Trabalho Prático: Soquetes

1 Introdução

O trabalho realizado propõe implementar um programa que permita que duas pessoas joguem uma partida de jogo da velha em um ambiente de rede.

Outro objetivo do trabalho é demonstrar de forma prática as diferenças entre o tráfego de informações usando o protocolo UDP e o protocolo TCP, não só tornando visíveis as diferenças entre os resultados mas também permitir a demonstração das diferenças entre as implementações necessárias para o uso de cada protocolo.

2 Metodologia

O software foi implementado na linguagem de programação Java, devido à sua portabilidade entre diferentes máquinas e sistemas e a implementação prévia de diversas classes para a abstração de sockets, datagramas e outras estruturas necessárias para a comunicação.

Isso permitiu que o esforço fosse focado na estrutura de funcionamento do software e na solução de problemas inerentes ao ambiente distribuído, liberando os autores de lidar com a implementação específica do protocolo e das características de hardware dos sistemas que executam o programa e da rede que os interliga.

Para a criação da interface com o usuário foi utilizado o designer de JDialog disponível na IDE Netbeans 8.0.2, que gerou automaticamente a seção do código-fonte responsável pela criação e visualização da interface, cabendo aos desenvolvedores definir o modelo de tela e implementar a comunicação entre os elementos da interface e as ações a serem executadas pelo algoritmo.

O código-fonte foi dividido em 3 (três) pacotes: "gui", "ctrl" e "net". Esses pacotes buscam organizar a estrutura do programa adaptando o renomado modelo MVC.

O pacote "gui" contém a classe Screen, responsável pela seção de visão do software, implementando a interface com o usuário.

Já o pacote "ctrl" contém a classe TicTacToe, responsável pela seção de controle, implementado o algoritmo do jogo da velha, realizando jogadas e determinando o resultado da partida.

Por último, o pacote "net" substitui a seção de modelo já que não há necessidade para persistência de dados. Esse pacote contém o conjunto de classes utilizadas para realizar a comunicação entre as instâncias do programa utilizando uma arquitetura *peer-to-peer* e os protocolos UDP e TCP.

3 Resultados

O software resultante possui uma interface na qual o usuário pode informar o endereco de rede de um outro usuario, a porta do processo na outra máquina e o protocolo a ser usado ou esperar outro host da rede se conectar a seu jogo.

O número da porta de cada protocolo é imprimido no console da aplicação dos usuários, indicando que o programa estará escutando naquelas portas por tentativas de conexões.

Assim que conectados um ao outro, seja com uma conexão TCP ou com uma simulação de conexão UDP, o jogo começa, com o usuário que buscou a conexão realizando a primeira jogada.

A interface possui 9 (nove) quadrados que simulam um tabuleiro virtual de jogo da velha, no qual a jogada realizada pelo jogador local são marcadas de azul e as realizadas pelo oponente em vermelho.

