**Homework 5**

**Github:** <https://github.com/Dubs2310/CS5800-Homework-5.git>

* MessageMemento.java

public class MessageMemento {  
 private String timestamp;  
 private String messageContent;  
  
 public MessageMemento(String timestamp, String messageContent) {  
 this.timestamp = timestamp;  
 this.messageContent = messageContent;  
 }  
  
 public String getTimestamp() { return this.timestamp; }  
 public String getMessageContent() { return this.messageContent; }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (obj == null) return false;  
 if (!(obj instanceof MessageMemento memento)) return false;  
 return memento.getTimestamp().equals(timestamp) && memento.getMessageContent().equals(messageContent);  
 }  
}

* Message.java

import java.util.List;  
  
public class Message {  
 private User sender;  
 private List<User> recipients;  
 private String timestamp;  
 private String messageContent;  
  
 public Message(User sender, List<User> recipients, String timestamp, String messageContent) {  
 this.sender = sender;  
 this.recipients = recipients;  
 this.timestamp = timestamp;  
 this.messageContent = messageContent;  
 }  
  
 public User getSender() { return this.sender; }  
 public List<User> getRecipients() { return this.recipients; }  
 public String getTimestamp() { return this.timestamp; }  
 public String getMessageContent() { return this.messageContent; }  
  
 public MessageMemento createSnapshot() { return new MessageMemento(this.timestamp, this.messageContent); }  
}

* SearchMessagesByUserIterator.java

import java.util.Iterator;  
import java.util.List;  
  
public class SearchMessagesByUserIterator implements Iterator<Message> {  
 private int index;  
 private int size;  
 private User user;  
 private List<Message> messages;  
 private User userToSearchFor;  
  
 public SearchMessagesByUserIterator(User userToSearchFor, ChatHistory history) {  
 this.userToSearchFor = userToSearchFor;  
 this.user = history.getUser();  
 this.messages = history.getMessages();  
 this.index = 0;  
 this.size = messages.size();  
 }  
  
 @Override  
 public boolean hasNext() {  
 Message message = null;  
 while (index < size) {  
 message = messages.get(index);  
 if (  
 (message.getSender().equals(user) && message.getRecipients().contains(userToSearchFor)) ||  
 message.getSender().equals(userToSearchFor)  
 )  
 return true;  
 index++;  
 }  
 return false;  
 }  
  
 @Override  
 public Message next() {  
 if (hasNext())  
 return messages.get(index++);  
 return null;  
 }  
  
 @Override  
 public void remove() {  
 Iterator.super.remove();  
 }  
}

* IterableByUser.java

import java.util.Iterator;  
  
public interface IterableByUser {  
 Iterator iterator(User userToSearchWith);  
}

* User.java

import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Iterator;  
import java.util.List;  
  
public class User implements IterableByUser {  
 private String name;  
 private ChatServer server;  
  
 public User(String name, ChatServer server) {  
 this.name = name;  
 this.server = server;  
 }  
  
 public String getName() { return this.name; }  
 public ChatServer getServer() { return server; }  
  
 public void sendMessage(List<User> recipients, String messageContent) {  
 String timestamp = new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new Date());  
 Message message = new Message(this, recipients, timestamp, messageContent);  
 server.sendMessage(message);  
 }  
  
 public void receiveMessage(Message message) {  
 server.updateChatHistory(this, message);  
 }  
  
 public void blockUser(User toBlock) {  
 server.blockUser(this, toBlock);  
 }  
  
 public void unblockUser(User toUnblock) {  
 server.unblockUser(this, toUnblock);  
 }  
  
 public void undoLastMessageSent() {  
 server.undoLastMessageSent(this);   
 }  
  
 public void viewCompleteChatHistory() {  
 server.viewCompleteChatHistory(this);  
 }  
  
 @Override  
 public Iterator iterator(User userToSearchWith) {  
 return server.getUserChatHistory(this).iterator(userToSearchWith);  
 }  
  
 public void viewChatHistoryWithUser(User user) {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\* " + name + "'s Chat History with " + user.getName() + " \*\*\*\*\*\*\*\*\*\*\*");  
 SearchMessagesByUserIterator myMessages = (SearchMessagesByUserIterator) iterator(user);  
 while (myMessages.hasNext()) {  
 Message message = myMessages.next();  
 String senderName = message.getSender().getName();  
 System.*out*.println(  
 "[" + message.getTimestamp() + "] " +  
 (senderName.equals(name) ? "You" : senderName) + ": " +  
 message.getMessageContent()  
 );  
 }  
 System.*out*.println();  
 }  
}

* ChatHistory.java

import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
import java.util.Stack;  
  
public class ChatHistory implements IterableByUser {  
 private User user;  
 private List<Message> messages;  
 private Stack<MessageMemento> messagesSent;  
  
 public ChatHistory(User user) {  
 this.user = user;  
 this.messages = new ArrayList<>();  
 this.messagesSent = new Stack<>();  
 }  
  
 public User getUser() { return user; }  
 public List<Message> getMessages() { return messages; }  
  
 public void addMessage(Message message, boolean isSender) {  
 messages.add(message);  
 if (isSender) messagesSent.add(message.createSnapshot());  
 }  
   
 public Message getLastMessageSent() {  
 if (!messagesSent.empty())  
 for (Message message : messages)  
 if (message.createSnapshot().equals(messagesSent.peek()))  
 return message;  
 return null;  
 }  
  
 public Message getLastMessage() {  
 if (messages.size() == 0) return null;  
 return messages.get(messages.size() - 1);  
 }  
  
 public void deleteMessage(Message message) {  
 if (messages.size() == 0) return;  
 int index = messages.lastIndexOf(message);  
 messages.set(  
 index,  
 new Message(message.getSender(), message.getRecipients(), message.getTimestamp(), "[\_\_MESSAGE\_DELETED\_\_]")  
 );  
 }  
  
 public void undoLastMessageSent() {  
 if (messages.size() == 0) return;  
 deleteMessage(getLastMessageSent());  
 messagesSent.pop();  
 }  
  
 public void viewCompleteChatHistory() {  
 for (Message message : messages)  
 System.*out*.println(  
 "[" + message.getTimestamp() + "] " +  
 (messagesSent.contains(message.createSnapshot()) ? "You" : message.getSender().getName()) + ": " +  
 message.getMessageContent()  
 );  
 System.*out*.println();  
 }  
  
