

```

/Users/Nico/Desktop/Code/RMI/Client.java
Page 1 of 5
Mon Jan 25 20:47:31 2016

1 package chatrmi.client;
2
3 import chatrmi.protocol.ClientInterface;
4 import chatrmi.protocol.Message;
5 import chatrmi.protocol.MessageInterface;
6 import chatrmi.protocol.ServerInterface;
7 import java.rmi.RemoteException;
8 import java.rmi.registry.LocateRegistry;
9 import java.rmi.registry.Registry;
10 import java.rmi.server.UnicastRemoteObject;
11 import java.util.Date;
12
13 /**
14  * Classe permettant de créer un objet Remote de type client pour un
15  * server de
16  * chat
17  * Implémente l'interface Remote "ClientInterface"
18  * @author Nico
19  */
20 public class Client implements ClientInterface{
21
22     /**
23      * Adresse du serveur sur lequel le client est connecté
24      */
25     protected String serverAddress;
26
27     /**
28      * Port par lequel le client et le serveur communiquent
29      */
30     protected int serverPort;
31
32     /**
33      * Nom de l'utilisateur
34      */
35     protected String userName;
36
37     /**
38      * Reference sur le serveur (Remote) sur lequel on est connecté
39      */
40     ServerInterface server;
41
42     /**
43      * Reference sur la fenetre du client
44      */
45     ClientChatView clientChatView;
46
47     /**
48      * Constructeur
49      */
50     public Client() {
51         try {
52             clientChatView = new ClientChatView(this);
53         }
54     }
55 }

```

- 1 -

```

/Users/Nico/Desktop/Code/RMI/Client.java
Page 2 of 5
Mon Jan 25 20:47:31 2016

52     serverAddress = null;
53     serverPort = 1099;
54     UnicastRemoteObject.exportObject(this, 0);
55     clientChatView.setVisible(true);
56     } catch (Exception e) {
57         System.err.println("Client exception : " + e.toString());
58         e.printStackTrace();
59     }
60 }
61
62 /**
63  * Getter
64  * @return Le nom de l'utilisateur
65  */
66 @Override
67 public String getUsername() {
68     return userName;
69 }
70
71 /**
72  * Getter
73  * @return l'adresse du serveur
74  */
75 public String getServerAddress() {
76     return serverAddress;
77 }
78
79 /**
80  * Getter
81  * @return le port
82  */
83 public int getServerPort() {
84     return serverPort;
85 }
86
87 /**
88  * Setter
89  * @param name nom de l'utilisateur
90  */
91 public void setUsername(String name) {
92     userName = name;
93 }
94
95 /**
96  * Setter
97  * @param address adresse du serveur
98  */
99 public void setServerAddress(String address) {
100     serverAddress = address;
101 }
102
103 /**

```

- 2 -

```

/Users/Nico/Desktop/Code/RMI/Client.java
Page 3 of 5
Mon Jan 25 20:47:31 2016

104     * Setter
105     * @param port port de communication
106     */
107     public void setServerPort(int port) {
108         serverPort = port;
109     }
110
111     /**
112      * Methode permettant d'afficher un message sur la vue
113      * @param m message à afficher
114      * @throws RemoteException
115      */
116     @Override
117     public void displayMessage(MessageInterface m) throws RemoteException {
118         clientChatView.displayMessage(m);
119     }
120
121     /**
122      * Methode notifiant la connexion d'un autre utilisateur.
123      * Ajoute le nom du client à la vue
124      * @param c Client
125      * @throws RemoteException
126      */
127     @Override
128     public void newUserConnected(ClientInterface c) throws RemoteException {
129         clientChatView.addUser(c);
130     }
131
132     /**
133      * Methode notifiant la déconnexion d'un autre utilisateur.
134      * Enlève le nom du client à la vue
135      * @param c Client
136      * @throws RemoteException
137      */
138     @Override
139     public void userDisconnected(ClientInterface c) throws RemoteException {
140         if (c.getUsername().compareTo(userName)==0){
141             server = null;
142             clientChatView.refresh();
143         }
144         clientChatView.removeUser(c);
145     }
146
147     /**
148      * Methode permettant d'envoyer un message à tous le client
149      * connecté au
150      * serveur
151      * On crée un stub pour le message écrit par le client et on
152      * l'envoie au

```

- 3 -

```

/Users/Nico/Desktop/Code/RMI/Client.java
Page 4 of 5
Mon Jan 25 20:47:31 2016

151     * serveur.
152     * @param m texte du message
153     */
154     public void sendMessageToAll(String m) {
155         try {
156             Message msg = new Message(new Date(), userName, m);
157             MessageInterface stubMsg = (MessageInterface)
158                 UnicastRemoteObject
159                     .exportObject(msg, 0);
160             server.sendMessageToAll(stubMsg);
161         } catch (Exception e) {
162             System.err.println("Client exception: " + e.toString());
163             e.printStackTrace();
164         }
165     }
166
167     /**
168      * Methode permettant de se connecter au serveur.
169      * Le client établit une connexion au RMI registry crée dans la
170      * classe
171      * Server.java. Si la tentative de connexion réussit, le client
172      * est rajouté
173      * à la liste de Client du serveur.
174      * @param sAddress adresse du serveur
175      * @param sPort port de communication
176      * @param name nom du client
177      * @return
178      */
179     public Boolean connect(String sAddress, String sPort, String name) {
180         Boolean result = true;
181         try {
182             serverAddress = sAddress;
183             userName = name;
184             serverPort = Integer.parseInt(sPort);
185             Registry registry = LocateRegistry
186                 .getRegistry(serverAddress, serverPort);
187             server = (ServerInterface) registry.lookup("server");
188             if (server != null && !server.addClient(this)) {
189                 server = null;
190                 result = false;
191             }
192         } catch (Exception e) {
193             System.err.println("Client exception: " + e.toString());
194             try {
195                 clientChatView.
196                     displayMessage(new Message(new Date(),
197                         "Error", "Impossible to connect."));
198             } catch (Exception e2) {
199                 System.err.println("Client exception: " +
200                     e2.toString());
201             }
202         }
203     }