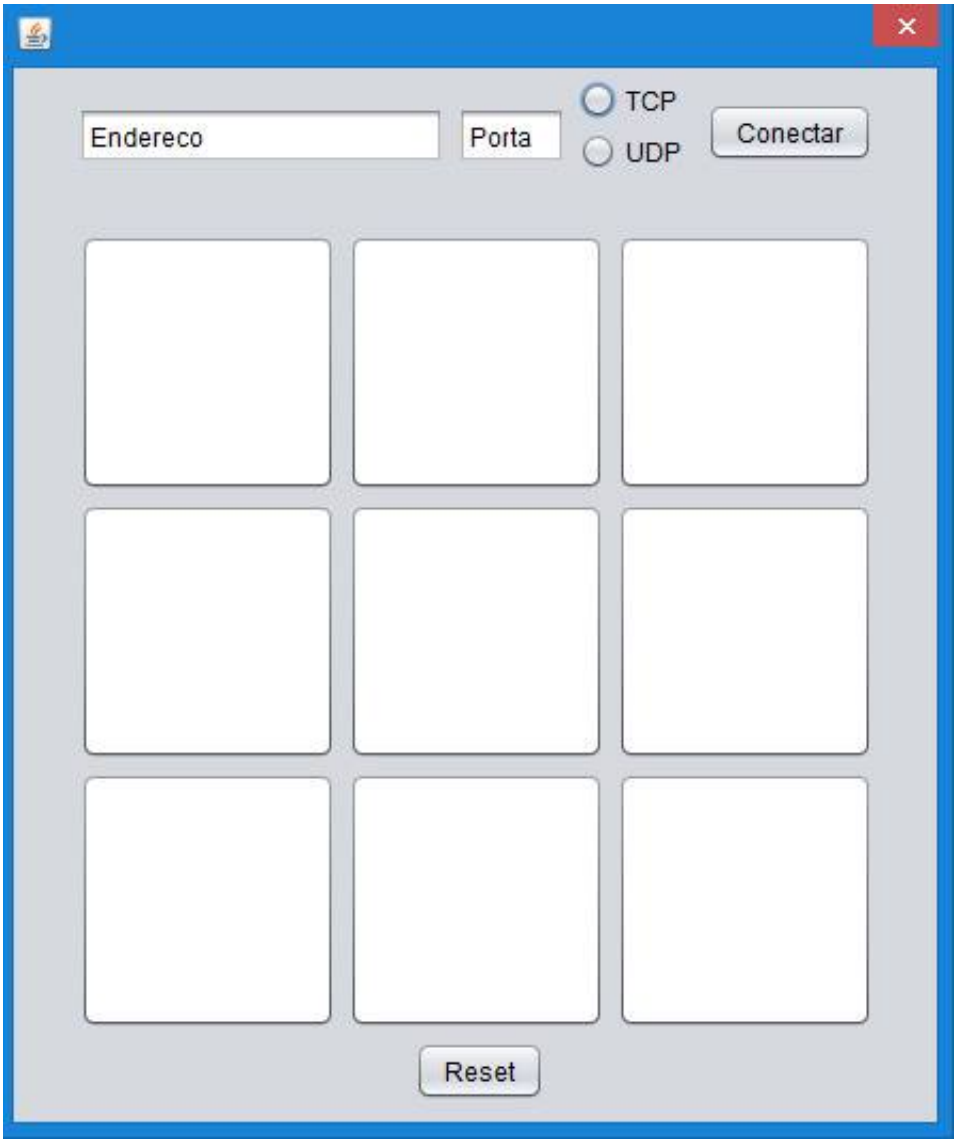

Figura 1: Interface do software

4 Análise

Foram encontrados alguns problemas durante o desenvolvimento do programa que fizeram com que os resultados esperados não fossem atingidos.

Para conexões do tipo TCP houve um comportamente inesperado em todos os testes executados, nos quais o jogador que iniciava a partida realizava sua jogada, enviando um pacote pela rede usando a conexão previamente estabelecida. Mas o outro host, após perceber que alguém havia se conectado ao jogo, esperava indefinidamente por uma jogada do oponente, não sendo capaz de ler o pacote enviado e entrando em um *loop* infinito.

Já no caso do protocolo UDP o jogador que inicia a partida realiza sua jogada e espera pela jogada de resposta do oponente. Enquanto isso, o outro host recebe a jogada, marca a devida posição como estando ocupada pelo oponente e aguarda o jogador local escolher uma posição para jogar. Quando o jogador a escolhe a jogada não é recebida pelo outro host e ambos passam a esperar indefinidamente por uma jogada do oponente, entrando também em um *loop* infinito.

