Centro Federal de Educação Tecnológica de Minas Gerais
Campus VII - Unidade Timóteo - Engenharia da Computação
Trabalho Prático 2

Trabalho Prático II: RPC/RMI

Arthur Morais Pimentel

Trabalho prático de RPC/RMI, implementação de um Jogo Da Velha usando chamada remota de procedimento/método.

Professor: Lucas Pantuza Amorim

Timóteo, Outubro de 2022

1 Introdução

Neste trabalho foi implementado um programa que opera no modelo ponta-a-ponta que exercita a transmissão usando chamada remota de procedimento/método.
Para fins didáticos, e consequentemente poder exercitar o contéudo teórico foi escolhido a funcionalidade de um jogo (Jogo da Velha), para poder contemplar a usabilidade da comunicação do tipo procedimento/método. Neste contexto foi aprofundando o uso de dois clientes, e um servidor que fará o intermédio entre eles.

2 Metodologia

Para desenvolvimento da aplicação foi utilizado a linguagem de programação Java, que disponibiliza de bibliotecas próprias para o desenvolvimento de comunicação na rede. O Java disponibiliza uma interface de programação que permite a execução de chamadas remotas no estilo RPC, o RMI (Remote Method Invocation).

Foi criado uma interface atrávez da lib JFrame, própria da linguagem Java, onde a mesma proporciona uma melhor iteração do sistema com o usuário, e oferece uma melhor visibilidade das funcionalidades pautadas. De acordo com a figura capturada no inicio da execução do sistema, temos a seguinte imagem ilustrativa 1.

Figura 1: Interface gráfica

A figura 1, apresenta alguns campos a serem informados pelo usuário, bem como o tabuleiro do jogo. O usuário precisa informar o tipo de conexão, no caso sendo servidor ou cliente.Para servidor a tela ficará inoperante, uma vez que o servidor atua apenas como uma ponte para comunicação dos clientes. Para um tipo de aplicação como cliente, o usuário precisa informar um nome como jogador que será conhecido internamente como o nome do serviço disponibilizado pelo cliente. O jogo será iniciado apenas quando o servidor receber a conexão de dois clientes, o primeiro a iniciar a jogar será o primeiro que conectou ao servidor.

A IDE utilizada foi o IntelliJIDEA, o principal motivo foi poder subir as 3 aplicações ao mesmo tempo, de um mesmo projeto. Para isso foi necessário ativar a seguinte configuração que permitiu executar mais de uma aplicação ao mesmo tempo, sendo ela: "Allow multiple intances", ou pela tecla de atalho "Alt + U", que se encontra na tela "Run/Debug Configurations"da IDE. Após ativar a configuração, basta apenas iniciar 3 aplições, uma operando como servidor, e as duas restantes como clientes.
As conexões precisam ser do mesmo tipo para poder realizar as comunicações e poder efetuar as jogadas. Com o fim do jogo, ambas as 3 aplicações se encerram.

3 Resultados

Inicilizando o jogo como servidor, para poder receber a conexão dos clientes, ao clicar em conectar aparecerá uma mensagem informando que o servidor foi inicializado, conforme a figura 2

Figura 2: Inicialização do servidor

Após subir o servidor, na tela do mesmo não acontecerá mais nenhuma iteração, portanto agora é preciso subir os dois clientes. Conectado os dois clientes no servidor, o jogo é inicializado, e com isso um cliente por vez vem a clicar em algum campo disponível do tabulereiro e realizar sua jogada.

Há um controle de usabilidade no que diz respeito quando é a vez de um cliente jogar. O outro jogador(cliente) não consegue iterar na tela enquanto aguarda a jogada do joagador anterior, o servidor fica no aguarde da mensagem do cliente operante para encaminha-lá para o cliente ocioso, e assim procede o decorrer do jogo. Desta forma o fluxo vai procedindo até que o tabuleiro fique completo e dê velha, ou até que algum jogador ganhe. Conforme podemos visualizar nas figura 3 e 4 .

4 Análise

Em meios de análise, pode-se observar como os dados são trafegados, dado como uma invocação remota disponibiliza seus métodos na rede sit [1].