```

- 4 -

```

199     server = null;
200     result = false;
201 }
202 if (result){
203     clientChatView.refresh();
204 }
205 return result;
206 }
207
208 /**
209  * Methode permettant de se deconnecter du serveur.
210  * Le client se deconnecte du serveur. On supprime le lien vers
211  * le serveur
212  * et on supprime le client de la liste Clients dans la classe
213  * Server.java .
214 */
215 public void disconnect() {
216     try {
217         server.deleteClient(this);
218         server = null;
219         clientChatView.refresh();
220     } catch (Exception e) {
221         System.err.println("Client exception: " + e.toString());
222         e.printStackTrace();
223     }
224 }
225
226 /**
227  * Methode permettant de savoir si le client est connecté a un
228  * serveur.
229  * On vérifie que le client est bien connecté au serveur
230  * (l'attribut "
231  * server" qui constitue la référence vers le serveur est non nul).
232  * @return Retourne true si connecté, false sinon
233 */
234 public Boolean isConnected() {
235     return null != server;
236 }

```

```

1 package chatrmi.protocol;
2
3 import java.rmi.Remote;
4 import java.rmi.RemoteException;
5
6 /**
7  * Interface permettant de définir les methodes necessaire pour un
8  * client d'un
9  * chat developper en RMI
10  * @author Nico
11 */
12 public interface ClientInterface extends Remote {
13
14     /**
15      * Methode permettant d'afficher un message.
16      * @param m Message à afficher.
17      * @throws RemoteException
18      */
19     void displayMessage(MessageInterface m) throws RemoteException;
20
21     /**
22      * Getter
23      * @return Nom de l'utilisateur.
24      * @throws RemoteException
25      */
26     String getUserName() throws RemoteException;
27
28     /**
29      * Methode notifiant la connexion d'un autre utilisateur.
30      * @param c Client.
31      * @throws RemoteException
32      */
33     void newUserConnected(ClientInterface c) throws RemoteException;
34
35     /**
36      * Methode notifiant la déconnexion d'un autre utilisateur.
37      * @param c Client.
38      * @throws RemoteException
39      */
40     void userDisconnected(ClientInterface c) throws RemoteException;
41 }

```

```

1 package chatrmi.protocol;
2
3 import java.text.DateFormat;
4 import java.text.SimpleDateFormat;
5 import java.util.*;
6
7 /**
8  *
9  * @author Nico
10 */
11 public class Message implements MessageInterface{
12
13     /**
14      * Date de l'envoi du message.
15      */
16     private final Date date;
17
18     /**
19      * Nom de l'expediteur.
20      */
21     private final String userName;
22
23     /**
24      * Texte du message.
25      */
26     private final String textMessage;
27
28     /**
29      * Constructeur.
30      * @param d Date de l'envoi du message.
31      * @param usr Nom de l'expediteur.
32      * @param msg Texte du message.
33      */
34     public Message(Date d, String usr, String msg) {
35         date = d;
36         userName = usr;
37         textMessage = msg;
38     }
39
40     /**
41      * Getter.
42      * @return Date de l'envoi du message.
43      */
44     @Override
45     public Date getDate() {
46         return date;
47     }
48
49     /**
50      * Getter.
51      * @return Date de l'envoi du message en String.
52      */

```

```

53     @Override
54     public String getDateToString() {
55         DateFormat formatter = new SimpleDateFormat("MM/dd/yy h:m:s");
56         return formatter.format(date);
57     }
58
59     /**
60      * Getter.
61      * @return Nom de l'expediteur.
62      */
63     @Override
64     public String getUserName() {
65         return userName;
66     }
67
68     /**
69      * Getter
70      * @return Texte du message.
71      */
72     @Override
73     public String getTextMessage() {
74         return textMessage;
75     }
76 }
77

```

```

/Users/Nico/Desktop/Code/RMI/MessageInterface.java
Page 1 of 1
Mon Jan 25 20:48:11 2016

1  /*
2  * To change this license header, choose License Headers in Project
3  * Properties.
4  * To change this template file, choose Tools | Templates
5  * and open the template in the editor.
6  */
7  package chatrmi.protocol;
8
9  import java.rmi.Remote;
10 import java.rmi.RemoteException;
11 import java.util.Date;
12
13 /**
14  * Interface permettant de définir les methodes necessaire pour un
15  * message d'un
16  * chat developper en RMI
17  *
18  * @author Nico
19  */
20 public interface MessageInterface extends Remote{
21
22     /**
23      * Getter
24      * @return Date de l'envoi du message
25      * @throws RemoteException
26      */
27     Date getDate() throws RemoteException;
28
29     /**
30      * Getter
31      * @return Date en String.
32      * @throws RemoteException
33      */
34     String getDateToString() throws RemoteException;
35
36     /**
37      * Getter
38      * @return Nom de l'expediteur.
39      * @throws RemoteException
40      */
41     String getUsername() throws RemoteException;
42
43     /**
44      * Getter
45      * @return Text du message .
46      * @throws RemoteException
47      */
48     String getTextMessage() throws RemoteException;
49 }

```

- 1 -

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 1 of 11
Mon Jan 25 20:48:19 2016