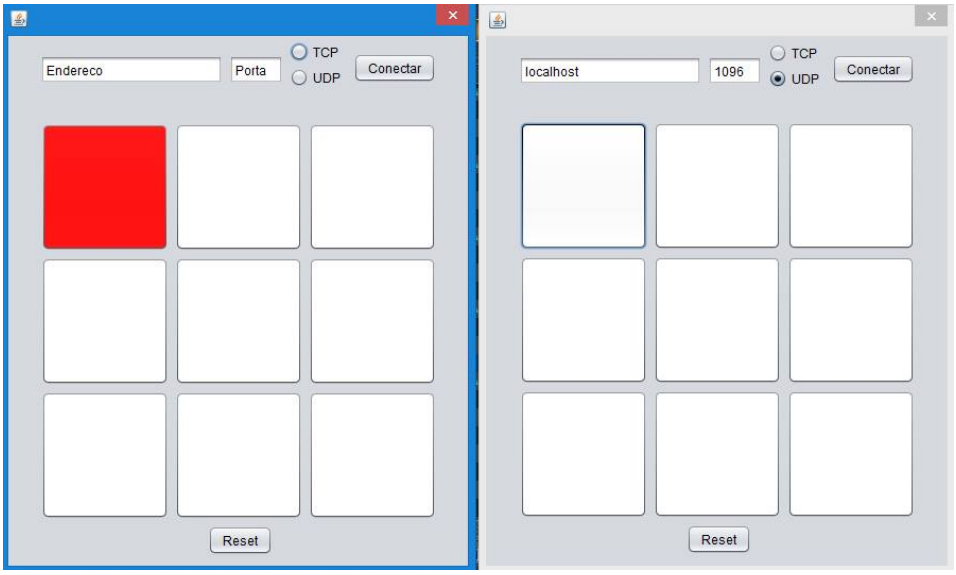

Figura 2: *Loop* infinito utilizando UDP

Outro problema foi gerenciar o sincronismo entre as aplicações, fazendo com que cada uma fosse capaz de enviar pacotes para a outra e receber sem que a ordem do jogo fosse comprometida. Para que tal requerimento fosse atingido foi necessário bloquear a aplicação enquanto esperava para leitura, situação que pode ter vindo a causar os problemas supra-citados.

5 Conclusão

A realização deste trabalho possibilitou ao grupo perceber as dificuldades da implementação de um sistema distribuído, principalmente no que tange a confiabilidade e o sincronismo da comunicação entre os processos.

Além disso, pode se perceber como certas arquiteturas de rede, como a *peer-to-peer*, dificultam o gerenciamento da lógica da rede pela aplicação e como os conceitos de sistemas distribuídos auxiliam na superação dessas dificuldades.