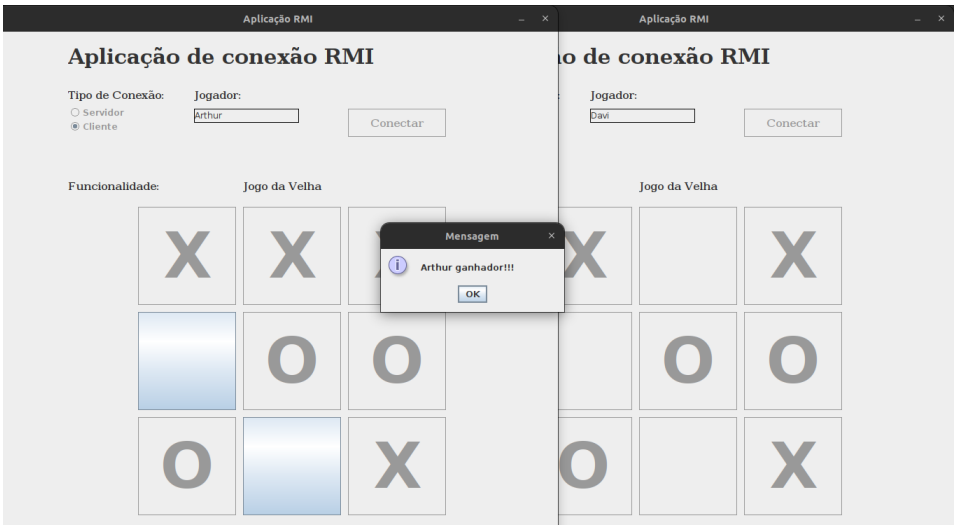

Figura 3: Vitória de um jogador

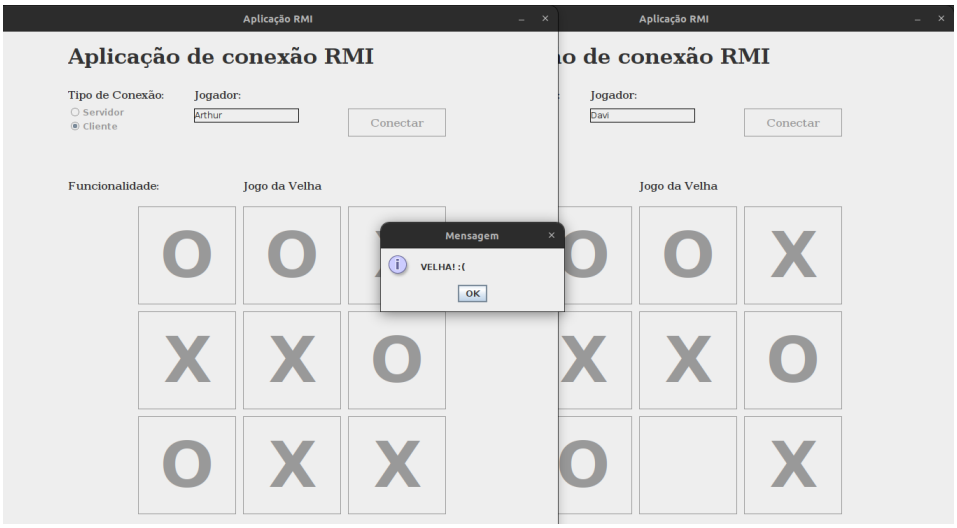

Figura 4: Derrota dos jogadores

Resumidamente temos que o RMI uma aplicação Java de uma Chamada remota de procedimento (RPC) é uma tecnologia popular para a implementação do modelo cliente-servidor de computação distribuída que permite a um programa de computador chamar um procedimento em outro espaço de endereçamento (geralmente em outro computador, conectado por uma rede).

Vale frisar que para o trabalho proposto, em que é executado três aplicações, sendo elas servidor e clientes, temos que tanto como o seervidor disponibiliza seus métodos e procedimentos instanciados

de uma interface remota, o cliente também tem sua interface remota e disponibiliza alguns métodos para que possa receber mensagem do servidor, que no caso desta aplicação se trata de um retorno de outro jogador.

Dito isso vemos pela figura 5 e 6, a captura da conexão dos cliente ao servidor, podemos notar que para meios de conexão e acesso remoto é usado o protocolo RMI. Ja para a transferência de dados podemos visualizar pela figura 7 que o protocolo utilizado 'por debaixo dos panos' é o TCP. Para realizar estas capturas foi utilizado o software Wireshark sit [2], que é um programa que analisa o tráfego de rede, e o organiza por protocolos.

