

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;
2
3 import java.util.Iterator;
4 import java.util.List;
5 import java.util.Objects;
6
7 public class User {
8     private static int idIncrement = 0;
9     private final int id;
10    private final String username;
11    private final ChatServer chatServer;
12    private Message.Memento memento;
13
14    public User(String username, ChatServer chatServer) {
15        this.id = ++idIncrement;
16        this.username = username;
17        this.chatServer = chatServer;
18    }
19
20    public String getUsername() {
21        return username;
22    }
23
24    public int sendMessage(String content, List<User> recipients) {
25        Message message = new Message(content, recipients, this);
26        this.memento = message.takeSnapshot();
27        return this.chatServer.sendMessage(message);
28    }
29
30    public void receiveMessage(Message message) {
31        System.out.println(getUsername() + " received message: `" + message.getContent() +
32        "` from " + message.getSender().getUsername());
33    }
34
35    public boolean undoLastMessage() {
36        if (memento == null) {
37            System.out.println(getUsername() + " has no message to undo.");
38            return false;
39        } else {
40            Message message = Message.restoreSnapshot(memento);
41            return this.chatServer.unsendMessage(message);
42        }
43    }
44
45    public String getLastMessage() {
46        Message message = this.chatServer.getLastMessage(this);
47        return String.format("%s sent to %s: %s", getUsername(), message.getRecipients(),
48        message.getContent());
49    }
50
51    public Iterator<Message> getChatIterator() {
52        return this.chatServer.getChatIterator(this);
53    }
54}
```

```
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
```

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;
2
3
4 import java.util.Arrays;
5 import java.util.Iterator;
6
7 public class Driver {
8     public static void main(String[] args) {
9         ChatServer chatServer = new ChatServer();
10
11         User user1 = new User("Jack", chatServer);
12         User user2 = new User("Mark", chatServer);
13         User user3 = new User("Lucy", chatServer);
14
15         chatServer.registerUser(user1);
16         chatServer.registerUser(user2);
17         chatServer.registerUser(user3);
18
19         user1.sendMessage("Hello everyone", Arrays.asList(user2, user3));
20         user2.sendMessage("Hi " + user1.getUsername(), Arrays.asList(user1));
21         user3.sendMessage("What's up " + user1.getUsername(), Arrays.asList(user1));
22         user3.sendMessage("Hope you all are doing great.", Arrays.asList(user1, user2));
23         user3.sendMessage("How was your weekend " + user2.getUsername() + "?", Arrays.
asList(user2));
24         printChatHistory(user2);
25
26         System.out.printf("-----%s's Last Message-----\n", user3.getUsername());
27         System.out.println(user3.getLastMessage());
28         printChatHistory(user3);
29
30         System.out.println();
31         user3.undoLastMessage();
32         printChatHistory(user3);
33
34         user3.blockUser(user1);
35         user1.sendMessage("Hi " + user3.getUsername(), Arrays.asList(user3));
36         System.out.println();
37
38         user3.unblockUser(user1);
39         user1.sendMessage("Hi " + user3.getUsername(), Arrays.asList(user3));
40         System.out.println();
41         printChatHistory(user3);
42     }
43
44     private static void printChatHistory(User user) {
45         System.out.printf("-----%s's Chat History-----\n", user.getUsername());
46         Iterator<Message> chatIterator= user.getChatIterator();
47         while (chatIterator.hasNext()) {
48             Message message = chatIterator.next();
49             String time = message.getTimestamp().toString();
50             String senderName = message.getSender().equals(user) ? user.getUsername():
message.getSender().getUsername();
51             String recipient = senderName.equals(user.getUsername()) ? message.
```

Chat App (Memento, Mediator and Iterator) Implementation

```
51 getRecipients().toString() : user.getUsername();
52     String data = String.format("[%s]%s -> %s: %s",
53         time,
54         senderName,
55         recipient,
56         message.getContent());
57     System.out.println(data);
58 }
59 System.out.println();
60 }
61 }
62
```

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;
2
3 import java.util.Date;
4 import java.util.List;
5 import java.util.Objects;
6
7 public class Message {
8     private static int idIncrement = 0;
9     private int id;
10    private String content;
11    private List<User> recipients;
12    private User sender;
13    private Date timestamp;
14
15    public Message(String content, List<User> recipients, User sender) {
16        this.id = ++idIncrement;
17        this.content = content;
18        this.recipients = recipients;
19        this.sender = sender;
20        this.timestamp = new Date();
21    }
22
23    private Message(int id, String content, List<User> recipients, User sender, Date
timestamp) {
24        this.id = id;
25        this.content = content;
26        this.recipients = recipients;
27        this.sender = sender;
28        this.timestamp = timestamp;
29    }
30
31    public int getId() {
32        return id;
33    }
34
35    public String getContent() {
36        return content;
37    }
38
39    public List<User> getRecipients() {
40        return recipients;
41    }
42
43    public User getSender() {
44        return sender;
45    }
46
47    public Date getTimestamp() {
48        return timestamp;
49    }
50
51    public Memento takeSnapshot() {
52        return new Memento(this.getId(), this.getContent(), this.getRecipients(), this.
```

Chat App (Memento, Mediator and Iterator) Implementation