6 Código

6.1 Pacote "gui"

```
1  package gui;
2
3  import ctrl.TicTacToe;
4  import java.awt.Color;
5  import javax.swing.JOptionPane;
6
7  public class Screen extends javax.swing.JDialog {
8
9      private final TicTacToe game;
10     private final Color[] playerColor;
11
12     /**
13      * Creates new form Screen
14      * @param parent
15      * @param modal
16      */
17
18     private boolean isTcp;
19
20     public Screen(java.awt.Frame parent, boolean modal) {
21         super(parent, modal);
22         initComponents();
23         bgTipo.add(rdTCP);
24         bgTipo.add(rdUDP);
25         isTcp = true;
26         playerColor = new Color[3];
27         playerColor[0] = new Color(255, 255, 255);
28         playerColor[1] = new Color(0, 0, 255);
29         playerColor[2] = new Color(255, 0, 0);
30         game = new TicTacToe(this);
31     }
32
33     /**
34      * This method is called from within the constructor to initialize the form.
```

```
35      * WARNING: Do NOT modify this code. The content of this method is always
36      * regenerated by the Form Editor.
37      */
38     @SuppressWarnings("unchecked")
39     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
40     private void initComponents() {
41
42         bgTipo = new javax.swing.ButtonGroup();
43         jPanel1 = new javax.swing.JPanel();
44         txtAddress = new javax.swing.JTextField();
45         rdTCP = new javax.swing.JRadioButton();
46         rdUDP = new javax.swing.JRadioButton();
47         btnConnect = new javax.swing.JButton();
48         btn2 = new javax.swing.JButton();
49         btn3 = new javax.swing.JButton();
50         btn1 = new javax.swing.JButton();
51         btn4 = new javax.swing.JButton();
52         btn5 = new javax.swing.JButton();
53         btn6 = new javax.swing.JButton();
54         btn7 = new javax.swing.JButton();
55         btn9 = new javax.swing.JButton();
56         btn8 = new javax.swing.JButton();
57         btnReset = new javax.swing.JButton();
58         txtPort = new javax.swing.JTextField();
59
60         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
61
62         txtAddress.setText("Endereco");
63
64         rdTCP.setText("TCP");
65         rdTCP.addActionListener(new java.awt.event.ActionListener() {
66             public void actionPerformed(java.awt.event.ActionEvent evt) {
67                 rdTCPActionPerformed(evt);
68             }
69         });
70
71         rdUDP.setText("UDP");
72         rdUDP.addActionListener(new java.awt.event.ActionListener() {
73             public void actionPerformed(java.awt.event.ActionEvent evt) {
74                 rdUDPActionPerformed(evt);
75             }
76         });
77
78         btnConnect.setText("Conectar");
79         btnConnect.addActionListener(new java.awt.event.ActionListener() {
80             public void actionPerformed(java.awt.event.ActionEvent evt) {
81                 btnConnectActionPerformed(evt);
82             }
83         });
84
85         btn2.addActionListener(new java.awt.event.ActionListener() {
86             public void actionPerformed(java.awt.event.ActionEvent evt) {
87                 btn2ActionPerformed(evt);
88             }
89         });
90
91         btn3.addActionListener(new java.awt.event.ActionListener() {
92             public void actionPerformed(java.awt.event.ActionEvent evt) {
93                 btn3ActionPerformed(evt);
94             }
95         });
```

```
96
97          btn1.addActionListener(new java.awt.event.ActionListener() {
98              public void actionPerformed(java.awt.event.ActionEvent evt) {
99                  btn1ActionPerformed(evt);
100             }
101         });
102
103         btn4.addActionListener(new java.awt.event.ActionListener() {
104             public void actionPerformed(java.awt.event.ActionEvent evt) {
105                 btn4ActionPerformed(evt);
106             }
107         });
108
109         btn5.addActionListener(new java.awt.event.ActionListener() {
110             public void actionPerformed(java.awt.event.ActionEvent evt) {
111                 btn5ActionPerformed(evt);
112             }
113         });
114
115         btn6.addActionListener(new java.awt.event.ActionListener() {
116             public void actionPerformed(java.awt.event.ActionEvent evt) {
117                 btn6ActionPerformed(evt);
118             }
119         });
120
121         btn7.addActionListener(new java.awt.event.ActionListener() {
122             public void actionPerformed(java.awt.event.ActionEvent evt) {
123                 btn7ActionPerformed(evt);
124             }
125         });
126
127         btn9.addActionListener(new java.awt.event.ActionListener() {
128             public void actionPerformed(java.awt.event.ActionEvent evt) {
129                 btn9ActionPerformed(evt);
130             }
131         });
132
133         btn8.addActionListener(new java.awt.event.ActionListener() {
134             public void actionPerformed(java.awt.event.ActionEvent evt) {
135                 btn8ActionPerformed(evt);
136             }
137         });
138
139         btnReset.setText("Reset");
140         btnReset.addActionListener(new java.awt.event.ActionListener() {
141             public void actionPerformed(java.awt.event.ActionEvent evt) {
142                 btnResetActionPerformed(evt);
143             }
144         });
145
146         txtPort.setText("Porta");
147
148         javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
149         jPanel1.setLayout(jPanel1Layout);
150         jPanel1Layout.setHorizontalGroup(
151             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
152             .addGroup(jPanel1Layout.createSequentialGroup()
153                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                        .LEADING, false)
154                     .addGroup(jPanel1Layout.createSequentialGroup()
155                         .addGap(31, 31, 31)
```

```
156                         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                                Alignment.LEADING)
157                             .addGroup(jPanel1Layout.createSequentialGroup()
158                                 .addComponent(btn1, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE)
159                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
160                                 .addComponent(btn2, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE)
161                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
162                                 .addComponent(btn3, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE))
