

Selected files

8 printable files

src\main\java\org\example\ChatHistory.java
src\main\java\org\example\ChatServer.java
src\main\java\org\example\IterableByUser.java
src\main\java\org\example\Main.java
src\main\java\org\example\Message.java
src\main\java\org\example\MessageMemento.java
src\main\java\org\example\searchMessagesByUser.java
src\main\java\org\example\User.java

src\main\java\org\example\ChatHistory.java

```
1  package org.example;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.Iterator;
6  import java.util.Map;
7
8  //A class that stores the chat history for a user. It should have methods to add a new
9  //message to the history and get the last message sent
10 public class ChatHistory implements IterableByUser{
11     // private ArrayList<String> savedMessages;
12     private HashMap<String, String> savedMessages;
13     MessageMemento messageMemento = new MessageMemento(savedMessages);
14     public ChatHistory(){
15         this.savedMessages = new HashMap<>();
16     }
17
18     public HashMap<String, String> getSavedMessages() {return this.savedMessages;}
19     public void saveMessage(String username, String message)
20     {
21         String newMessage;
22
23         //get previous chat history of user and append new message to save
24         if (savedMessages.get(username) == null)
25         {
26             newMessage = message;
27         }
28         else {
29
30             newMessage = message + "\n" + savedMessages.get(username);
31         }
32         savedMessages.put(username, newMessage );
33
34
35     }
36
37     public void saveMessageMemento()
38     {
39         //save new memento state
40         messageMemento.setState(savedMessages);
```

```

41         System.out.println("*****Save message memento*****");
42
43     }
44     public void undoMessageMemento()
45     {
46         savedMessages = messageMemento.getState();
47     }
48
49     public MessageMemento getMessageMemento()
50     {
51         return messageMemento;
52     }
53
54     public String printChatHistoryFromUser( String username ) {
55         if (savedMessages.get(username) == null)
56             return "No messages from " + username;
57         else {
58             return savedMessages.get(username);
59         }
60     }
61     @Override
62     public String toString()
63     {
64         if( savedMessages.isEmpty() )
65             return "Chat History is empty";
66
67         //getEntireChatHistory()
68         String messageHistory = "";
69         for( Map.Entry<String, String> m : savedMessages.entrySet() )
70             messageHistory += m.getValue() + "\n";
71         return messageHistory;
72     }
73
74
75
76
77     public Iterator iterator(User userToSearchWith) {
78         return new searchMessagesByUser(userToSearchWith);
79     }
80 }
81

```

src\main\java\org\example\ChatServer.java

```

1  package org.example;
2
3  import java.util.ArrayList;
4
5  //The mediator class that manages communication between users. It should have methods to
6  //register and unregister users, send messages from one user to one or more other users, and
7  //block
8  //messages from specific users
9  public class ChatServer {
10     private ArrayList<User> activeUsers, blockedUsers;
11     private Message message;

```

```
11     public ChatServer()  
12     {  
13         this.activeUsers = new ArrayList<>();  
14         this.blockedUsers = new ArrayList<>();  
15     }  
16  
17     public void setMessage(Message message)  
18     {  
19         this.message = message;  
20     }  
21  
22  
23  
24     public void sendMessage(){  
25  
26         System.out.println("Sending messages to users...");  
27         User sender = message.getSender();  
28         sender.receiveMessage(sender.getUserName(), message.getFormattedMessage());  
29  
30         if(activeUsers.contains( sender ) ) {  
31             ArrayList<User> usersToReceiveMessage = message.getRecipients();  
32  
33             for (User user : usersToReceiveMessage){  
34                 if( blockedUsers.contains( user ) ) {  
35                     System.out.println("User is blocked from receiving messages." );  
36                 } else{  
37                     user.receiveMessage(sender.getUserName(),  
message.getFormattedMessage());  
38                 }  
39             }  
40         } else if ( blockedUsers.contains( sender ) ) {  
41             System.out.println("User is blocked from sending messages." );  
42  
43         } else {  
44             System.out.println("Message not sent to recipients. " + message.getSender()  
.getUserName() + " is not a registered user.");  
45  
46         }  
47     }  
48  
49     //can send messages  
50     public void registerUser(User user){  
51         activeUsers.add(user);  
52         System.out.println("Registered user " + user.getUserName());  
53     }  
54  
55     public void unregisterUser(User user){  
56         activeUsers.remove(user);  
57         System.out.println("Unregistered user " + user.getUserName());  
58     }  
59  
60     //can not send messages  
61     public void blockUser(User user){  
62         blockedUsers.add(user);  
63         System.out.println("Blocked user " + user.getUserName());  
64     }  
65
```

```
66     public void unblockUser(User user){
67         blockedUsers.remove(user);
68         System.out.println("Unblocked user " + user.getUserName());
69     }
70 }
71
```

src\main\java\org\example\IterableByUser.java

```
1  package org.example;
2
3  import java.util.Iterator;
4
5  public interface IterableByUser {
6      Iterator iterator(User userToSearchWith);
7  }
8
```