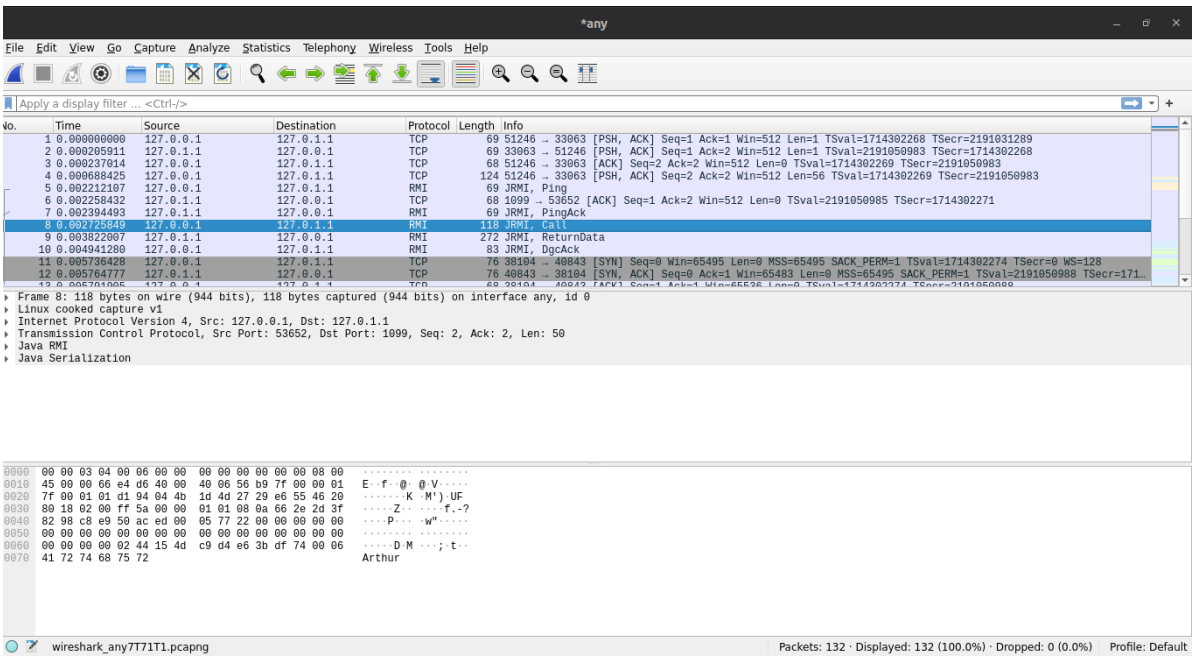

Figura 5: Conexão Cliente 1

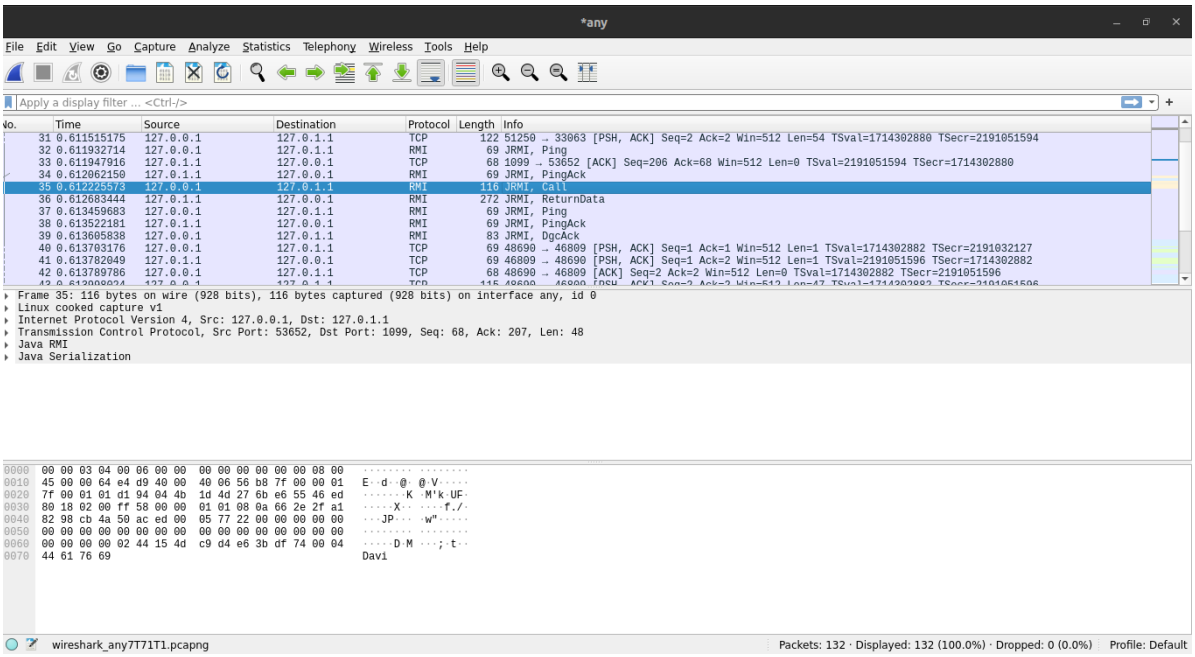

Figura 6: Conexão Cliente 2

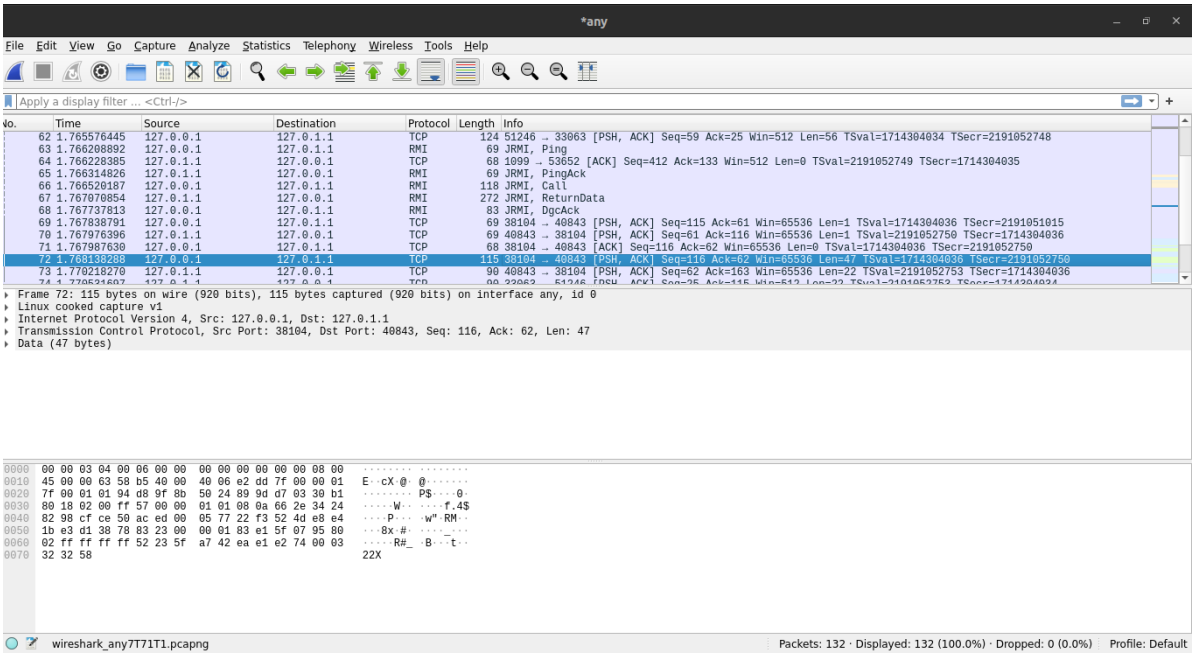

Figura 7: Transferência de dados

5 Conclusão

Pode-se contemplar a facilidade que o protocolo RMI proporcionou para poder se comunicar na rede, uma vez definido uma interface e seus procedimentos podemos utilizar seus métodos em outra instâncias, sendo possível atualizar dinâmicamente o código remoto.

Com a realização do trabalho apresentado, foi possível entender na prática um pouco melhor sobre os protocolos de transporte estudados em sala de aula, bem como ter a experiência de capturar os dados trafegados em rede pelo Wireshark.

