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 4 (quatro) pacotes: "SDTP1", "gui", "ctrl" e "net". Esses pacotes buscam organizar a estrutura do programa adaptando o renomado modelo MVC.

No pacote "SDTP1" está a classe SDTP1, responsável pelo método principal do programa inicializando a interface gráfica do programa.

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 exibido no início da aplicação, 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.

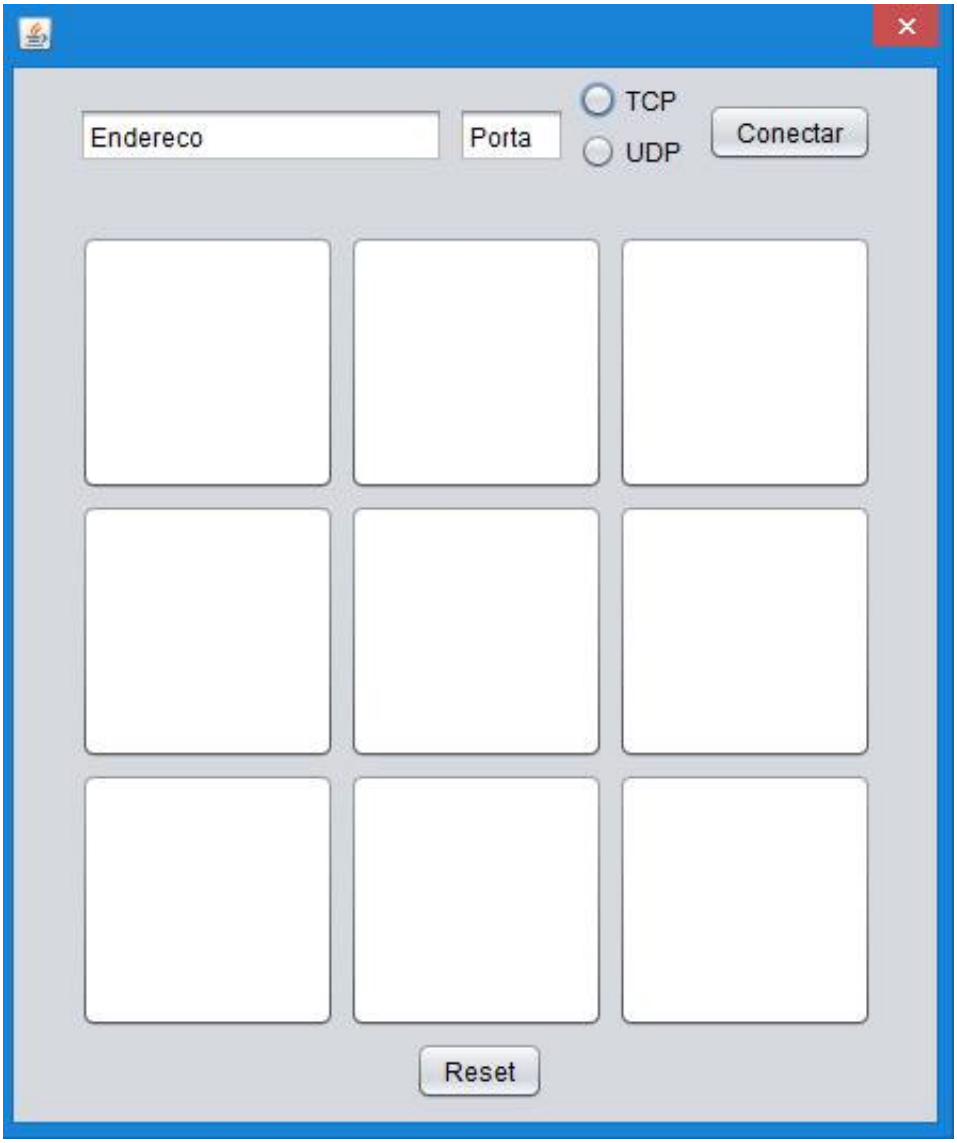

Figura 1: Interface do software

4 Análise

Houveram alguns contratempos durante o desenvolvimento, se destacando a gerência do uso da rede juntamente com a sincronia interna e entre as aplicações.