```
52 getSender(), this.getTimestamp());
53     }
54
55     public static Message restoreSnapshot(Memento memento) {
56         return new Message(memento.getId(), memento.getContent(), memento.getRecipients
57         ()), memento.getSender(), memento.getTimestamp());
58     }
59
60     @Override
61     public boolean equals(Object o) {
62         if (this == o) return true;
63         if (o == null || getClass() != o.getClass()) return false;
64         Message message = (Message) o;
65         return id == message.id;
66     }
67
68     @Override
69     public int hashCode() {
70         return Objects.hashCode(id);
71     }
72
73     @Override
74     public String toString() {
75         return "Message{" +
76             "id=" + id +
77             ", content='" + content + '\'' +
78             ", recipients=" + recipients +
79             ", sender=" + sender +
80             ", timestamp=" + timestamp +
81             '}';
82     }
83
84     public static class Memento {
85         private int id;
86         private String content;
87         private List<User> recipients;
88         private User sender;
89         private Date timestamp;
90
91         public Memento(int id,
92             String content,
93             List<User> recipients,
94             User sender,
95             Date timestamp) {
96             this.id = id;
97             this.content = content;
98             this.recipients = recipients;
99             this.sender = sender;
100             this.timestamp = timestamp;
101         }
102
103         public int getId() {
104             return id;
105         }
106     }
107 }
```

Chat App (Memento, Mediator and Iterator) Implementation

```
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
```

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;
2
3
4 import java.util.*;
5
6 public class ChatServer {
7     private Set<User> users = new HashSet<>();
8     private Map<User, Set<User>> blockedUsers = new HashMap<>();
9     private ChatHistory chatHistory = new ChatHistory();
10
11     public boolean registerUser(User user) {
12         System.out.println(user.getUsername() + " has been registered!");
13         return users.add(user);
14     }
15
16     public boolean unregisterUser(User user) {
17         System.out.println(user.getUsername() + " has been unregistered!");
18         return users.remove(user);
19     }
20
21     public boolean blockUser(User blocker, User blockee) {
22         if (!blockedUsers.containsKey(blocker)) {
23             blockedUsers.put(blocker, new HashSet<>());
24         }
25         System.out.println(blocker.getUsername() + " has blocked " + blockee.getUsername() + " !");
26         return blockedUsers.get(blocker).add(blockee);
27     }
28
29     public boolean unblockUser(User blocker, User blockee) {
30         System.out.println(blocker.getUsername() + " has unblocked " + blockee.getUsername() + " !");
31         return blockedUsers.get(blocker) != null && blockedUsers.get(blocker).remove(blockee);
32     }
33
34
35     public int sendMessage(Message message) {
36         User sender = message.getSender();
37         if (!users.contains(sender)) {
38             System.out.println("Failed to send message, " + sender.getUsername() + " is not registered!");
39             return 0;
40         }
41
42         int count = 0;
43         for (User recipient : message.getRecipients()) {
44             if (!users.contains(recipient)) {
45                 System.out.println("Failed to send message, " + recipient.getUsername() + " is not registered!");
46             } else if (blockedUsers.get(recipient) != null && blockedUsers.get(recipient).contains(sender)) {
47                 System.out.println("Failed to send message, " + sender.getUsername() + " is
```


Chat App (Memento, Mediator and Iterator) Implementation

```
47 blocked by " + recipient.getUsername() + "!");
48     } else {
49         System.out.println(sender.getUsername() + " sent message: `" + message.
getContent() + "` to " + recipient.getUsername());
50         recipient.receiveMessage(message);
51
52         count++;
53     }
54 }
55 if (count > 0) {
56     this.chatHistory.addMessage(message);
57 }
58 return count;
59 }
60
61 public boolean unsendMessage(Message message) {
62     if (this.chatHistory.removeMessage(message)) {
63         System.out.println(message.getSender()
64             .getUsername() + " unsent message: `" + message.getContent() + "` to "
+ message.getRecipients());
65         return true;
66     } else {
67         System.out.println(message.getSender()
68             .getUsername() + " unsent message failed " + message.getContent() + "
to " + message.getRecipients());
69         return false;
70     }
71 }
72 }
73
74 public Iterator<Message> getChatIterator(User user) {
75     return this.chatHistory.iterator(user);
76 }
77
78 public Message getLastMessage(User user) {
79     return this.chatHistory.getLastMessage(user);
80 }
81 }
82
```

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;
2
3 import java.util.ArrayList;
4 import java.util.Iterator;
5 import java.util.List;
6
7 public class ChatHistory implements IterableByUser {
8     private List<Message> history = new ArrayList<>();
9
10    public boolean addMessage(Message message) {
11        return history.add(message);
12    }
13
14    public Message getLastMessage(User user) {
15        return history
16            .stream()
17            .filter(message -> message.getSender().equals(user))
18            .reduce((m1, m2) -> m2)
19            .orElse(null);
20    }
21
22    public List<Message> getHistory(User user) {
23        return history.stream()
24            .filter(
25                message -> message.getSender().equals(user)
26                    || message.getRecipients().contains(user)
27            ).toList();
28    }
29
30    public boolean removeMessage(Message message) {
31        return history.remove(message);
32    }
33
34    @Override
35    public Iterator<Message> iterator(User userToSearchWith) {
36        return new ChatIterator(getHistory(userToSearchWith));
37    }
38 }
39
```

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;
2
3 import java.util.Iterator;
4 import java.util.List;
5
6 public class ChatIterator implements Iterator<Message> {
7     private List<Message> messages;
8     private int currentIndex;
9
10    public ChatIterator(List<Message> userMessages) {
11        this.messages = userMessages;
12        this.currentIndex = 0;
13    }
14
15    @Override
16    public boolean hasNext() {
17        return this.messages.size() > this.currentIndex;
18    }
19
20    @Override
21    public Message next() {
22        return this.messages.get(this.currentIndex++);
23    }
24 }
25
```

Chat App (Memento, Mediator and Iterator) Implementation

```
1 package edu.cpp.cs5800.chatapp;  
2  
3 import java.util.Iterator;  
4  
5 public interface IterableByUser {  
6     Iterator<Message> iterator(User userToSearchWith);  
7 }  
8
```