src\main\java\org\example\Main.java

```
1  package org.example;
2
3  import java.util.ArrayList;
4  import java.util.Iterator;
5
6  public class Main {
7      public static void main(String[] args) {
8          ChatServer chatServer = new ChatServer();
9
10         User person1 = new User("Amanda");
11         User person2 = new User("Alice");
12         User person3 = new User("Bob");
13         User person4 = new User("Eve");
14         User person5 = new User("John");
15
16         chatServer.registerUser(person1);
17         chatServer.registerUser(person2);
18         chatServer.registerUser(person3);
19         chatServer.registerUser(person4);
20         chatServer.registerUser(person5);
21
22         ArrayList<User> person1SendTo = new ArrayList<>();
23         person1SendTo.add(person2);
24         person1SendTo.add(person3);
25         person1SendTo.add(person4);
26         person1SendTo.add(person5);
27         ArrayList<User> person2SendTo = new ArrayList<>();
28         person2SendTo.add(person1);
29         person2SendTo.add(person3);
30         person2SendTo.add(person4);
31         person2SendTo.add(person5);
32         ArrayList<User> person4SendTo = new ArrayList<>();
33         person4SendTo.add(person1);
```

```
34     person4SendTo.add(person3);
35     person4SendTo.add(person2);
36     person4SendTo.add(person5);
37
38     System.out.println("Users can send messages to one or more other users through the
chat server.\n=====");
39     Message person1Message1 = new Message(person1, person1SendTo, "15:47:14", "Hey guys,
anyone want to hangout this weekend?");
40     Message person1Message2 = new Message(person1, person1SendTo, "15:47:25", "Let's do
something fun!");
41
42     //     person1Message1.printMessage();
43     chatServer.setMessage(person1Message1);
44     chatServer.sendMessage();
45
46     //     person1Message2.printMessage();
47     chatServer.setMessage(person1Message2);
48     chatServer.sendMessage();
49
50     Message person2Message1 = new Message(person2, person2SendTo, "15:49:52", "I'm not
available this weekend. Remove me please, thank you.");
51     //     person2Message1.printMessage();
52     chatServer.setMessage(person2Message1);
53     chatServer.sendMessage();
54
55     System.out.println("\nUsers can block messages from specific users using the Mediator
design pattern.\n=====");
56     chatServer.blockUser(person2);
57     Message person4Message1 = new Message(person4, person4SendTo, "15:55:12", "I blocked
them from the chat.");
58     Message person4Message2 = new Message(person4, person4SendTo, "15:55:21", "I'm free
both days.");
59     Message person4Message3 = new Message(person4, person4SendTo, "15:55:25", "We should
go bowling.");
60
61     //     person4Message1.printMessage();
62     chatServer.setMessage(person4Message1);
63     chatServer.sendMessage();
64
65     //     person4Message2.printMessage();
66     chatServer.setMessage(person4Message2);
67     chatServer.sendMessage();
68
69     System.out.println("\nUsers can undo the last message they sent using the Memento
design pattern.\n=====");
70     person4.backupMessages();
71     System.out.println( person4.getEntireChatHistory().getMessageMemento());
72     System.out.println( person4.getEntireChatHistory());
73
74     //     person4Message3.printMessage();
75     chatServer.setMessage(person4Message3);
76     chatServer.sendMessage();
77
78     person4.undoLastMessage();
79     System.out.println( person4.getEntireChatHistory().getMessageMemento());
80     System.out.println( person4.getEntireChatHistory() );
81
82
```

```

83     Message person4Message4 = new Message(person4, person4SendTo, "15:55:25", "We should
go roller skating.");
84
85     //     person4Message4.printMessage();
86     chatServer.setMessage(person4Message4);
87     chatServer.sendMessage();
88
89     System.out.println("\nUsers can receive messages from other users and view the chat
history for a specific user.\n=====");
90
91     System.out.println("\nGetting entire chat history of user person5...");
92     System.out.println( person5.getEntireChatHistory() );
93
94     System.out.println("\nGetting chat history of person5 from person1...");
95     System.out.println( person4.printChatHistoryFromUser( person1.getUserName() ) );
96
97     System.out.println("\nUsers can iterate through previous sent messages from a
specified user.\n=====");
98     Iterator searchP1P3 = person2.iterator(person1);
99     while( searchP1P3.hasNext() ){
100         System.out.println( searchP1P3.next() );
101     }
102 }
103 }

```

src\main\java\org\example\Message.java

```

1  package org.example;
2
3  import java.util.ArrayList;
4
5  //A class representing a message sent by a user. It should have properties for the sender,
6  //recipient(s), timestamp, and message content
7  public class Message {
8      private User sender;
9      private ArrayList<User> recipients;
10     private String timestamp, message;
11
12
13     public Message( User sender, ArrayList<User> recipients, String timestamp, String message)
14     {
15         this.sender = sender;
16         this.recipients = recipients;
17         this.timestamp = timestamp;
18         this.message = message;
19     }
20
21     public ArrayList<User> getRecipients() { return recipients;}
22     public User getSender() { return sender;}
23     public String getMessage() { return this.message;}
24     public String getFormattedMessage() { return "<" + this.sender.getUserName() + ">\t[" +
this.timestamp + "]\t" + this.message;}
25     public void printMessage() { System.out.println(this);}
26
27     @Override
28     public String toString()