Pelo fato de a rede *peer-to-peer* utilizada ser de apenas 2 (dois) integrantes a gerência dos hosts não foi algo que se demonstrasse extremamente complexo. A maior dificuldade encontrada foi como gerenciar o uso da rede, fazendo com que cada jogador só pudesse a utilizar quando fosse sua vez, sincronizando as ações das aplicações.

A solução para isso foi utilizando a lógica interna do jogo definir quem deveria jogar e impedindo que o outro cliente tivesse acesso à camada de rede do programa, não sendo necessário então a implementação de um protocolo para a comunicação entre elas. Porém pode se notar que caso não houvessem regras tão claras e simples a sincronia entre as instâncias seria uma tarefa muito mais árdua.

Também foram encontradas dificuldades na sincronia interna da aplicação, devido a necessidade de estar sempre escutando por conexões ou dados na rede e, simultaneamente, por comandos do usuário intermediados pela interface gráfica.

Com o uso de diversas threads foi possível realizar isso de forma não bloqueante para o usuário e que não fossem perdidos dados enviados pela rede. Como foram necessárias poucas threads de vida curta e com tarefa bem específica, os problemas de sincronismo foram solucionados de maneira relativamente simples, sendo o grande custo o tempo de desenvolvimente desprendido para teste e detecção dos problemas durante a produção do software.