6 Código

Todo o código fonte se encontra disponível em GitHub.

Segue o código comentado:

6.1 Main

Onde será inicializado a aplicação

```
1  package Main;
2
3  import Interface.Aplication;
4  import RMI.Client.ClientImp;
5  import RMI.Server.ServerImp;
6  import Utils.Window;
7
8  import java.rmi.RemoteException;
9
10 public class Main implements Aplication {
11
12     /*
13      * Variaveis Estaticas
14      *
15      */
```

```
16      public static Window window;
17
18      public static ClientImp client;
19
20      public static ServerImp server;
21
22      public static void main(String[] args) {
23          window = new Window();
24      }
25
26      /*
27       * Call back:
28       *
29       * Ao efetuar a conexao na JFrame a aplica   o
30       * RMI respectiva     inicializada
31       */
32      @Override
33      public void initialize() {
34          try {
35              if(window.getAplicationConnection().equals("Server")){
36                  server = new ServerImp();
37                  server.startServer();
38              }
39              if(window.getAplicationConnection().equals("Client")){
40                  client = new ClientImp(window);
41                  client.startClient();
42                  client.connectServer();
43              }
44
45          } catch (RemoteException remoteException) {
46              remoteException.printStackTrace();
47          }
48      }
49
50      @Override
51      public void sendClient() {
52          client.sendServer(window.getMove(), window.getNamePlayer());
53      }
54  }
```

codes/Main/Main.java

6.2 RMI

Onde será implementado as interfaces remotas, bem como a sua implementação

6.2.1 Client

Interface Remota do cliente

```java
package RMI.Client;

import java.rmi.Remote;
import java.rmi.RemoteException;

/*
 *
 * Interface Remota do Cliente
 */
public interface Client extends Remote {

    // Metodo responsavel pela troca de jogador, dado o proximo jogador
    void changePlayer(String message) throws RemoteException;

    void initializePlayer(String client) throws RemoteException;

}
```

codes/RMI/Client/Client.java

Implementação da interface remota do cliente

```java
package RMI.Client;

import RMI.Server.Server;
import Utils.Messages;
import Utils.Window;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class ClientImp extends UnicastRemoteObject implements Client {

    Window window;

    Server server;
    public ClientImp(Window window) throws RemoteException{
        super();
        this.window = window;
    }
```

```
21
22      @Override
23      public void changePlayer(String message) throws RemoteException {
24          if(message.equals(Messages.FINISH.getValue())){
25              System.exit(0);
26          }
27          window.updateTable(message);
28      }
29
30      @Override
31      public void initializePlayer(String client) throws RemoteException {
32          window.isClient(client);
33
34          if(client.equals(Messages.CONNECTCLIENT1.getValue()))
35              window.enableTable();
36      }
37
38      // Disponibiliza os servi os do cliente
39      public  void startClient(){
40          try{
41              LocateRegistry.getRegistry(1099);
42              Client client = new ClientImp(window);
43              Naming.rebind(window.getNamePlayer(), client);
44          }catch (Exception e){
45              e.printStackTrace();
46          }
47      }
48
49      // Conex o
50      // Primeiro contato do cliente com o servidor
51      public  void connectServer(){
52          try {
53              server = (Server) Naming.lookup("Servidor");
54              server.connect(window.getNamePlayer());
55          }catch (Exception e){
56              e.printStackTrace();
57          }
58      }
59
60      // Enviar mensagem para o servidor
61      public void sendServer(String message, String clientServer){
62          try{
63              window.disabledTable();
64              server.receiveMessage(message, clientServer);
65              if(message.equals(Messages.FINISH.getValue())){
66                  exitClient();
67              }
68          }catch (Exception e){
```

```
69                 e.printStackTrace();
70             }
71         }
72
73
74         private void exitClient(){
75             try{
76                 server.disconnect(window.getNamePlayer());
77                 System.exit(0);
78             }catch (Exception e){
79                 e.printStackTrace();
80             }
81         }
82 }
```

codes/RMI/Client/ClientImp.java

6.2.2 Server

Interface Remota do servidor

```
1  package RMI.Server;
2
3
4  import java.rmi.Remote;
5  import java.rmi.RemoteException;
6
7  /*
8  *
9  * Interface Remota do Servidor
10 */
11 public interface Server extends Remote {
12
13     // Metodo responsavel por receber e armazenar a conexao do cliente
14     void connect(String client) throws RemoteException;
15
16     // Metodo responsavel por receber a mensagem(jogada) do cliente
17     void receiveMessage(String message, String clientServer) throws
           RemoteException;
18
19     // Metodo responsavel por encerrar a conexao de determinado cliente
20     void disconnect(String client) throws RemoteException;
21
22 }
```