1  /*
2  * To change this license header, choose License Headers in Project
3  * Properties.
4  * To change this template file, choose Tools | Templates
5  * and open the template in the editor.
6  */
7  package chatrmi.server;
8
9  import chatrmi.protocol.ClientInterface;
10 import chatrmi.protocol.Message;
11 import chatrmi.protocol.MessageInterface;
12 import chatrmi.protocol.ServerInterface;
13 import java.io.File;
14 import static java.lang.System.exit;
15 import java.net.InetAddress;
16 import java.rmi.RemoteException;
17 import java.rmi.registry.LocateRegistry;
18 import java.rmi.registry.Registry;
19 import java.rmi.server.UnicastRemoteObject;
20 import java.text.DateFormat;
21 import java.text.SimpleDateFormat;
22 import java.util.Date;
23 import java.util.LinkedList;
24 import javax.xml.parsers.DocumentBuilder;
25 import javax.xml.parsers.DocumentBuilderFactory;
26 import javax.xml.transform.OutputKeys;
27 import javax.xml.transform.Transformer;
28 import javax.xml.transform.TransformerFactory;
29 import javax.xml.transform.dom.DOMSource;
30 import javax.xml.transform.stream.StreamResult;
31 import org.w3c.dom.Document;
32 import org.w3c.dom.Element;
33 import org.w3c.dom.NodeList;
34
35 /**
36  * Classe permettant de créer un objet Remote de type server pour un
37  * chat
38  *
39  * Implémente l'interface Remote "ServerInterface"
40  *
41  * @author Nico
42  */
43 public class Server implements ServerInterface {
44
45     /**
46      * Nom du serveur.
47      */
48     private String serverName;
49
50     /**
51      * Port de communication.
52      */

```

- 1 -

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 2 of 11
Mon Jan 25 20:48:19 2016

51 private int serverPort;
52
53 /**
54  * Chemin d'accès du fichier contenant l'historique des messages.
55  */
56 private String historicsFile;
57
58 /**
59  * Registre remote.
60  */
61 private Registry registry;
62
63 /**
64  * Etat du serveur. (Allumé : true, éteind : false).
65  */
66 private boolean state;
67
68 /**
69  * Liste des clients connectés.
70  */
71 private LinkedList<ClientInterface> clients;
72
73 /**
74  * Historiques de messages.
75  */
76 private LinkedList<MessageInterface> messages;
77
78 /**
79  * Vue serveur.
80  */
81 private ServerRunningView serverRunningView;
82
83 /**
84  * Constructeur. Initialise le nom du serveur à "server", le
85  * port de
86  * communication à 1099 et le fichier d'historique à
87  * "historics.xml".
88  */
89 public Server() {
90     initialize("server",1099,"historics.xml");
91 }
92
93 /**
94  * Permet d'initiliser les attributs.
95  * @param sName Nom du serveur.
96  * @param sPort Port de communication.
97  * @param hFile Chemin d'accès du fichier d'historique.
98  */
99 private void initialize(String sName, int sPort, String hFile) {
100     serverName = sName;
101     serverPort = sPort;
102     historicsFile = hFile;

```

- 2 -

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 3 of 11
Mon Jan 25 20:48:19 2016