163                             .addGroup(jPanel1Layout.createSequentialGroup()
164                                 .addComponent(btn4, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE)
165                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
166                                 .addComponent(btn5, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE)
167                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
168                                 .addComponent(btn6, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE))
169                             .addGroup(jPanel1Layout.createSequentialGroup()
170                                 .addComponent(btn7, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE)
171                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
172                                 .addComponent(btn8, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE)
173                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
174                                 .addComponent(btn9, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        120, javax.swing.GroupLayout.PREFERRED_SIZE))))
175                     .addGroup(jPanel1Layout.createSequentialGroup()
176                         .addGap(30, 30, 30)
177                         .addComponent(txtAddress, javax.swing.GroupLayout.PREFERRED_SIZE,
                                172, javax.swing.GroupLayout.PREFERRED_SIZE)
178                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
179                         .addComponent(txtPort)
180                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
181                         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                                Alignment.LEADING, false)
182                             .addComponent(rdUDP, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                                    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
183                             .addComponent(rdTCP, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                                    swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
184                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED
                                )
185                         .addComponent(btnConnect)))
186                 .addContainerGap(30, Short.MAX_VALUE))
187             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.
                    createSequentialGroup()
188                 .addGap(0, 0, Short.MAX_VALUE)
189                 .addComponent(btnReset)
190                 .addGap(184, 184, 184))
191         );
192
193         jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component
                [] {btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9});
```

```
194
195         jPanel1Layout.setVerticalGroup(
196             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
197             .addGroup(jPanel1Layout.createSequentialGroup()
198                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                        .LEADING)
199                     .addGroup(jPanel1Layout.createSequentialGroup()
200                         .addContainerGap()
201                         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                                Alignment.LEADING)
202                             .addGroup(jPanel1Layout.createSequentialGroup()
203                                 .addComponent(rdTCP)
204                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
205                                 .addComponent(rdUDP))
206                             .addGroup(jPanel1Layout.createSequentialGroup()
207                                 .addGap(10, 10, 10)
208                                 .addComponent(btnConnect))))
209                     .addGroup(jPanel1Layout.createSequentialGroup()
210                         .addGap(18, 18, 18)
211                         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                                Alignment.BASELINE)
212                             .addComponent(txtAddress, javax.swing.GroupLayout.PREFERRED_SIZE,
                                    27, javax.swing.GroupLayout.PREFERRED_SIZE)
213                             .addComponent(txtPort, javax.swing.GroupLayout.PREFERRED_SIZE,
                                    27, javax.swing.GroupLayout.PREFERRED_SIZE))))
214                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 30,
                        Short.MAX_VALUE)
215                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                        .BASELINE)
216                     .addComponent(btn2, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE)
217                     .addComponent(btn3, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE)
218                     .addComponent(btn1, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE))
219                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
220                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                        .BASELINE)
221                     .addComponent(btn5, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE)
222                     .addComponent(btn6, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE)
223                     .addComponent(btn4, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE))
224                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
225                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                        .BASELINE)
226                     .addComponent(btn8, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE)
227                     .addComponent(btn9, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE)
228                     .addComponent(btn7, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
                            swing.GroupLayout.PREFERRED_SIZE))
229                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
230                 .addComponent(btnReset)
231                 .addGap(10, 10, 10))
232         );
233
234         jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
                {btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9});
```

```
235
236         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
237         getContentPane().setLayout(layout);
238         layout.setHorizontalGroup(
239             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
240             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
                GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
