {
59          return this.chatServer.unblockUser(this, user);
60      }
61
62      @Override
63      public boolean equals(Object o) {
64          if (this == o) return true;
65          if (o == null || getClass() != o.getClass()) return false;
66          User user = (User) o;
67          return id == user.id && Objects.equals(username, user.username);
68      }
69
70      @Override
71      public int hashCode() {
72          return Objects.hash(id, username);
73      }
74
75      @Override
76      public String toString() {
77          return username;
78      }
79  }
80
```

```java
package edu.cpp.cs5800.chatapp;


import java.util.Arrays;
import java.util.Iterator;

public class Driver {
    public static void main(String[] args) {
        ChatServer chatServer = new ChatServer();

        User user1 = new User("Jack", chatServer);
        User user2 = new User("Mark", chatServer);
        User user3 = new User("Lucy", chatServer);

        chatServer.registerUser(user1);
        chatServer.registerUser(user2);
        chatServer.registerUser(user3);

        user1.sendMessage("Hello everyone", Arrays.asList(user2, user3));
        user2.sendMessage("Hi " +  user1.getUsername(), Arrays.asList(user1));
        user3.sendMessage("What's up " +  user1.getUsername(), Arrays.asList(user1));
        user3.sendMessage("Hope you all are doing great.", Arrays.asList(user1, user2));
        user3.sendMessage("How was your weekend " +  user2.getUsername() + "?", Arrays.asList(user2));
        printChatHistory(user2);

        System.out.printf("------%s's Last Message------\n", user3.getUsername());
        System.out.println(user3.getLastMessage());
        printChatHistory(user3);

        System.out.println();
        user3.undoLastMessage();
        printChatHistory(user3);

        user3.blockUser(user1);
        user1.sendMessage("Hi " + user3.getUsername(), Arrays.asList(user3));
        System.out.println();

        user3.unblockUser(user1);
        user1.sendMessage("Hi " + user3.getUsername(), Arrays.asList(user3));
        System.out.println();
        printChatHistory(user3);
    }

    private static void printChatHistory(User user) {
        System.out.printf("------%s's Chat History------\n", user.getUsername());
        Iterator<Message> chatIterator= user.getChatIterator();
        while (chatIterator.hasNext()) {
            Message message = chatIterator.next();
            String time = message.getTimestamp().toString();
            String senderName = message.getSender().equals(user) ? user.getUsername(): message.getSender().getUsername();
            String recipient = senderName.equals(user.getUsername()) ? message.
```

```
getRecipients().toString() : user.getUsername();
            String data = String.format("[%s]%s -> %s: %s",
                    time,
                    senderName,
                    recipient,
                    message.getContent());
            System.out.println(data);
        }
        System.out.println();
    }
}

```

```java
package edu.cpp.cs5800.chatapp;

import java.util.Date;
import java.util.List;
import java.util.Objects;

public class Message {
    private static int idIncrement = 0;
    private int id;
    private String content;
    private List<User> recipients;
    private User sender;
    private Date timestamp;

    public Message(String content, List<User> recipients, User sender) {
        this.id = ++idIncrement;
        this.content = content;
        this.recipients = recipients;
        this.sender = sender;
        this.timestamp = new Date();
    }

    private Message(int id, String content, List<User> recipients, User sender, Date timestamp) {
        this.id = id;
        this.content = content;
        this.recipients = recipients;
        this.sender = sender;
        this.timestamp = timestamp;
    }

    public int getId() {
        return id;
    }

    public String getContent() {
        return content;
    }

    public List<User> getRecipients() {
        return recipients;
    }

    public User getSender() {
        return sender;
    }

    public Date getTimestamp() {
        return timestamp;
    }

    public Memento takeSnapshot() {
        return new Memento(this.getId(), this.getContent(), this.getRecipients(), this.
```

```
getSender(), this.getTimestamp());
    }

    public static Message restoreSnapshot(Memento memento) {
        return new Message(memento.getId(), memento.getContent(), memento.getRecipients
(), memento.getSender(), memento.getTimestamp());
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Message message = (Message) o;
        return id == message.id;
    }

    @Override
    public int hashCode() {
        return Objects.hashCode(id);
    }

    @Override
    public String toString() {
        return "Message{" +
                "id=" + id +
                ", content='" + content + '\'' +
                ", recipients=" + recipients +
                ", sender=" + sender +
                ", timestamp=" + timestamp +
                '}';
    }

    public static class Memento {
        private int id;
        private String content;
        private List<User> recipients;
        private User sender;
        private Date timestamp;

        public Memento(int id,
                               String content,
                               List<User> recipients,
                               User sender,
                               Date timestamp) {
            this.id = id;
            this.content = content;
            this.recipients = recipients;
            this.sender = sender;
            this.timestamp = timestamp;
        }

        public int getId() {
            return id;
```

```
104         }
105
106         public String getContent() {
107             return content;
108         }
109
110         public List<User> getRecipients() {
111             return recipients;
112         }
113
114         public User getSender() {
115             return sender;
116         }
117
118         public Date getTimestamp() {
119             return timestamp;
120         }
121     }
122 }
123
```