101 state = false;
102 clients = new LinkedList<>();
103 messages = new LinkedList<>();
104 serverRunningView = new ServerRunningView(this);
105 serverRunningView.setLocationRelativeTo(null);
106 serverRunningView.setVisible(true);
107 loadHistorics();
108 try
109 {
110     UnicastRemoteObject.exportObject(this, 0);
111 } catch (Exception e) {
112     System.err.println("Server exception: " + e.toString());
113     exit(0);
114 }
115
116 /**
117  * Bind le serveur au registre Remote sur le port "serverPort"
118  * et avec le
119  * nom "serverName".
120  * @return True si réussi, false sinon.
121  */
122 private boolean run() {
123     boolean result = true;
124     try {
125         LocateRegistry.createRegistry(serverPort);
126     } catch (Exception e) {
127         //System.err.println("Server exception: " + e.toString());
128     } finally {
129         try {
130             registry = LocateRegistry.getRegistry();
131             registry.bind(serverName, this);
132         } catch (Exception e) {
133             System.err.println("Server exception: " +
134             e.toString());
135             result = false;
136         }
137     }
138     return result;
139 }
140
141 /**
142  * Permet de lancer le serveur avec comme port de communication
143  * "port".
144  * Modifie l'état du serveur si réussi ou non (valeur retournée).
145  * @param port Port de communication.
146  * @return True si réussi, false sinon.
147  */
148 public boolean start(String port)
149 {
150     try
151     {

```

- 3 -

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 4 of 11
Mon Jan 25 20:48:19 2016

150     serverPort = Integer.parseInt(port);
151     state = run();
152
153     serverRunningView.displayServerLocalAdress(InetAddress.getLocalHost().getHostAddress());
154 }
155 catch(Exception e)
156 {
157     System.err.println("Server exception: " + e.toString());
158 }
159 return state;
160 }
161
162 /**
163  * Permet de stopper le serveur.
164  * Modifie l'état du serveur.
165  * @return True si réussi, false sinon.
166  */
167 public boolean stop()
168 {
169     try {
170         registry.unbind(serverName);
171         state = false;
172
173         for (ClientInterface c1 : clients) {
174             c1.userDisconnected(c1);
175         }
176
177         clients.removeAll(clients);
178         serverRunningView.setClientNumber(clients.size());
179     } catch (Exception e) {
180         System.err.println("Server exception: " + e.toString());
181     }
182     return !state;
183 }
184
185 /**
186  * Méthode permettant de savoir si le serveur est en train de tourner ou non
187  * @return True si le serveur est en train de tourner, false sinon.
188 */
189 public Boolean isRunning() {
190     return state;
191 }
192
193 /**
194  * Méthode permettant d'envoyer un message à tous les clients connectés au serveur.
195
- 4 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 5 of 11
Mon Jan 25 20:48:19 2016

198     * Affiche le message envoyé au serveur par un client à tous les clients de
199     * la liste Clients et rajoute le message à la liste de Messages.
200     * @param m Message à envoyer.
201     * @throws RemoteException
202     */
203     @Override
204     public void sendMessageToAll(MessageInterface m) throws RemoteException {
205         // TODO Auto-generated method stub
206         Message msg = new Message(m.getDate(), m.getUserName(), m.getTextMessage());
207         MessageInterface stubMsg = (MessageInterface) UnicastRemoteObject.exportObject(msg, 0);
208
209         for (ClientInterface client : clients) {
210             client.displayMessage(stubMsg);
211         }
212         messages.addLast(stubMsg);
213     }
214
215     /**
216     * Methode permettant d'ajouter un nouveau client à liste de clients connectés.
217     * Elle permet de rajouter le client qui souhaite se connecter au serveur
218     * afin qu'il bénéficie des services offert par celui-ci. Elle vérifie bien
219     * que le username choisi par le client n'est pas déjà utilisé. Elle le
220     * rajoute à la liste des Clients et lui affiche tous les messages existants
221     * dans la liste Messages.
222     * @param c Client
223     * @return True si l'ajout a pu se faire, false sinon.
224     * @throws RemoteException
225     */
226     @Override
227     public boolean addClient(ClientInterface c) throws RemoteException {
228         // TODO Auto-generated method stub
229         Boolean result = true;
230         if (c == null) {
231             result = false;
232         } else if (contains(c)) {
233             result = false;
234         } else {
235             sendServerMessage(c, "Username \"" + c.getUserName() + "\" already used.");
236         } else {
237             sendServerMessageToAll(c.getUserName() + " has joined
- 5 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 6 of 11
Mon Jan 25 20:48:19 2016

241     the chat.");
242     for (ClientInterface client : clients) {
243         c.newUserConnected(client);
244     }
245
246     clients.add(c);
247     sendHistorics(c);
248
249     for (ClientInterface client : clients) {
250         client.newUserConnected(c);
251     }
252
253     serverRunningView.setClientNumber(clients.size());
254     result = true;
255 }
256 return result;
257 }
258
259 /**
260  * Methode permettant d'enlever un client à liste de clients connectés.
261  * Le serveur informe tous les clients que l'utilisateur c a quitté le chat.
262  * Tous les clients affichent sur le interface graphique : « c has left ».
263  * le client c est supprimé de la liste Clients.
264  * @param c Client.
265  * @throws RemoteException
266  */
267 @Override
268 public void deleteClient(ClientInterface c) throws RemoteException {
269     // TODO Auto-generated method stub
270
271     for (ClientInterface client : clients) {
272         client.userDisconnected(c);
273     }
274     clients.remove(c);
275
276     sendServerMessage(c, "You have left the chat.");
277     sendServerMessageToAll(c.getUserName() + " has left the chat.");
278
279     serverRunningView.setClientNumber(clients.size());
280 }
281
282 /**
283  * Methode permettant de savoir si un client ayant le même non d'utilisateur
284  * que "client" est connecté.
285  * @param client Client.
- 6 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 7 of 11
Mon Jan 25 20:48:19 2016

286     * @return True si oui, false sinon.
287     * @throws RemoteException
288     */
289     private boolean contains(ClientInterface client) throws RemoteException {
290         Boolean result = false;
291         for (ClientInterface c : clients) {
292             if (client.getUserName().compareTo(c.getUserName()) == 0) {