241         );
242         layout.setVerticalGroup(
243             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
244             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
                GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
245         );
246
247         pack();
248     }// </editor-fold>//GEN-END:initComponents
249
250     private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn1ActionPerformed
251         game.makePlay(1, 0);
252     }//GEN-LAST:event_btn1ActionPerformed
253
254     private void btn2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn2ActionPerformed
255         game.makePlay(1, 1);
256     }//GEN-LAST:event_btn2ActionPerformed
257
258     private void btn3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn3ActionPerformed
259         game.makePlay(1, 2);
260     }//GEN-LAST:event_btn3ActionPerformed
261
262     private void btn4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn4ActionPerformed
263         game.makePlay(1, 3);
264     }//GEN-LAST:event_btn4ActionPerformed
265
266     private void btn5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn5ActionPerformed
267         game.makePlay(1, 4);
268     }//GEN-LAST:event_btn5ActionPerformed
269
270     private void btn6ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn6ActionPerformed
271         game.makePlay(1, 5);
272     }//GEN-LAST:event_btn6ActionPerformed
273
274     private void btn7ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn7ActionPerformed
275         game.makePlay(1, 6);
276     }//GEN-LAST:event_btn7ActionPerformed
277
278     private void btn8ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn8ActionPerformed
279         game.makePlay(1, 7);
280     }//GEN-LAST:event_btn8ActionPerformed
281
282     private void btn9ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btn9ActionPerformed
283         game.makePlay(1, 8);
284     }//GEN-LAST:event_btn9ActionPerformed
```

```
285
286     private void rdTCPActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_rdTCPActionPerformed
287         isTcp = true;
288     }//GEN-LAST:event_rdTCPActionPerformed
289
290     private void rdUDPActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_rdUDPActionPerformed
291         isTcp = false;
292     }//GEN-LAST:event_rdUDPActionPerformed
293
294     private void btnResetActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btnResetActionPerformed
295         game.reset();
296     }//GEN-LAST:event_btnResetActionPerformed
297
298     private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
            event_btnConnectActionPerformed
299         game.connect(txtAddress.getText(), txtPort.getText(), isTcp);
300     }//GEN-LAST:event_btnConnectActionPerformed
301
302
303     public void changeButtonColor(int player, int button){
304         switch(button){
305             case 1:
306                 btn1.setBackground(playerColor[player]);
307                 break;
308             case 2:
309                 btn2.setBackground(playerColor[player]);
310                 break;
311             case 3:
312                 btn3.setBackground(playerColor[player]);
313                 break;
314             case 4:
315                 btn4.setBackground(playerColor[player]);
316                 break;
317             case 5:
318                 btn5.setBackground(playerColor[player]);
319                 break;
320             case 6:
321                 btn6.setBackground(playerColor[player]);
322                 break;
323             case 7:
324                 btn7.setBackground(playerColor[player]);
325                 break;
326             case 8:
327                 btn8.setBackground(playerColor[player]);
328                 break;
329             case 9:
330                 btn9.setBackground(playerColor[player]);
331         }
332     }
333
334     public void createDialog(String message, String title){
335         JOptionPane.showMessageDialog(jPanel1, message, title, JOptionPane.
                INFORMATION_MESSAGE);
336     }
337
338
339     /**
340      * @param args the command line arguments
```

```
341      */
342     public static void main(String args[]) {
343         /* Set the Nimbus look and feel */
344         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
                ">
345         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
                and feel.
346          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/
                plaf.html
347          */
348         try {
349             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
                    getInstalledLookAndFeels()) {
350                 if ("Nimbus".equals(info.getName())) {
351                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
352                     break;
353                 }
354             }
355         } catch (ClassNotFoundException ex) {
356         } catch (InstantiationException ex) {
357         } catch (IllegalAccessException ex) {
358         } catch (javax.swing.UnsupportedLookAndFeelException ex) {}
359         //</editor-fold>
360 
361         /* Create and display the dialog */
362         java.awt.EventQueue.invokeLater(new Runnable() {
363             public void run() {
364                 Screen dialog = new Screen(new javax.swing.JFrame(), true);
365                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
366                     @Override
367                     public void windowClosing(java.awt.event.WindowEvent e) {
368                         System.exit(0);
369                     }
370                 });
371                 dialog.setVisible(true);
372             }
373         });
374     }
375 
376     // Variables declaration - do not modify//GEN-BEGIN:variables
377     private javax.swing.ButtonGroup bgTipo;
378     private javax.swing.JButton btn1;
379     private javax.swing.JButton btn2;
380     private javax.swing.JButton btn3;
381     private javax.swing.JButton btn4;
382     private javax.swing.JButton btn5;
383     private javax.swing.JButton btn6;
384     private javax.swing.JButton btn7;
385     private javax.swing.JButton btn8;
386     private javax.swing.JButton btn9;
387     private javax.swing.JButton btnConnect;
388     private javax.swing.JButton btnReset;
389     private javax.swing.JPanel jPanel1;
390     private javax.swing.JRadioButton rdTCP;
391     private javax.swing.JRadioButton rdUDP;
392     private javax.swing.JTextField txtAddress;
393     private javax.swing.JTextField txtPort;
394     // End of variables declaration//GEN-END:variables
395 }
```