```java
package edu.cpp.cs5800.chatapp;


import java.util.*;

public class ChatServer {
    private Set<User> users = new HashSet<>();
    private Map<User, Set<User>> blockedUsers = new HashMap<>();
    private ChatHistory chatHistory = new ChatHistory();

    public boolean registerUser(User user) {
        System.out.println(user.getUsername() + " has been registered!");
        return users.add(user);
    }

    public boolean unregisterUser(User user) {
        System.out.println(user.getUsername() + " has been unregistered!");
        return users.remove(user);
    }

    public boolean blockUser(User blocker, User blockee) {
        if (!blockedUsers.containsKey(blocker)) {
            blockedUsers.put(blocker, new HashSet<>());
        }
        System.out.println(blocker.getUsername() + " has blocked " + blockee.getUsername() + "!");
        return blockedUsers.get(blocker).add(blockee);
    }

    public boolean unblockUser(User blocker, User blockee) {
        System.out.println(blocker.getUsername() + " has unblocked " + blockee.getUsername() + "!");
        return blockedUsers.get(blocker) != null && blockedUsers.get(blocker).remove(blockee);
    }


    public int sendMessage(Message message) {
        User sender = message.getSender();
        if (!users.contains(sender)) {
            System.out.println("Failed to send message, " + sender.getUsername() + "  is not registered!");
            return 0;
        }

        int count = 0;
        for (User recipient : message.getRecipients()) {
            if (!users.contains(recipient)) {
                System.out.println("Failed to send message, " + recipient.getUsername() + " is not registered!");
            } else if (blockedUsers.get(recipient) != null && blockedUsers.get(recipient).contains(sender)) {
                System.out.println("Failed to send message, " + sender.getUsername() + " is
```

```
 blocked by " + recipient.getUsername() + "!");
            } else {
                System.out.println(sender.getUsername() + " sent message: `" + message.getContent() + "` to " + recipient.getUsername());
                recipient.receiveMessage(message);

                count++;
            }
        }
        if (count > 0) {
            this.chatHistory.addMessage(message);
        }
        return count;
    }

    public boolean unsendMessage(Message message) {
        if (this.chatHistory.removeMessage(message)) {
            System.out.println(message.getSender()
                    .getUsername() + " unsent message: `" + message.getContent() + "` to " + message.getRecipients());
            return true;
        } else {
            System.out.println(message.getSender()
                    .getUsername() + " unsent message failed " + message.getContent() + " to " + message.getRecipients());
            return false;
        }

    }

    public Iterator<Message> getChatIterator(User user) {
        return this.chatHistory.iterator(user);
    }

    public Message getLastMessage(User user) {
        return this.chatHistory.getLastMessage(user);
    }
}

```

```java
package edu.cpp.cs5800.chatapp;

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

public class ChatHistory implements IterableByUser {
    private List<Message> history = new ArrayList<>();

    public boolean addMessage(Message message) {
        return history.add(message);
    }

    public Message getLastMessage(User user) {
        return history
                .stream()
                .filter(message -> message.getSender().equals(user))
                .reduce((m1, m2) -> m2)
                .orElse(null);
    }

    public List<Message> getHistory(User user) {
        return history.stream()
                .filter(
                        message -> message.getSender().equals(user)
                                || message.getRecipients().contains(user)
                ).toList();
    }

    public boolean removeMessage(Message message) {
        return history.remove(message);
    }

    @Override
    public Iterator<Message> iterator(User userToSearchWith) {
        return new ChatIterator(getHistory(userToSearchWith));
    }
}

```

```java
package edu.cpp.cs5800.chatapp;

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

public class ChatIterator implements Iterator<Message> {
    private List<Message> messages;
    private int currentIndex;

    public ChatIterator(List<Message> userMessages) {
        this.messages = userMessages;
        this.currentIndex = 0;
    }

    @Override
    public boolean hasNext() {
        return this.messages.size() > this.currentIndex;
    }

    @Override
    public Message next() {
        return this.messages.get(this.currentIndex++);
    }
}

```

```java
package edu.cpp.cs5800.chatapp;

import java.util.Iterator;

public interface IterableByUser {
    Iterator<Message> iterator(User userToSearchWith);
}

```