293                 result = true;
294                 break;
295             }
296         }
297         return result;
298     }
299
300     /**
301     * Methode permettant d'envoyer un message à tous les clients connectés au serveur.
302     * Elle permet d'afficher à tous les clients connectés des messages informatifs qui viennent du serveur et non pas d'autres clients
303     * (exemple : informer de la connexion d'un nouveau client est un message du serveur).
304     * @param m Message à envoyer.
305     * @throws RemoteException
306     */
307     private void sendServerMessageToAll(String m) throws RemoteException {
308         Message msg = new Message(new Date(), "Server", m);
309         MessageInterface stubMsg = (MessageInterface) UnicastRemoteObject.exportObject(msg, 0);
310
311         for (ClientInterface client : clients) {
312             client.displayMessage(stubMsg);
313         }
314         messages.addLast(stubMsg); //Tout depend si on enregistre les messages //du server
315     }
316
317     /**
318     * Methode permettant d'envoyer un message à un client connecté au serveur.
319     * @param c Client à qui envoyer le message.
320     * @param m Message à envoyer.
321     * @throws RemoteException
322     */
323
- 7 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 8 of 11
Mon Jan 25 20:48:19 2016

329 private void sendServerMessage(ClientInterface c,String m)
330     throws RemoteException {
331     Message msg = new Message(new Date(),"Server",m);
332     MessageInterface stubMsg = (MessageInterface)
333         UnicastRemoteObject
334             .exportObject(msg, 0);
335     c.displayMessage(stubMsg);
336 }
337
338 /**
339  * Méthode permettant d'envoyer l'historique des messages à un
340  * client
341  * spécifique.
342  * @param c Client à qui envoyer le message.
343  * @throws RemoteException
344  */
345 private void sendHistorics(ClientInterface c) throws
346     RemoteException {
347     for ( MessageInterface m : messages)
348     {
349         c.displayMessage(m);
350     }
351 }
352
353 /**
354  * Methode permettant de charger l'historique des messages
355  * (provenant du
356  * fichier "historicsFile").
357  * Tous les messages enregistrés dans le document XML sont
358  * ajoutés dans
359  * l'ordre chronologique à la liste "messages". Ils sont stubés
360  * pour
361  * pouvoir être envoyés aux différents clients qui se connecterons.
362  */
363 public void loadHistorics() {
364     try {
365         DateFormat formatter = new SimpleDateFormat("MM/dd/yy
366             h:m:s");
367         final DocumentBuilderFactory factory =
368             DocumentBuilderFactory
369                 .newInstance();
370         final DocumentBuilder builder =
371             factory.newDocumentBuilder();
372
373         final Document document= builder.parse(new
374             File(historicsFile));
375
376         final Element racine = document.getDocumentElement();
377         final NodeList racineNoeuds =
378             racine.getElementsByTagName("Message");
379         final int nbRacineNoeuds = racineNoeuds.getLength();
380     }
381 }
382
383 - 8 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 9 of 11
Mon Jan 25 20:48:19 2016

370 System.out.println(nbRacineNoeuds);
371
372 for (int i = 0; i<nbRacineNoeuds; i++) {
373     System.out.println(i);
374     final Element message = (Element) racineNoeuds.item(i);
375
376     final Element date = (Element)
377         message.getElementsByTagName("Date").item(0);
378     final Element username = (Element)
379         message.getElementsByTagName("Username").item(
380             0);
381     final Element text = (Element)
382         message.getElementsByTagName("Text").item(0);
383
384     Message msg = new Message(formatter.parse(
385         date.getTextContent()),username.getTextContent(
386             0),
387         text.getTextContent());
388     MessageInterface stubMsg = (MessageInterface)
389         UnicastRemoteObject.exportObject(msg, 0);
390
391     messages.addLast(stubMsg);
392 }
393
394 } catch(Exception e) {
395 }
396 }
397
398 /**
399  * Methode permettant d'enregistrer l'historique des messages
400  * (dans le fichier "historicsFile").
401  * Tous les messages enregistrés dans la liste "messages" sont
402  * stockés à la suite dans un document XML avec précision de la
403  * date
404  * ainsi que le nom d'utilisateur pour chaque message dans
405  * l'ordre d'ajout
406  * à la liste.
407  */
408 public void saveHistorics() {
409     try {
410         DateFormat formatter = new SimpleDateFormat("MM/dd/yy
411             h:m:s");
412         final DocumentBuilderFactory factory =
413             DocumentBuilderFactory
414                 .newInstance();
415         final DocumentBuilder builder = factory.newDocumentBuilder();
416         final Document document= builder.newDocument();
417         final Element racine = document.createElement("Messages");
418     }
419 }
420
421 - 9 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 10 of 11
Mon Jan 25 20:48:19 2016

414 document.appendChild(racine);
415
416 for(MessageInterface m : messages){
417     final Element message =
418         document.createElement("Message");
419     racine.appendChild(message);
420
421     final Element date = document.createElement("Date");
422
423     date.appendChild(document.createTextNode(formatter.for
424         mat(
425             m.getDate())));
426
427     final Element username =
428         document.createElement("Username");
429     username.appendChild(document.createTextNode(m.getUser
430         Name()));
431
432     final Element text = document.createElement("Text");
433     text.appendChild(document.createTextNode(m.getTextMess
434         age()));
435
436     message.appendChild(date);
437     message.appendChild(username);
438     message.appendChild(text);
439 }
440
441 final TransformerFactory transformerFactory =
442     TransformerFactory
443         .newInstance();
444     final Transformer transformer =
445         transformerFactory.newTransformer();
446     final DOMSource source = new DOMSource(document);
447     final StreamResult sortie = new StreamResult(new
448         File(historicsFile));
449
450     transformer.setOutputProperty(OutputKeys.VERSION, "1.0");
451     transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
452     transformer.setOutputProperty(OutputKeys.STANDALONE,
453         "yes");
454
455     transformer.setOutputProperty(OutputKeys.INDENT, "yes");
456     transformer.setOutputProperty(
457         "{http://xml.apache.org/xslt}indent-amount", "2");
458
459     transformer.transform(source, sortie);
460 } catch(Exception e) {
461 }
462
463 }
464
465 - 10 -