gui/Screen.java

6.2 Pacote “ctrl”

```
1  package ctrl;
2
3  import gui.Screen;
4  import net.Sender;
5
6  public final class TicTacToe {
7
8      private int[][] grid;
9      private int lastPlay;
10     private final Screen screen;
11     private final Sender sender;
12
13     public TicTacToe(Screen screen) {
14         this.screen = screen;
15         sender = new Sender(this);
16         reset();
17     }
18
19     public void reset() {
20         lastPlay = 2;
21         grid = new int[3][3];
22         for(int i = 0; i < 3; i++)
23             for(int j = 0; j < 3; j++)
24                 grid[i][j] = 0;
25         for(int i = 0; i < 10; i++)
26             screen.changeButtonColor(0, i);
27     }
28
29     public int winner() {
30         for(int player = 1; player <= 2; player++){
31             if((grid[0][0] == grid[1][1] && grid[0][0] == grid[2][2] && grid[0][0] == player)
                    ||
32                (grid[0][2] == grid[1][1] && grid[0][2] == grid[2][0] && grid[0][2] == player)
                    )
33                 return player;
34             for(int i = 0; i < 3; i++)
35                 if((grid[i][0] == grid[i][1] && grid[i][0] == grid[i][2] && grid[i][0] ==
                        player) ||
36                    (grid[0][i] == grid[1][i] && grid[0][i] == grid[2][i] && grid[0][i] ==
                        player))
37                     return player;
38         }
39         return 0;
40     }
41
42     public void makePlay(int player, int pos) {
43         if(grid[ pos / 3 ][ pos % 3 ] == 0 && !isOver() && lastPlay != player){
44             lastPlay = player;
45             grid[ pos / 3 ][ pos % 3 ] = player;
46             screen.changeButtonColor(player, pos+1);
47             if(player == 1){
48                 sender.writePlay(pos);
49                 sender.readPlay();
50             }
51         }
52     }
53
54     public void connect(String address, String port, boolean isTcp) {
```

```
55          sender.connect(address, port, isTcp);
56          lastPlay = 2;
57      }
58
59      public void connected(){
60          lastPlay = 1;
61          sender.readPlay();
62      }
63
64      public boolean isOver() {
65          if(winner() != 0){
66              screen.createDialog("Jogador " + winner() + " venceu!", "Resultado");
67              return true;
68          }
69
70          for(int i = 0; i < 3; i++)
71              for(int j = 0; j < 3; j++)
72                  if(grid[i][j] == 0)
73                      return false;
74
75          screen.createDialog("Deu velha!", "Resultado");
76          return true;
77      }
78
79      public void setLastPlay(int lastPlay) {
80          this.lastPlay = lastPlay;
81      }
82  }
```

ctrl/TicTacToe.java

6.3 Pacote "net"

```java
package net;

import ctrl.TicTacToe;
import java.io.IOException;

public class Sender {

    private final TicTacToe game;
    private final TcpServer tcp;
    private final UdpServer udp;
    private boolean isTcp;

    public Sender(TicTacToe t){
        game = t;
        tcp = new TcpServer(this);
        udp = new UdpServer(this);
    }

    public void connect(String address, String port, boolean isTcp) {
        this.isTcp = isTcp;

        if(isTcp)
            tcp.connect(address, port);
        else
            udp.connect(address, port, null);
    }

    public void writePlay(int pos) {
        if(isTcp)
            tcp.write(pos + "\n");
        else
            udp.write(pos + "\n");
    }

    public void readPlay() {
        String read = "";
        if(isTcp)
            while("".equals(read)){
                try { read = tcp.read(); }
                catch (IOException ex) { ex.printStackTrace(); }
            }
        else
            while("".equals(read)){
                try { read = udp.read(); }
                catch (IOException ex) { ex.printStackTrace(); }
            }

        game.makePlay(2, Integer.parseInt(read));
    }

    public void connected(boolean isTcp) {
        this.isTcp = isTcp;
        game.connected();
    }

    public void makePlay(int player, int pos){
        game.setLastPlay(1);
        game.makePlay(2, pos);
```

```
59      }
60
61  }
```