```

```

29     {
30         String formattedMessage = "\n";
31
32         formattedMessage += "Timestamp: " + this.timestamp + "\nFrom: <" +
this.sender.getUserName() + ">\nTo: ";
33         for(User user: this.getRecipients() )
34             formattedMessage += "<" +user.getUserName() + "> ";
35         formattedMessage += "\n" + this.message + "\n";
36
37         return formattedMessage;
38     }
39 }
40

```

src\main\java\org\example\MessageMemento.java

```

1  package org.example;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  //A class that represents a snapshot of a message sent by a user. It should have
7  //properties for the message content and timestamp
8  public class MessageMemento {
9      private HashMap<String, String> state;
10
11      public MessageMemento(HashMap<String, String> state)
12      {
13          this.state = state;
14      }
15
16      public HashMap<String, String> getState()
17      {
18          return state;
19      }
20
21      public void setState(HashMap<String, String> state)
22      {
23          this.state = state;
24      }
25
26      @Override
27      public String toString()
28      {
29          System.out.println( "\n*****\nmessage memento state\n*****" );
30
31          if( state.isEmpty() )
32              return "empty";
33
34          //getEntireChatHistory()
35          String messageHistory = "";
36          for( Map.Entry<String, String> m : state.entrySet() )
37              messageHistory += m.getValue() + "\n";
38          return messageHistory;
39

```

```
40     }
41
42 }
43
```

src\main\java\org\example\searchMessagesByUser.java

```
1  package org.example;
2
3  import java.util.ArrayList;
4  import java.util.Arrays;
5  import java.util.Iterator;
6  import java.util.List;
7
8  public class searchMessagesByUser implements Iterator {
9      private String userName;
10     private List<String> chatHistoryFromUser;
11     private int indexInChatHistory;
12     private int chatHistoryFromUserSize;
13
14     public searchMessagesByUser(User userToSearch){
15         this.userName = userToSearch.getUserName();
16
17         //history in form "message\nanother message\nlast message\n"
18         String allMessages = userToSearch.getEntireChatHistory().getSavedMessages()
19         .get(this.userName);
20         String[] messages = allMessages.split("\\r?\\n|\\r");
21         this.chatHistoryFromUser = Arrays.asList(messages);
22
23         this.indexInChatHistory = 0;
24         this.chatHistoryFromUserSize = chatHistoryFromUser.size();
25     }
26
27     @Override
28     public boolean hasNext() {
29         String message = null;
30
31         while( indexInChatHistory < chatHistoryFromUserSize ) {
32             message = chatHistoryFromUser.get(indexInChatHistory);
33             if( message != null) {
34                 return true;
35             }
36             else {
37                 indexInChatHistory++;
38             }
39         }
40
41         return false;
42     }
43
44     @Override
45     public String next() {
46         if( hasNext()){
47             return chatHistoryFromUser.get( indexInChatHistory++ );
48         }
49     }
50 }
```



```
48         return null;
49     }
50
51     @Override
52     public void remove() {
53         Iterator.super.remove();
54     }
55 }
56
```

src\main\java\org\example\User.java

```
1  package org.example;
2
3  import java.util.Iterator;
4
5  //A class representing a user of the chat application. It should have methods to send messages
6  //to
7  //other users, receive messages from other users, and undo the last message sent.
8  //NOTE: You will NOT communicate with other Users directly you will use Mediator!
9  public class User implements IterableByUser{
10     private String userName;
11     private ChatHistory chatHistory;
12
13     public User(String userName){
14         this.userName = userName;
15         this.chatHistory = new ChatHistory();
16     }
17
18     public String getUserName(){ return this.userName;}
19     public ChatHistory getEntireChatHistory(){return this.chatHistory;}
20     public String printChatHistoryFromUser( String username ){
21         return this.chatHistory.printChatHistoryFromUser(username);
22     }
23
24     public void receiveMessage(String senderUsername, String message)
25     {
26         chatHistory.saveMessage(senderUsername, message);
27         System.out.println(this.userName + ": Message received!");
28     }
29
30     public void undoLastMessage()
31     {
32         chatHistory.undoMessageMemento();
33         System.out.println("Last message undone!");
34     }
35
36     public void backupMessages()
37     {
38         chatHistory.saveMessageMemento();
39         System.out.println("Messages backed up successfully!");
40     }
41
42     @Override
43     public Iterator iterator(User userToSearchWith) {
```

```
43 |         return this.chatHistory.iterator(userToSearchWith);
44 |     }
45 | }
46 |
```