```

```

/Users/Nico/Desktop/Code/RMI/Server.java
Page 11 of 11
Mon Jan 25 20:48:19 2016

454 }
455
456 }
457
458 }
459
460 }
461
462 }
463
464 }
465
466 }
467
468 }
469
470 }
471
472 }
473
474 }
475
476 }
477
478 }
479
480 }
481
482 }
483
484 }
485
486 }
487
488 }
489
490 }
491
492 }
493
494 }
495
496 }
497
498 }
499
500 }
501
502 }
503
504 }
505
506 }
507
508 }
509
510 }
511
512 }
513
514 }
515
516 }
517
518 }
519
520 }
521
522 }
523
524 }
525
526 }
527
528 }
529
530 }
531
532 }
533
534 }
535
536 }
537
538 }
539
540 }
541
542 }
543
544 }
545
546 }
547
548 }
549
550 }
551
552 }
553
554 }
555
556 }
557
558 }
559
560 }
561
562 }
563
564 }
565
566 }
567
568 }
569
570 }
571
572 }
573
574 }
575
576 }
577
578 }
579
580 }
581
582 }
583
584 }
585
586 }
587
588 }
589
590 }
591
592 }
593
594 }
595
596 }
597
598 }
599
600 }
601
602 }
603
604 }
605
606 }
607
608 }
609
610 }
611
612 }
613
614 }
615
616 }
617
618 }
619
620 }
621
622 }
623
624 }
625
626 }
627
628 }
629
630 }
631
632 }
633
634 }
635
636 }
637
638 }
639
640 }
641
642 }
643
644 }
645
646 }
647
648 }
649
650 }
651
652 }
653
654 }
655
656 }
657
658 }
659
660 }
661
662 }
663
664 }
665
666 }
667
668 }
669
670 }
671
672 }
673
674 }
675
676 }
677
678 }
679
680 }
681
682 }
683
684 }
685
686 }
687
688 }
689
690 }
691
692 }
693
694 }
695
696 }
697
698 }
699
700 }
701
702 }
703
704 }
705
706 }
707
708 }
709
710 }
711
712 }
713
714 }
715
716 }
717
718 }
719
720 }
721
722 }
723
724 }
725
726 }
727
728 }
729
730 }
731
732 }
733
734 }
735
736 }
737
738 }
739
740 }
741
742 }
743
744 }
745
746 }
747
748 }
749
750 }
751
752 }
753
754 }
755
756 }
757
758 }
759
760 }
761
762 }
763
764 }
765
766 }
767
768 }
769
770 }
771
772 }
773
774 }
775
776 }
777
778 }
779
780 }
781
782 }
783
784 }
785
786 }
787
788 }
789
790 }
791
792 }
793
794 }
795
796 }
797
798 }
799
800 }
801
802 }
803
804 }
805
806 }
807
808 }
809
810 }
811
812 }
813
814 }
815
816 }
817
818 }
819
820 }
821
822 }
823
824 }
825
826 }
827
828 }
829
830 }
831
832 }
833
834 }
835
836 }
837
838 }
839
840 }
841
842 }
843
844 }
845
846 }
847
848 }
849
850 }
851
852 }
853
854 }
855
856 }
857
858 }
859
860 }
861
862 }
863
864 }
865
866 }
867
868 }
869
870 }
871
872 }
873
874 }
875
876 }
877
878 }
879
880 }
881
882 }
883
884 }
885
886 }
887
888 }
889
890 }
891
892 }
893
894 }
895
896 }
897
898 }
899
900 }
901
902 }
903
904 }
905
906 }
907
908 }
909
910 }
911
912 }
913
914 }
915
916 }
917
918 }
919
920 }
921
922 }
923
924 }
925
926 }
927
928 }
929
930 }
931
932 }
933
934 }
935
936 }
937
938 }
939
940 }
941
942 }
943
944 }
945
946 }
947
948 }
949
950 }
951
952 }
953
954 }
955
956 }
957
958 }
959
960 }
961
962 }
963
964 }
965
966 }
967
968 }
969
970 }
971
972 }
973
974 }
975
976 }
977
978 }
979
980 }
981
982 }
983
984 }
985
986 }
987
988 }
989
990 }
991
992 }
993
994 }
995
996 }
997
998 }
999
1000 }
1001
1002 }
1003
1004 }
1005
1006 }
1007
1008 }
1009
1010 }
1011
1012 }
1013
1014 }
1015
1016 }
1017
1018 }
1019
1020 }
1021
1022 }
1023
1024 }
1025
1026 }
1027
1028 }
1029
1030 }
1031
1032 }
1033
1034 }
1035
1036 }
1037
1038 }
1039
1040 }
1041
1042 }
1043
1044 }
1045
1046 }
1047
1048 }
1049
1050 }
1051
1052 }
1053
1054 }
1055
1056 }
1057
1058 }
1059
1060 }
1061
1062 }
1063
1064 }
1065
1066 }
1067
1068 }
1069
1070 }
1071
1072 }
1073
1074 }
1075
1076 }
1077
1078 }
1079
1080 }
1081
1082 }
1083
1084 }
1085
1086 }
1087
1088 }
1089
1090 }
1091
1092 }
1093
1094 }
1095
1096 }
1097
1098 }
1099
1100 }
1101
1102 }
1103
1104 }
1105
1106 }
1107
1108 }
1109
1110 }
1111
1112 }
1113
1114 }
1115
1116 }
1117
1118 }
1119
1120 }
1121
1122 }
1123
1124 }
1125
1126 }
1127
1128 }
1129
1130 }
1131
1132 }
1133
1134 }
1135
1136 }
1137
1138 }
1139
1140 }
1141
1142 }
1143
1144 }
1145
1146 }
1147
1148 }
1149
1150 }
1151
1152 }
1153
1154 }
1155
1156 }
1157
1158 }
1159
1160 }
1161
1162 }
1163
1164 }
1165
1166 }
1167
1168 }
1169
1170 }
1171
1172 }
1173
1174 }
1175
1176 }
1177
1178 }
1179
1180 }
1181
1182 }
1183
1184 }
1185
1186 }
1187
1188 }
1189
1190 }
1191
1192 }
1193
1194 }
1195
1196 }
1197
1198 }
1199
1200 }
1201
1202 }
1203
1204 }
1205
1206 }
1207
1208 }
1209
1210 }
1211
1212 }
1213
1214 }
1215
1216 }
1217
1218 }
1219
1220 }
1221
1222 }
1223
1224 }
1225
1226 }
1227
1228 }
1229
1230 }
1231
1232 }
1233
1234 }
1235
1236 }
1237
1238 }
1239
1240 }