net/Sender.java

```
1   package net;
2
3   import java.io.IOException;
4   import java.net.ServerSocket;
5   import java.net.Socket;
6
7   public class TcpListener extends Thread {
8
9       private ServerSocket serverSocket;
10      private TcpServer server;
11
12      public TcpListener(TcpServer server, int port) throws IOException {
13          this.server = server;
14          serverSocket = new ServerSocket(port);
15      }
16
17      @Override
18      public void run(){
19          Socket socket = null;
20          while(socket == null){
21              try{ socket = serverSocket.accept(); }
22              catch (IOException ex) { ex.printStackTrace(); }
23          }
24          server.connected(socket);
25      }
26
27  }
```

net/TcpListener.java

```
1   package net;
2
3   import java.io.BufferedReader;
4   import java.io.BufferedWriter;
5   import java.io.IOException;
6   import java.io.InputStreamReader;
7   import java.io.OutputStreamWriter;
8   import java.net.Socket;
9
10  public class TcpServer{
11
12      private final Sender sender;
13      private TcpListener tcp;
14      private Socket socket;
15      private BufferedReader in;
16      private BufferedWriter out;
17
18      public TcpServer(Sender sender) {
19          this.sender = sender;
20          clear();
21      }
22
23      public final void clear(){
24          try {
```

```
            int p = (int) (Math.random() * 100 + 1000);
            System.out.println("porta tcp: " + p);
            tcp = new TcpListener(this, p);
            tcp.start();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        socket = null;
    }

    public void setBuffers(Socket socket){
        this.socket = socket;
        try {
            in =  new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public void connect(String address, String port) {
        try {
            socket = new Socket(address, Integer.parseInt(port));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        setBuffers(socket);
    }

    public void connected(Socket socket) {
        setBuffers(socket);
        tcp = null;
        sender.connected(true);
    }

    public void write(String data){
        try {
            out.write(data);
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public String read() throws IOException {
        while(!in.ready()) {}
        return in.readLine();
    }

}
```

net/TcpServer.java

```
package net;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class UdpListener extends Thread {

```

```
    private DatagramSocket socket;
    private UdpServer server;

    public UdpListener(UdpServer server, DatagramSocket socket){
        this.server = server;
        this.socket = socket;
    }

    @Override
    public void run(){
        try {
            byte[] bMsg = new byte[256];
            DatagramPacket pkg = new DatagramPacket(bMsg, bMsg.length);
            socket.receive(pkg);
            server.connect(pkg.getAddress().getHostAddress(), pkg.getPort() + "", pkg);
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

net/UdpListener.java

```
package net;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UdpServer {

    private final Sender sender;
    UdpListener listener;
    private DatagramSocket serverSocket;
    private String address;
    private int port;

    public UdpServer(Sender sender) {
        this.sender = sender;
        clear();
    }

    public final void clear() {
        try {
            int p = (int) (Math.random() * 100 + 1000);
            System.out.println("porta udp: " + p);
            serverSocket = new DatagramSocket(p);
            listener = new UdpListener(this, serverSocket);
            listener.start();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public void connect(String address, String port, DatagramPacket pkg) {
        this.address = address;
        this.port = Integer.parseInt(port);
        if(pkg != null){
            System.out.println(Integer.parseInt(new String(pkg.getData(), 0, pkg.getLength())
                .trim()));
```

```
38              sender.makePlay(2, Integer.parseInt(new String(pkg.getData(), 0, pkg.getLength())
                    .trim()));
39          }
40      }
41
42      public void write(String data) {
43
44          try {
45              InetAddress addr = InetAddress.getByName(this.address);
46              String s = data;
47              byte[] bMsg = s.getBytes();
48              DatagramPacket pkg = new DatagramPacket(bMsg, bMsg.length, addr, this.port);
49              serverSocket.send(pkg);
50          } catch (Exception ex) {
51              ex.printStackTrace();
52          }
53      }
54
55      public String read() throws IOException {
56          byte[] bMsg = new byte[256];
57          DatagramPacket pkg = new DatagramPacket(bMsg, bMsg.length);
58          serverSocket.receive(pkg);
59          return pkg.toString();
60      }
61
62  }
```

net/UdpServer.java