codes/RMI/Server/Server.java

Implementação da interface remota do servidor

```java
package RMI.Server;

import RMI.Client.Client;
import Utils.Messages;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;

public class ServerImp extends UnicastRemoteObject implements Server{
    private static final long serialVersionUID = 1L;

    private List<Client> clients = new ArrayList<Client>();
    private boolean client1Connection = false;
    private boolean client2Connection = false;
    public ServerImp() throws RemoteException {
       super();
    }

    @Override
    public void connect(String clientServer) throws RemoteException {
        if(clientServer != null && clients.size() <= 2){
          try{
              Client client = (Client) Naming.lookup(clientServer);
              clients.add(client);
          }catch (Exception e){
              e.printStackTrace();
          }
        }
        if(clients.size() == 2){
            clients.get(0).initializePlayer(Messages.CONNECTCLIENT1.
               getValue());
            client1Connection = true;
        }
    }

    @Override
    public void receiveMessage(String message, String clientServer) throws
```

```
        RemoteException {
42          try{
43              Client client = (Client) Naming.lookup(clientServer);
44              int index = clients.indexOf(client);
45              if(!message.equals(Messages.FINISH.getValue())){
46                  switch (index){
47                      case 0:
48                          if(!client2Connection){
49                              client2Connection = true;
50                              clients.get(1).initializePlayer(Messages.
                                    CONNECTCLIENT2.getValue());
51                          }
52                          clients.get(1).changePlayer(message);
53                          break;
54                      case 1:
55                          clients.get(0).changePlayer(message);
56                          break;
57                  }
58              }
59
60
61          }catch (Exception e){
62              e.printStackTrace();
63          }
64      }
65
66      @Override
67      public void disconnect(String clientServer) throws RemoteException {
68         try{
69              Client client = (Client) Naming.lookup(clientServer);
70              int index = clients.indexOf(client);
71             if(clients.size() != 0){
72                 switch (index){
73                     case 0:
74                         if(!client2Connection){
75                             client2Connection = true;
76                             clients.get(1).initializePlayer(Messages.
                                    CONNECTCLIENT2.getValue());
77                         }
78                         clients.get(1).changePlayer(Messages.FINISH.getValue
                                ());
79                         break;
80                     case 1:
81                         clients.get(0).changePlayer(Messages.FINISH.getValue
                                ());
82                         break;
83                 }
84             }
```

```
85              clients.clear();
86              System.exit(0);
87          }catch (Exception e){
88              e.printStackTrace();
89          }
90      }
91
92      // Disponibiliza os servi os do servidor
93      public  void startServer(){
94
95          try{
96              Registry registry = LocateRegistry.createRegistry(1099);
97              Server server = new ServerImp();
98              Naming.rebind("Servidor", server);
99          }catch (Exception e){
100             e.printStackTrace();
101         }
102     }
103
104 //     public static void main(String[] args) throws RemoteException {
105 //         new ServerImp();
106 //         startServer();
107 //     }
108 }
```

codes/RMI/Server/ServerImp.java

6.3 Interfaces

Esta interface implementada para uso de callback, é utilizada para dado certa iteração do usuário com a JFrame a mesma poder iteragir com a Main.

6.3.1 Aplication

Usada principalmente na inicialização do fluxo de comunicação, apartir da tela da JFrame

```
1 package Interface;
2
3 public interface Aplication {
4     void initialize();
5
```

```
6       void sendClient();
7   }
```

codes/Interfaces/Aplication.java

6.4 Utilitários

6.4.1 Messages

Trata-se de um Enum - Java, que abstrai algumas mensagens padrão para comunicação de ambos clientes e servidores.

```
1   package Utils;
2
3   public enum Messages {
4
5       CONNECTCLIENT1((String) "1"),
6       CONNECTCLIENT2((String) "2"),
7
8       CONNECTSERVER((String) "3"),
9       FINISH((String) "4");
10
11
12      private final String value;
13
14      Messages(String value){
15          this.value = value;
16      }
17
18      public String getValue(){
19          return  value;
20      }
21
22      public String getValueString() {
23          return  String.valueOf(value);
24      }
25
26  }
```