1241
1242 }
1243
1244 }
1245
1246 }
1247
1248 }
1249
1250 }
1251
1252 }
1253
1254 }
1255
1256 }
1257
1258 }
1259
1260 }
1261
1262 }
1263
1264 }
1265
1266 }
1267
1268 }
1269
1270 }
1271
1272 }
1273
1274 }
1275
1276 }
1277
1278 }
1279
1280 }
1281
1282 }
1283
1284 }
1285
1286 }
1287
1288 }
1289
1290 }
1291
1292 }
1293
1294 }
1295
1296 }
1297
1298 }
1299
1300 }
1301
1302 }
1303
1304 }
1305
1306 }
1307
1308 }
1309
1310 }
1311
1312 }
1313
1314 }
1315
1316 }
1317
1318 }
1319
1320 }
1321
1322 }
1323
1324 }
1325
1326 }
1327
1328 }
1329
1330 }
1331
1332 }
1333
1334 }
1335
1336 }
1337
1338 }
1339
1340 }
1341
1342 }
1343
1344 }
1345
1346 }
1347
1348 }
1349
1350 }
1351
1352 }
1353
1354 }
1355
1356 }
1357
1358 }
1359
1360 }
1361
1362 }
1363
1364 }
1365
1366 }
1367
1368 }
1369
1370 }
1371
1372 }
1373
1374 }
1375
1376 }
1377
1378 }
1379
1380 }
1381
1382 }
1383
1384 }
1385
1386 }
1387
1388 }
1389
1390 }
1391
1392 }
1393
1394 }
1395
1396 }
1397
1398 }
1399
1400 }
1401
1402 }
1403
1404 }
1405
1406 }
1407
1408 }
1409
1410 }
1411
1412 }
1413
1414 }
1415
1416 }
1417
1418 }
1419
1420 }
1421
1422 }
1423
1424 }
1425
1426 }
1427
1428 }
1429
1430 }
1431
1432 }
1433
1434 }
1435
1436 }
1437
1438 }
1439
1440 }
1441
1442 }
1443
1444 }
1445
1446 }
1447
1448 }
1449
1450 }
1451
1452 }
1453
1454 }
1455
1456 }
1457
1458 }
1459
1460 }
1461
1462 }
1463
1464 }
1465
1466 }
1467
1468 }
1469
1470 }
1471
1472 }
1473
1474 }
1475
1476 }
1477
1478 }
1479
1480 }
1481
1482 }
1483
1484 }
1485
1486 }
1487
1488 }
1489
1490 }
1491
1492 }
1493
1494 }
1495
1496 }
1497
1498 }
1499
1500 }
1501
1502 }
1503
1504 }
1505
1506 }
1507
1508 }
1509
1510 }
1511
1512 }
1513
1514 }
1515
1516 }
1517
1518 }
1519
1520 }
1521
1522 }
1523
1524 }
1525
1526 }
1527
1528 }
1529
1530 }
1531
1532 }
1533
1534 }
1535
1536 }
1537
1538 }
1539
1540 }
1541
1542 }
1543
1544 }
1545
1546 }
1547
1548 }
1549
1550 }
1551
1552 }
1553
1554 }
1555
1556 }
1557
1558 }
1559
1560 }
1561
1562 }
1563
1564 }
1565
1566 }
1567
1568 }
1569
1570 }
1571
1572 }
1573
1574 }
1575
1576 }
1577
1578 }
1579
1580 }
1581
1582 }
1583
1584 }
1585
1586 }
1587
1588 }
1589
1590 }
1591
1592 }
1593
1594 }
1595
1596 }
1597
1598 }
1599
1600 }
1601
1602 }
1603
1604 }
1605
1606 }
1607
1608 }
1609
1610 }
1611
1612 }
1613
1614 }
1615
1616 }
1617
1618 }
1619
1620 }
1621
1622 }
1623
1624 }
1625
1626 }
1627
1628 }
1629
1630 }
1631
1632 }
1633
1634 }
1635
1636 }
1637
1638 }
1639
1640 }
1641
1642 }
1643
1644 }
1645
1646 }
1647
1648 }
1649
1650 }
1651
1652 }
1653
1654 }
1655
1656 }
1657
1658 }
1659
1660 }
1661
1662 }
1663
1664 }
1665
1666 }
1667
1668 }
1669
1670 }
1671
1672 }
1673
1674 }
1675
1676 }
1677
1678 }
1679
1680 }
1681
1682 }
1683
1684 }
1685
1686 }
1687
1688 }
1689
1690 }
1691
1692 }
1693
1694 }
1695
1696 }
1697
1698 }
1699
1700 }
1701
1702 }
1703
1704 }
1705
1706 }
1707
1708 }
1709
1710 }
1711
1712 }
1713
1714 }
1715
1716 }
1717
1718 }
1719
1720 }
1721
1722 }
1723
1724 }
1725
1726 }
1727
1728 }
1729
1730 }
1731
1732 }
1733
1734 }
1735
1736 }
1737
1738 }
1739
1740 }
1741
1742 }
1743
1744 }
1745
1746 }
1747
1748 }
1749
1750 }
1751
1752 }
1753
1754 }
1755
1756 }
1757
1758 }
1759
1760 }
1761
1762 }
1763
1764 }
1765
1766 }
1767
1768 }
1769
1770 }
1771
1772 }
1773
1774 }
1775
1776 }
1777
1778 }
1779
1780 }
1781
1782 }
1783
1784 }
1785
1786 }
1787
1788 }
1789
1790 }
1791
1792 }
1793
1794 }
1795
1796 }
1797
1798 }
1799
1800 }
1801
1802 }
1803
1804 }
1805
1806 }
1807
1808 }
1809
1810 }
1811
1812 }
1813
1814 }
1815
1816 }
1817
1818 }
1819
1820 }
1821
1822 }
1823
1824 }
1825
1826 }
1827
1828 }
1829
1830 }
1831
1832 }
1833
1834 }
1835
1836 }
1837
1838 }
1839
1840 }
1841
1842 }
1843
1844 }
1845
1846 }
1847
1848 }
1849
1850 }
1851
1852 }
1853
1854 }
1855
1856 }
1857
1858 }
1859
1860 }
1861
1862 }
1863
1864 }
1865
1866 }
1867
1868 }
1869
1870 }
1871
1872 }
1873
1874 }
1875
1876 }
1877
1878 }
1879
1880 }
1881
1882 }
1883
1884 }
1885
1886 }
1887
1888 }
1889
1890 }
1891
1892 }
1893
1894 }
1895
1896 }
1897
1898 }
1899
1900 }
1901
1902 }
1903
1904 }
1905
1906 }
1907
1908 }
1909
1910 }
1911
1912 }
1913
1914 }
1915
1916 }
1917
1918 }
1919
1920 }
1921
1922 }
1923
1924 }
1925
1926 }
1927
1928 }
1929
1930 }
1931
1932 }
1933
1934 }
1935
1936 }
1937
1938 }
1939
1940 }
1941
1942 }
1943
1944 }
1945
1946 }
1947
1948 }
1949
1950 }
1951
1952 }
