```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package chat.server;

import chat.protocol.Message;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.Socket;

/**
 *
 * @author Nico
 */
public class ServerThread extends Thread {
    /**
     * Référence sur le serveur.
     */
    private final Server server;

    /**
     * Réference sur la socket de communication.
     */
    private final Socket clientSocket;

    /**
     * Référence sur le thread parent.
     */
    private final ServerMultiThread serverMultiThread;

    /**
     * Flux d'écriture de message.
     */
    PrintStream socOut;

    /**
     * Flux de lecture de message.
     */
    BufferedReader socIn;

    /**
     * Nom de l'utilisateur relié.
     */
    private String userName;

    /**
     * COnstructeur.
     * @param s Référence sur le serveur.
```

```
52          * @param smt Référence sur le thread parent.
53          * @param c Socket de communication.
54          */
55         public ServerThread(Server s,ServerMultiThread smt, Socket c) {
56             server = s;
57             clientSocket = c;
58             serverMultiThread = smt;
59             try {
60                 socIn = new BufferedReader(new
                   InputStreamReader(clientSocket
61                         .getInputStream()));
62                 socOut = new PrintStream(clientSocket.getOutputStream());
63             } catch (Exception e) {
64                 System.err.println("ServerThread  exception : " +
                   e.toString());
65             }
66         }
67
68         /**
69          * Getter.
70          * @return Nom de l'utilisateur.
71          */
72         public String getUserName() {
73             return userName;
74         }
75
76         /**
77          * Méthode exécutée par le thread. Boucle tant que
           server.isRunning() est
78          * true. Attend que le client envoi une ligne et la traite.
79          */
80         @Override
81         public void run(){
82             try {
83                 String line;
84                 while (server.isRunning()) {
85                     line = socIn.readLine();
86                     treatLine(line);
87                 }
88             } catch (Exception e) {
89                 System.err.println("ServerThread  exception : " +
                   e.toString());
90             }
91         }
92
93         /**
94          * Méthode d'intéruption du thread. Deconnecte l'utilisateur en
           lui envoyant
95          * un message de deconnexion et ferme la sockets.
96          */
97         @Override
98         public void interrupt() {
```

```
 99            super.interrupt();
100            try {
101                sendMessage("SIGNOUT "+userName);
102                socOut.close();
103                socIn.close();
104                clientSocket.close();
105            } catch (Exception e) {
106                System.err.println("Client exception : "+ e.toString());
107            }
108        }
109
110        /**
111         * Methode traitant une ligne recue par le flux d'entrée "socIn".
112         * @param line Ligne lu.
113         */
114        private void treatLine(String line){
115            if(line.contains("CONNECT")) {
116                String[] splits = line.split("CONNECT ");
117                userName = splits[1];
118                serverMultiThread.signin(this);
119            } else if(line.compareTo("QUIT")==0) {
120                serverMultiThread.signout(this);
121            } else if(line.contains("SENDTO") &&                              ↵
            line.contains("CONTENT")) {
122                String[] splits1 = line.split("SENDTO ");
123                String[] splits2 = splits1[1].split(" CONTENT ");
124                String sender = userName;
125                String receiver = splits2[0];
126                String textMessage = "MESSAGE FROM " + sender + " TO " +   ↵
                receiver
127                        + " CONTENT " + splits2[1];
128                serverMultiThread.sendMessage(
129                        new Message(sender,receiver,textMessage));
130            }
131        }
132
133        /**
134         * Méthode permettant d'envoyer un message au client.
135         * @param m
136         */
137        public void sendMessage(String m) {
138            try{
139                socOut.println(m);
140            } catch (Exception e) {
141                System.err.println("ServerThread  exception : " +          ↵
                e.toString());
142            }
143        }
144    }
145
```