codes/Utils/Messages.java

6.4.2 Window

Por fim a classe responsável por criação e configuração da JFrame, na mesma se encontra toda a lógica e funcionalidade da janela iterativa java.

```java
package Utils;

import Interface.Aplication;
import Main.Main;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Objects;

public class Window  {

    /*
     * Variables
     * */
    JFrame jFrame = new JFrame();
    Font big = new Font("Serif", Font.BOLD, 30);
    Font small = new Font("Serif", Font.BOLD, 15);
    Font button = new Font("Tahoma", Font.BOLD, 90);

    String[][] table = new String[3][3];

    String idPlayer;
    String namePlayer;
    String move = "";

    // Type Aplication
    JRadioButton optionServer;
    JRadioButton optionClient;

    // Gamer

    JLabel gamer;
    JTextField nameGamer;

    // Connection
    JButton connectionAplication;

    // Table buttons
    JButton btn1;
    JButton btn2;
```

```
43      JButton btn3;
44      JButton btn4;
45      JButton btn5;
46      JButton btn6;
47      JButton btn7;
48      JButton btn8;
49      JButton btn9;
50
51
52      public Window() {
53          initComponents();
54      }
55
56      private void initComponents() {
57          setConfigsWindow();
58          // Title
59          JLabel header = new JLabel("Aplica   o de conex o RMI");
60          jFrame.add(header);
61          header.setFont(big);
62          header.setBounds(100, 10, 600, 50);
63
64          // Connexion
65          JLabel typeConnection = new JLabel("Tipo de Conex o:");
66          jFrame.add(typeConnection);
67          typeConnection.setFont(small);
68          typeConnection.setBounds(100, 80, 150, 20);
69
70
71          ButtonGroup groupTypeConnection = new ButtonGroup();
72
73          optionServer = new JRadioButton("Servidor");
74          jFrame.add(optionServer);
75          optionServer.setBounds(100, 105, 150, 20);
76
77
78          optionClient = new JRadioButton("Cliente");
79          jFrame.add(optionClient);
80          optionClient.setBounds(100, 125, 150, 20);
81
82
83          groupTypeConnection.add(optionServer);
84          groupTypeConnection.add(optionClient);
85
86          // Gamer
87          gamer = new JLabel("Jogador:");
88          jFrame.add(gamer);
89          gamer.setFont(small);
90          gamer.setBounds(280, 80, 150, 20);
```

```
91
92          nameGamer = new JTextField(30);
93          jFrame.add(nameGamer);
94          nameGamer.setBounds(280, 110, 150, 20);
95          nameGamer.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1,
                Color.BLACK));
96
97          // Initialize Connexion
98          connectionAplication = new JButton("Conectar");
99          jFrame.add(connectionAplication);
100         connectionAplication.setFont(small);
101         connectionAplication.setBounds(500, 110, 140, 40);
102         connectionAplication.addActionListener(new ActionListener() {
103             @Override
104             public void actionPerformed(ActionEvent actionEvent) {
105                 initializeAplication();
106             }
107         });
108
109
110         // Funcionality
111         JLabel functionality = new JLabel("Funcionalidade:");
112         jFrame.add(functionality);
113         functionality.setFont(small);
114         functionality.setBounds(100, 210, 150, 20);
115
116
117         JLabel typeFunctionality = new JLabel("Jogo da Velha");
118         jFrame.add(typeFunctionality);
119         typeFunctionality.setFont(small);
120         typeFunctionality.setBounds(350, 210, 150, 20);
121
122
123         // Buttons
124
125         btn1 = new javax.swing.JButton();
126         jFrame.add(btn1);
127         btn1.setFont(button);
128         btn1.setBounds(200, 250, 140, 140);
129         btn1.addActionListener(new ActionListener() {
130             @Override
131             public void actionPerformed(ActionEvent actionEvent) {
132                 btn1ActionPerformed();
133             }
134         });
135
136
137         btn2 = new javax.swing.JButton();
```

```
138         jFrame.add(btn2);
139         btn2.setFont(button);
140         btn2.setBounds(350, 250, 140, 140);
141         btn2.addActionListener(new ActionListener() {
142             @Override
143             public void actionPerformed(ActionEvent actionEvent) {
144                 btn2ActionPerformed();
145             }
146         });
147 
148 
149         btn3 = new javax.swing.JButton();
150         jFrame.add(btn3);
151         btn3.setFont(button);
152         btn3.setBounds(500, 250, 140, 140);
153         btn3.addActionListener(new ActionListener() {
154             @Override
155             public void actionPerformed(ActionEvent actionEvent) {
156                 btn3ActionPerformed();
157             }
158         });
159 
160 
161         btn4 = new javax.swing.JButton();
162         jFrame.add(btn4);
163         btn4.setFont(button);
164         btn4.setBounds(200, 400, 140, 140);
165         btn4.addActionListener(new ActionListener() {
166             @Override
167             public void actionPerformed(ActionEvent actionEvent) {
168                 btn4ActionPerformed();
169             }
170         });
171 
172 
173         btn5 = new javax.swing.JButton();
174         jFrame.add(btn5);
175         btn5.setFont(button);
176         btn5.setBounds(350, 400, 140, 140);
177         btn5.addActionListener(new ActionListener() {
178             @Override
179             public void actionPerformed(ActionEvent actionEvent) {
180                 btn5ActionPerformed();
181             }
182         });
183 
184 
185         btn6 = new javax.swing.JButton();
```

```
186         jFrame.add(btn6);
187         btn6.setFont(button);
188         btn6.setBounds(500, 400, 140, 140);
189         btn6.addActionListener(new ActionListener() {
190             @Override
191             public void actionPerformed(ActionEvent actionEvent) {
192                 btn6ActionPerformed();
193             }
194         });
195 
196 
197         btn7 = new javax.swing.JButton();
198         jFrame.add(btn7);
199         btn7.setFont(button);
200         btn7.setBounds(200, 550, 140, 140);
201         btn7.addActionListener(new ActionListener() {
202             @Override
203             public void actionPerformed(ActionEvent actionEvent) {
204                 btn7ActionPerformed();
205             }
206         });
207 
208 
209         btn8 = new javax.swing.JButton();
210         jFrame.add(btn8);
211         btn8.setFont(button);
212         btn8.setBounds(350, 550, 140, 140);
213         btn8.addActionListener(new ActionListener() {
214             @Override
215             public void actionPerformed(ActionEvent actionEvent) {
216                 btn8ActionPerformed();
217             }
218         });
219 
220 
221         btn9 = new javax.swing.JButton();
222         jFrame.add(btn9);
223         btn9.setFont(button);
224         btn9.setBounds(500, 550, 140, 140);