1953
1954 }
1955
1956 }
1957
1958 }
1959
1960 }
1961
1962 }
1963
1964 }
1965
1966 }
1967
1968 }
1969
1970 }
1971
1972 }
1973
1974 }
1975
1976 }
1977
1978 }
1979
1980 }
1981
1982 }
1983
1984 }
1985
1986 }
1987
1988 }
1989
1990 }
1991
1992 }
1993
1994 }
1995
1996 }
1997
1998 }
1999
2000 }
2001
2002 }
2003
2004 }
2005
2006 }
2007
2008 }
2009
2010 }
2011
2012 }
2013
2014 }
2015
2016 }
2017
2018 }
2019
2020 }
2021
2022 }
2023
2024 }
2025
2026 }
2027
2028 }
2029
2030 }
2031
2032 }
2033
2034 }
2035
2036 }
2037
2038 }
2039
2040 }
2041
2042 }
2043
2044 }
2045
2046 }
2047
2048 }
2049
2050 }
2051
2052 }
2053
2054 }
2055
2056 }
2057
2058 }
2059
2060 }
2061
2062 }
2063
2064 }
2065
2066 }
2067
2068 }
2069
2070 }
2071
2072 }
2073
2074 }
2075
2076 }
2077
2078 }
2079
2080 }
2081
2082 }
2083
2084 }
2085
2086 }
2087
2088 }
2089
2090 }
2091
2092 }
2093
2094 }
2095
2096 }
2097
2098 }
2099
2100 }
2101
2102 }
2103
2104 }
2105
2106 }
2107
2108 }
2109
2110 }
2111
2112 }
2113
2114 }
2115
2116 }
2117
2118 }
2119
2120 }
2121
2122 }
2123
2124 }
2125
2126 }
2127
2128 }
2129
2130 }
2131
2132 }
2133
2134 }
2135
2136 }
2137
2138 }
2139
2140 }
2141
2142 }
2143
2144 }
2145
2146 }
2147
2148 }
2149
2150 }
2151
2152 }
2153
2154 }
2155
2156 }
2157
2158 }
2159
2160 }
2161
2162 }
2163
2164 }
2165
2166 }
2167
2168 }
2169
2170 }
2171
2172 }
2173
2174 }
2175
2176 }
2177
2178 }
2179
2180 }
2181
2182 }
2183
2184 }
2185
2186 }
2187
2188 }
2189
2190 }
2191
2192 }
2193
2194 }
2195
2196 }
2197
2198 }
2199
2200 }
2201
2202 }
2203
2204 }
2205
2206 }
2207
2208 }
2209
2210 }
2211
2212 }
2213
2214 }
2215
2216 }
2217
2218 }
2219
2220 }
2221
2222 }
2223
2224 }
2225
2226 }
2227
2228 }
2229
2230 }
2231
2232 }
2233
2234 }
2235
2236 }
2237
2238 }
2239
2240 }
2241
2242 }
2243
2244 }
2245
2246 }
2247
2248 }
2249
2250 }
2251
2252 }
2253
2254 }
2255
2256 }
2257
2258 }
2259
2260 }
2261
2262 }
2263
2264 }
2265
2266 }
2267
2268 }
2269
2270 }
2271
2272 }
2273
2274 }
2275
2276 }
2277
2278 }
2279
2280 }
2281
2282 }
2283
2284 }
2285
2286 }
2287
2288 }
2289
2290 }
2291
2292 }
2293
2294 }
2295
2296 }
2297
2298 }
2299
2300 }
2301
2302 }
2303
2304 }
2305
2306 }
2307
2308 }
2309
2310 }
2311
2312 }
2313
2314 }
2315
2316 }
2317
2318 }
2319
2320 }
2321
2322 }
2323
2324 }
2325
2326 }
2327
2328 }
2329
2330 }
2331
2332 }
2333
2334 }
2335
2336 }
2337
2338 }
2339
2340 }
2341
2342 }
2343
2344 }
2345
2346 }
2347
2348 }
2349
2350 }
2351
2352 }
2353
2354 }
2355
2356 }
2357
2358 }
2359
2360 }
2361
2362 }
2363
2364 }
2365
2366 }
2367
2368 }
2369
2370 }
2371
2372 }
2373
2374 }
2375
2376 }
2377
2378 }
2379
2380 }
2381
2382 }
2383
2384 }
2385
2386 }
2387
2388 }
2389
2390 }
2391
2392 }
2393
2394 }
2395
2396 }
2397
2398 }
2399
2400 }
2401
2402 }
2403
2404 }
2405
2406 }
2407
2408 }
2409
2410 }
2411
2412 }
2413
2414 }
2415
2416 }
2417
2418 }
2419
2420 }
2421
2422 }
2423
2424 }
2425
2426 }
2427
2428 }
2429
2430 }
2431
2432 }
2433
2434 }
2435
2436 }
2437
2438 }
2439
2440 }
2441
2442 }
2443
2444 }
2445
2446 }
2447
2448 }
2449
2450 }
2451
2452 }
2453
2454 }
2455
2456 }
2457
2458 }
2459
2460 }
246
```

```
1 package chatrmi.protocol;
2
3
4 import java.rmi.Remote;
5 import java.rmi.RemoteException;
6
7 /**
8  * Interface permettant de définir les methodes necessaire pour un
9  * server d'un
10  * chat developper en RMI
11  * @author Nico
12  */
13 public interface ServerInterface extends Remote {
14
15     /**
16      * Methode permettant d'envoyer un message à tous les clients
17      * connectés au
18      * serveur.
19      * @param m Message à envoyer.
20      * @throws RemoteException
21      */
22     void sendMessageToAll(MessageInterface m) throws RemoteException;
23
24     /**
25      * Methode permettant d'ajouter un nouveau client à liste de
26      * clients
27      * connectés. (Connecter le client)
28      * @param c Client
29      * @return True si l'ajout a pu se faire, false sinon.
30      * @throws RemoteException
31      */
32     boolean addClient(ClientInterface c) throws RemoteException;
33
34     /**
35      * Methode permettant d'enlever un client à liste de clients
36      * connectés.
37      * (Deconnecter le client)
38      * @param c Client.
39      * @throws RemoteException
40      */
41     void deleteClient(ClientInterface c) throws RemoteException;
42 }
```