Repo: https://github.com/Cloudsted/CS4800-Homework6

Code:

```
import java.util.Iterator;

public class User implements IterableByUser {
  private String username;
  private ChatServer server;
  private ChatHistory history;

  public User(String username, ChatServer server) {
    this.username = username;
    this.server = server;
    this.history = new ChatHistory();
  }

  public void sendMessage(String[] recipients, String content) {
    Message message = new Message(this.username, recipients, content);
    server.sendMessage(this, message);
    history.addMessage(message);
  }

  public void receiveMessage(Message message) {
    System.out.println("Received by " + this.username + ": " + message);
  }

  public void undoLastMessage() {
    MessageMemento memento = history.saveToMemento();
    server.undoMessage(this, memento);
  }

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

	public String getUsername() {
		return username;
	}

	public ChatServer getServer() {
		return server;
	}

	public ChatHistory getHistory() {
		return history;
	}
}import java.util.Iterator;
import java.util.List;
import java.util.NoSuchElementException;

public class searchMessagesByUser implements Iterator<Message> {
  private Iterator<Message> iterator;
  private User userToSearchWith;
  private Message nextMessage;
  private boolean hasNextCalled = false;

  public searchMessagesByUser(User userToSearchWith, List<Message> messages) {
        this.userToSearchWith = userToSearchWith;
        this.iterator = messages.iterator();
    }

    @Override
    public boolean hasNext() {
        if (hasNextCalled) {
            return nextMessage != null;
        }
        hasNextCalled = true;
        while (iterator.hasNext()) {
            Message current = iterator.next();
            if (current.getSender().equals(userToSearchWith.getUsername()) ||
                        recipientsContainsUsername(current.getRecipients(),
userToSearchWith.getUsername())) {
                nextMessage = current;
                return true;
            }
        }
        nextMessage = null;
        return false;
    }

    @Override
    public Message next() {
        if (!hasNextCalled || nextMessage == null) {
            throw new NoSuchElementException();
        }
        hasNextCalled = false;
        return nextMessage;
    }

    private boolean recipientsContainsUsername(String[] recipients, String username) {
            for(String recipientUsername : recipients) {
                    if(recipientUsername.equals(username)) {
                            return true;
                    }
            }

            return false;
    }
}import java.time.LocalDateTime;

public class MessageMemento {
    private String content;
    private LocalDateTime timestamp;

    public MessageMemento(String content, LocalDateTime timestamp) {
        this.content = content;
        this.timestamp = timestamp;
    }

    public String getContent() {
        return content;
    }

    public LocalDateTime getTimestamp() {
        return timestamp;
    }
}import java.time.LocalDateTime;

public class Message {
    private String sender;
    private String[] recipients;
    private LocalDateTime timestamp;
    private String content;

    public Message(String sender, String[] recipients, String content) {
        this.sender = sender;
        this.recipients = recipients;
        this.content = content;
        this.timestamp = LocalDateTime.now();
    }

    public String getSender() {
        return sender;
    }

    public String[] getRecipients() {
        return recipients;
    }

    public LocalDateTime getTimestamp() {
        return timestamp;
    }

    public String getContent() {
        return content;
    }

    @Override
    public String toString() {
        return "From: " + sender + ", Message: " + content + ", At: " + timestamp;
    }
}import java.util.Iterator;

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

        User alice = new User("Alice", server);
        User bob = new User("Bob", server);
        User charlie = new User("Charlie", server);

        server.registerUser(alice);
        server.registerUser(bob);
        server.registerUser(charlie);

        alice.sendMessage(new String[] {"Bob", "Charlie"}, "Hello everyone!");
        bob.sendMessage(new String[] {"Alice"}, "Hello Alice!");
        charlie.sendMessage(new String[] {"Alice"}, "Hey Alice, how are you?");

        alice.undoLastMessage();

        Iterator<Message> it = alice.iterator(bob);
        while (it.hasNext()) {
            Message message = it.next();
            System.out.println("Iterated message: " + message);
        }
    }
}import java.util.Iterator;

public interface IterableByUser {
    Iterator<Message> iterator(User userToSearchWith);
}import java.util.HashMap;
import java.util.Map;

public class ChatServer {
    private Map<String, User> users = new HashMap<>();

    public void registerUser(User user) {
        users.put(user.getUsername(), user);
    }

    public void unregisterUser(User user) {
        users.remove(user.getUsername());
    }

    public void sendMessage(User sender, Message message) {
        for (String recipient : message.getRecipients()) {
            if (users.containsKey(recipient)) {
                users.get(recipient).receiveMessage(message);
            }
        }
    }
    public void undoMessage(User sender, MessageMemento memento) {
        if (memento != null) {
            Message toUndo = new Message(sender.getUsername(), null, memento.getContent());
        }
    }
}import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

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

    public void addMessage(Message message) {
        messages.add(message);
    }

    public Message getLastMessage() {
        if (!messages.isEmpty()) {
            return messages.get(messages.size() - 1);
        }
        return null;
    }

    public MessageMemento saveToMemento() {
        Message lastMessage = getLastMessage();
        if (lastMessage != null) {
            return new MessageMemento(lastMessage.getContent(), lastMessage.getTimestamp());
        }
        return null;
    }

    public void restoreFromMemento(MessageMemento memento) {
        if (memento != null) {
            Message lastMessage = getLastMessage();
            if (lastMessage != null && lastMessage.getTimestamp().equals(memento.getTimestamp())) {
                messages.remove(lastMessage);
            }
        }
    }

    @Override
    public Iterator<Message> iterator(User userToSearchWith) {
        return new searchMessagesByUser(userToSearchWith, messages);
    }
}
```