225         btn9.addActionListener(new ActionListener() {
226             @Override
227             public void actionPerformed(ActionEvent actionEvent) {
228                 btn9ActionPerformed();
229             }
230         });
231 
232 
233     }//
```

```
234
235     private void setConfigsWindow() {
236         jFrame.setTitle("Aplica   o RMI");
237         jFrame.setSize(800, 800);
238         jFrame.setLocationRelativeTo(null);
239         jFrame.setResizable(false);
240         jFrame.setVisible(true);
241         jFrame.setLayout(null);
242         jFrame.setDefaultCloseOperation(javax.swing.WindowConstants.
                EXIT_ON_CLOSE);
243     }
244
245     private void btn1ActionPerformed() {
246
247         if (Objects.equals(idPlayer, "1")) {
248             btn1.setText("X");
249             btn1.setEnabled(false);
250             table[0][0] = "X";
251             move = "00X";
252             verifyWinner();
253         } else {
254             if (Objects.equals(idPlayer, "2")) {
255                 btn1.setText("O");
256                 btn1.setEnabled(false);
257                 table[0][0] = "O";
258                 move = "00O";
259                 verifyWinner();
260             }
261         }
262     }
263
264     private void btn2ActionPerformed() {
265
266         if (Objects.equals(idPlayer, "1")) {
267             btn2.setText("X");
268             btn2.setEnabled(false);
269             table[0][1] = "X";
270             move = "01X";
271             verifyWinner();
272         } else {
273             if (Objects.equals(idPlayer, "2")) {
274                 btn2.setText("O");
275                 btn2.setEnabled(false);
276                 table[0][1] = "O";
277                 move = "01O";
278                 verifyWinner();
279             }
280         }
```

```
281     }
282
283     private void btn3ActionPerformed() {
284
285         if (Objects.equals(idPlayer, "1")) {
286             btn3.setText("X");
287             btn3.setEnabled(false);
288             table[0][2] = "X";
289             move = "02X";
290             verifyWinner();
291         } else {
292             if (Objects.equals(idPlayer, "2")) {
293                 btn3.setText("O");
294                 btn3.setEnabled(false);
295                 table[0][2] = "O";
296                 move = "02O";
297                 verifyWinner();
298             }
299         }
300     }
301
302     private void btn4ActionPerformed() {
303
304         if (Objects.equals(idPlayer, "1")) {
305             btn4.setText("X");
306             btn4.setEnabled(false);
307             table[1][0] = "X";
308             move = "10X";
309             verifyWinner();
310         } else {
311             if (Objects.equals(idPlayer, "2")) {
312                 btn4.setText("O");
313                 btn4.setEnabled(false);
314                 table[1][0] = "O";
315                 move = "10O";
316                 verifyWinner();
317             }
318         }
319     }
320
321     private void btn5ActionPerformed() {
322
323         if (Objects.equals(idPlayer, "1")) {
324             btn5.setText("X");
325             btn5.setEnabled(false);
326             table[1][1] = "X";
327             move = "11X";
328             verifyWinner();
```

```
329         } else {
330             if (Objects.equals(idPlayer, "2")) {
331                 btn5.setText("O");
332                 btn5.setEnabled(false);
333                 table[1][1] = "O";
334                 move = "11O";
335                 verifyWinner();
336             }
337         }
338     }
339 
340     private void btn6ActionPerformed() {
341 
342         if (Objects.equals(idPlayer, "1")) {
343             btn6.setText("X");
344             btn6.setEnabled(false);
345             table[1][2] = "X";
346             move = "12X";
347             verifyWinner();
348         } else {
349             if (Objects.equals(idPlayer, "2")) {
350                 btn6.setText("O");
351                 btn6.setEnabled(false);
352                 table[1][2] = "O";
353                 move = "12O";
354                 verifyWinner();
355             }
356         }
357     }
358 
359     private void btn7ActionPerformed() {
360 
361         if (Objects.equals(idPlayer, "1")) {
362             btn7.setText("X");
363             btn7.setEnabled(false);
364             table[2][0] = "X";
365             move = "20X";
366             verifyWinner();
367         } else {
368             if (Objects.equals(idPlayer, "2")) {
369                 btn7.setText("O");
370                 btn7.setEnabled(false);
371                 table[2][0] = "O";
372                 move = "20O";
373                 verifyWinner();
374             }
375         }
376     }
```

```
377
378     private void btn8ActionPerformed() {
379
380         if (Objects.equals(idPlayer, "1")) {
381             btn8.setText("X");
382             btn8.setEnabled(false);
383             table[2][1] = "X";
384             move = "21X";
385             verifyWinner();
386         } else {
387             if (Objects.equals(idPlayer, "2")) {
388                 btn8.setText("O");
389                 btn8.setEnabled(false);
390                 table[2][1] = "O";
391                 move = "21O";
392                 verifyWinner();
393             }
394         }
395     }
396
397     private void btn9ActionPerformed() {
398
399         if (Objects.equals(idPlayer, "1")) {
400             btn9.setText("X");
401             btn9.setEnabled(false);
402             table[2][2] = "X";
403             move = "22X";
404             verifyWinner();
405         } else {
406             if (Objects.equals(idPlayer, "2")) {
407                 btn9.setText("O");
408                 btn9.setEnabled(false);
409                 table[2][2] = "O";
410                 move = "22O";
411                 verifyWinner();
412             }
413         }
414     }
415
416     private void initializeAplication() {
417         if (getAplicationConnection() == null) {
418             alertMessage("   preciso informar um tipo de Aplica   o!");
419         } else {
420             if (Objects.equals(getAplicationConnection(), "Server")) {
421                 isServer();
422                 startAplication();
423             } else {
424                 if (nameGamer.getText().isEmpty()) {
```

```
425                     alertMessage("    preciso informar um nome!");
426                 } else {
427                     namePlayer = nameGamer.getText();
428                     startAplication();
429                 }
430             }
431 
432         }
433     }
434 
435     private void sendMove(){
436         Aplication inOut = new Main();
437         inOut.sendClient();
438     }
439 
440     public String getNamePlayer(){
441         return namePlayer;
442     }
443 
444     private void startAplication() {
445         connectionAplication.setEnabled(false);
446         Aplication aplication = new Main();
447         aplication.initialize();