 @Override  
 public Iterator iterator(User userToSearchWith) {  
 return new SearchMessagesByUserIterator(userToSearchWith, this);  
 }  
}

* ChatServer.java

import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class ChatServer {  
 private Map<User, ChatHistory> chatHistories;  
 private Map<User, List<User>> blockedUsers;  
  
 public ChatServer() {  
 chatHistories = new HashMap<>();  
 blockedUsers = new HashMap<>();  
 }  
  
 public void registerUser(User user) {  
 chatHistories.put(user, new ChatHistory(user));  
 blockedUsers.put(user, new ArrayList<>());  
 }  
  
 public void unregisterUser(User user) {  
 chatHistories.remove(user);  
 blockedUsers.remove(user);  
 }  
  
 public void sendMessage(Message message) {  
 User sender = message.getSender();  
 chatHistories.get(sender).addMessage(message, true);  
   
 // only recipients that haven't blocked the user receive the message  
 List<User> recipients = message.getRecipients();  
 for (User user: recipients)  
 if (!blockedUsers.get(user).contains(sender))  
 user.receiveMessage(message);  
 else  
 user.receiveMessage(new Message(message.getSender(), message.getRecipients(), message.getTimestamp(), "[\_\_BLOCKED\_MESSAGE\_\_]"));  
 }  
  
 public void updateChatHistory(User user, Message message) {  
 chatHistories.get(user).addMessage(message, false);  
 }  
  
 public void blockUser(User blocking, User beingBlocked) {  
 blockedUsers.get(blocking).add(beingBlocked);  
 }  
  
 public void unblockUser(User unblocking, User beingUnblocked) {  
 blockedUsers.get(unblocking).remove(beingUnblocked);  
 }  
  
 public void undoLastMessageSent(User user) {  
 Message lastMessageSent = chatHistories.get(user).getLastMessageSent();  
 chatHistories.get(user).undoLastMessageSent();  
  
 for (User u: chatHistories.keySet())  
 if (u != user) chatHistories.get(u).deleteMessage(lastMessageSent);  
 }  
  
 public void viewCompleteChatHistory(User user) {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\* " + user.getName() + "'s Complete Chat History \*\*\*\*\*\*\*\*\*\*\*");  
 chatHistories.get(user).viewCompleteChatHistory();  
 }  
  
 public ChatHistory getUserChatHistory(User user) {  
 return chatHistories.get(user);  
 }  
}

* Main.java

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Main {  
 public static void dialogueSoFar(User john, User sam, User dean) {  
 john.viewCompleteChatHistory();  
 dean.viewCompleteChatHistory();  
 sam.viewCompleteChatHistory();  
 }  
  
 public static void main(String[] args) {  
 ChatServer server = new ChatServer();  
  
 User john = new User("John", server);  
 User sam = new User("Sam", server);  
 User dean = new User("Dean", server);  
  
 server.registerUser(john);  
 server.registerUser(sam);  
 server.registerUser(dean);  
  
 System.*out*.println("----------------------------------------------------- ACT 1: THE BLOCK -----------------------------------------------------\n");  
  
 john.sendMessage(  
 new ArrayList<>(List.*of*(sam)),  
 "Sam, monster hunting is the family business. Do it for your mother."  
 );  
 sam.blockUser(john);  
 john.sendMessage(  
 new ArrayList<>(List.*of*(sam)),  
 "Did you really just block me?!"  
 );  
 john.sendMessage(  
 new ArrayList<>(List.*of*(dean)),  
 "Dean, tell Sam to get his head out of his a\*\* and unblock me! I'm not done with him!"  
 );  
 *dialogueSoFar*(john, sam, dean);  
  
 System.*out*.println("----------------------------------------------------- ACT 2: THE UNBLOCK AND APOLOGY -----------------------------------------------------\n");  
  
 dean.sendMessage(  
 new ArrayList<>(Arrays.*asList*(john, sam)),  
 "Hey Sammy, just unblock your old man. He's after me now... 😒"  
 );  
 sam.unblockUser(john);  
 john.sendMessage(  
 new ArrayList<>(Arrays.*asList*(sam, dean)),  
 "Okay Sam, I'm sorry..."  
 );  
  
 *dialogueSoFar*(john, sam, dean);  
  
 System.*out*.println("----------------------------------------------------- ACT 3: THE UNDO -----------------------------------------------------\n");  
  
 john.undoLastMessageSent();  
 sam.sendMessage(  
 new ArrayList<>(Arrays.*asList*(john, dean)),  
 "Woah, did you just apologize?! 😳"  
 );  
 john.sendMessage(  
 new ArrayList<>(Arrays.*asList*(sam, dean)),  
 "I did, but looks like you two don't deserve it. Thank your elder brother for his smarta\*\* comment"  
 );  
 *dialogueSoFar*(john, sam, dean);  
  
 System.*out*.println("----------------------------------------------------- TESTING ITERATOR -----------------------------------------------------\n");  
 john.viewChatHistoryWithUser(sam);  
 john.viewChatHistoryWithUser(dean);  
 }  
}

* Output

**A picture containing text, screenshot, font

Description automatically generated**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**