448     }
449 
450     private void verifyWinner() {
451         //Check Lines
452         for (int i = 0; i < 3; i++) {
453             if (table[i][0] == "X" && table[i][1] == "X" && table[i][2] ==
                    "X") {
454                 JOptionPane.showMessageDialog(null, nameGamer.getText() +
                        " ganhador!!!");
455                 endOfTheGame();
456             }
457             if (table[i][0] == "O" && table[i][1] == "O" && table[i][2] ==
                    "O") {
458                 JOptionPane.showMessageDialog(null, nameGamer.getText() +
                        " ganhador!!!");
459                 endOfTheGame();
460             }
461         }
462 
463         //Check Columns
464         for (int i = 0; i < 3; i++) {
465             if (table[0][i] == "X" && table[1][i] == "X" && table[2][i] ==
                    "X") {
466                 JOptionPane.showMessageDialog(null, nameGamer.getText() +
                        " ganhador!!!");
```

```
467                 endOfTheGame();
468             }
469             if (table[0][i] == "O" && table[1][i] == "O" && table[2][i] ==
                    "O") {
470                 JOptionPane.showMessageDialog(null, nameGamer.getText() +
                        " ganhador!!!");
471                 endOfTheGame();
472             }
473         }
474
475         //Check Main Diagonal
476         if (table[0][0] == "X" && table[1][1] == "X" && table[2][2] == "X"
                ) {
477             JOptionPane.showMessageDialog(null, nameGamer.getText() + "
                    ganhador!!!");
478             endOfTheGame();
479         }
480         if (table[0][0] == "O" && table[1][1] == "O" && table[2][2] == "O"
                ) {
481             JOptionPane.showMessageDialog(null, nameGamer.getText() + "
                    ganhador!!!");
482             endOfTheGame();
483         }
484
485         //Check Secondary Diagonal
486         if (table[0][2] == "X" && table[1][1] == "X" && table[2][0] == "X"
                ) {
487             JOptionPane.showMessageDialog(null, nameGamer.getText() + "
                    ganhador!!!");
488             endOfTheGame();
489         }
490         if (table[0][2] == "O" && table[1][1] == "O" && table[2][0] == "O"
                ) {
491             JOptionPane.showMessageDialog(null, nameGamer.getText() + "
                    ganhador!!!");
492             endOfTheGame();
493         }
494
495         verifyLoss();
496     }
497
498     private void verifyLoss() {
499         if(!move.equals(Messages.FINISH.getValueString())){
500             boolean loss = true;
501
502             for (int i = 0; i < 3; i++) {
503                 for (int j = 0; j < 3; j++) {
504                     if (table[i][j] == null) {
```

```
505                         loss = false;
506                     }
507                 }
508             }
509 
510             if (loss) {
511                 JOptionPane.showMessageDialog(null, "VELHA! :(");
512                 endOfTheGame();
513             }
514         }
515 
516         sendMove();
517     }
518 
519     public void disabledTable() {
520         btn1.setEnabled(false);
521         btn2.setEnabled(false);
522         btn3.setEnabled(false);
523         btn4.setEnabled(false);
524         btn5.setEnabled(false);
525         btn6.setEnabled(false);
526         btn7.setEnabled(false);
527         btn8.setEnabled(false);
528         btn9.setEnabled(false);
529     }
530 
531     public void enableTable() {
532         btn1.setEnabled(true);
533         btn2.setEnabled(true);
534         btn3.setEnabled(true);
535         btn4.setEnabled(true);
536         btn5.setEnabled(true);
537         btn6.setEnabled(true);
538         btn7.setEnabled(true);
539         btn8.setEnabled(true);
540         btn9.setEnabled(true);
541     }
542 
543     public void updateTable(String coordinates) {
544         enableTable();
545         String[] values = coordinates.split("");
546         int x = Integer.parseInt(values[0]);
547         int y = Integer.parseInt(values[1]);
548         if (table[x][y] == null) {
549             table[x][y] = values[2];
550             updateTableButtons();
551         }
552     }
```

```
553     private void updateTableButtons() {
554         if (table[0][0] != null) {
555             btn1.setText(table[0][0]);
556             btn1.setEnabled(false);
557         }
558         if (table[0][1] != null) {
559             btn2.setText(table[0][1]);
560             btn2.setEnabled(false);
561         }
562         if (table[0][2] != null) {
563             btn3.setText(table[0][2]);
564             btn3.setEnabled(false);
565         }
566         if (table[1][0] != null) {
567             btn4.setText(table[1][0]);
568             btn4.setEnabled(false);
569         }
570         if (table[1][1] != null) {
571             btn5.setText(table[1][1]);
572             btn5.setEnabled(false);
573         }
574         if (table[1][2] != null) {
575             btn6.setText(table[1][2]);
576             btn6.setEnabled(false);
577         }
578         if (table[2][0] != null) {
579             btn7.setText(table[2][0]);
580             btn7.setEnabled(false);
581         }
582         if (table[2][1] != null) {
583             btn8.setText(table[2][1]);
584             btn8.setEnabled(false);
585         }
586         if (table[2][2] != null) {
587             btn9.setText(table[2][2]);
588             btn9.setEnabled(false);
589         }
590 
591     }
592 
593     public void alertMessage(String message) {
594         JOptionPane.showMessageDialog(null, message);
595     }
596 
597     public String getAplicationConnection() {
598         if (optionServer.isSelected()) {
599             return "Server";
600         }
```

```
601             if (optionClient.isSelected()) {
602                 return "Client";
603             }
604             return null;
605         }
606 
607         public void endOfTheGame(){
608             move = Messages.FINISH.getValueString();
609         }
610         public String getMove(){
611             return move;
612         }
613 
614 
615         public void isServer() {
616             disabledTable();
617             nameGamer.setText("Servidor");
618             nameGamer.setEditable(false);
619             disableOptionsConnections();
620             alertMessage("Servidor Inicializado");
621         }
622 
623         public void isClient(String idPlayer){
624             disabledTable();
625             nameGamer.setEditable(false);
626             this.idPlayer = idPlayer;
627             disableOptionsConnections();
628         }
629         private void disableOptionsConnections(){
630             optionClient.setEnabled(false);
631             optionServer.setEnabled(false);
632         }
633 
634 }
```

codes/Utils/Window.java

Referências

[1] *RMI*, . URL https://pt.wikipedia.org/wiki/RMI.

[2] *Wireshark*, . URL https://www.wireshark.org/.