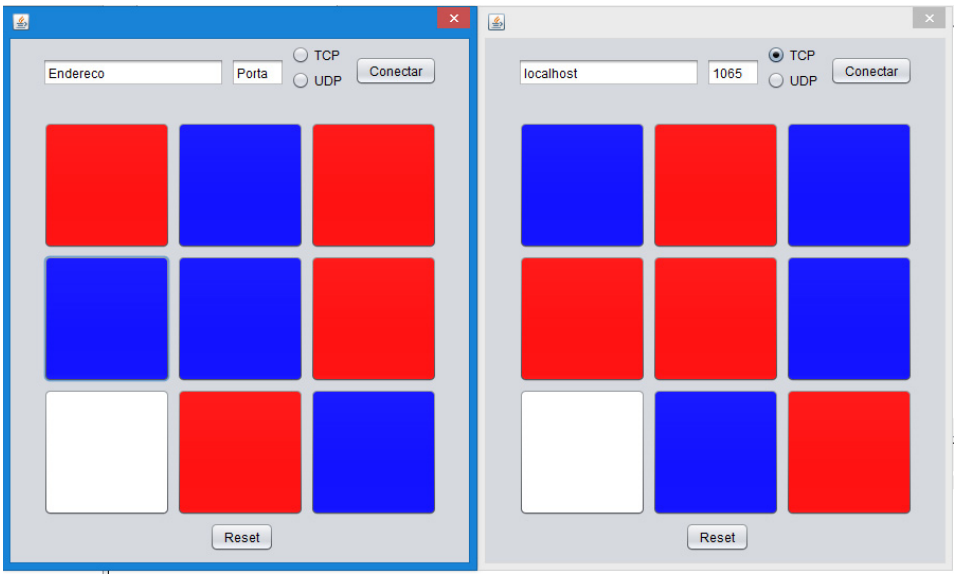

Figura 2: 2 instâncias jogando localmente

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-Fonte

6.1 Pacote "SDTP1"

```
1  package SDTP1;
2
3  import gui.Screen;
4  public class SDTP1 {
5
6      public static void main(String args[]) {
7          /* Set the Nimbus look and feel */
8          //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
               ">
9          /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
               and feel.
10          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/
                plaf.html
11          */
12         try {
13             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
                   getInstalledLookAndFeels()) {
14                 if ("Nimbus".equals(info.getName())) {
15                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
16                     break;
17                 }
18             }
19         } catch (ClassNotFoundException ex) {
20         } catch (InstantiationException ex) {
21         } catch (IllegalAccessException ex) {
22         } catch (javax.swing.UnsupportedLookAndFeelException ex) {}
23         //</editor-fold>
24
25         /* Create and display the dialog */
26         java.awt.EventQueue.invokeLater(new Runnable() {
27             public void run() {
28                 Screen dialog = new Screen(new javax.swing.JFrame(), true);
29                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
30                     @Override
31                     public void windowClosing(java.awt.event.WindowEvent e) {
32                         System.exit(0);
33                     }
34                 });
35                 dialog.setVisible(true);
36             }
37         });
38     }
39 }
```

SDTP1/SDTP1.java

6.2 Pacote “gui”

```
package gui;

import ctrl.TicTacToe;
import java.awt.Color;
import javax.swing.JOptionPane;

public class Screen extends javax.swing.JDialog {

    private final TicTacToe game;
    private final Color[] playerColor;

    /**
     * Creates new form Screen
     * @param parent
     * @param modal
     */

    private boolean isTcp;

    public Screen(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        bgTipo.add(rdTCP);
        bgTipo.add(rdUDP);
        isTcp = true;
        playerColor = new Color[3];
        playerColor[0] = new Color(255, 255, 255);
        playerColor[1] = new Color(0, 0, 255);
        playerColor[2] = new Color(255, 0, 0);
        game = new TicTacToe(this);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        bgTipo = new javax.swing.ButtonGroup();
        jPanel1 = new javax.swing.JPanel();
        txtAddress = new javax.swing.JTextField();
        rdTCP = new javax.swing.JRadioButton();
        rdUDP = new javax.swing.JRadioButton();
        btnConnect = new javax.swing.JButton();
        btn2 = new javax.swing.JButton();
        btn3 = new javax.swing.JButton();
        btn1 = new javax.swing.JButton();
        btn4 = new javax.swing.JButton();
        btn5 = new javax.swing.JButton();
        btn6 = new javax.swing.JButton();
        btn7 = new javax.swing.JButton();
        btn9 = new javax.swing.JButton();
        btn8 = new javax.swing.JButton();
        btnReset = new javax.swing.JButton();
        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))
```

```
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addComponent(btn7, javax.swing.GroupLayout.PREFERRED_SIZE,
                                120, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                RELATED)
                            .addComponent(btn8, javax.swing.GroupLayout.PREFERRED_SIZE,
                                120, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                RELATED)
                            .addComponent(btn9, javax.swing.GroupLayout.PREFERRED_SIZE,
                                120, javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(30, 30, 30)
                    .addComponent(txtAddress, javax.swing.GroupLayout.PREFERRED_SIZE,
                        172, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(txtPort)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.LEADING, false)
                        .addComponent(rdUDP, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                            swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(rdTCP, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                            swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED
                        )
                    .addComponent(btnConnect)))
            .addContainerGap(30, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.
            createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)
            .addComponent(btnReset)
            .addGap(184, 184, 184))
    );

    jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component
        [] {btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9});

    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
                .LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addComponent(rdTCP)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                RELATED)
                            .addComponent(rdUDP))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(10, 10, 10)
                            .addComponent(btnConnect))))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(18, 18, 18)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.BASELINE)
                        .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         // Variables declaration - do not modify//GEN-BEGIN:variables
339         private javax.swing.ButtonGroup bgTipo;
340         private javax.swing.JButton btn1;
341         private javax.swing.JButton btn2;
342         private javax.swing.JButton btn3;
343         private javax.swing.JButton btn4;
344         private javax.swing.JButton btn5;
345         private javax.swing.JButton btn6;
346         private javax.swing.JButton btn7;
347         private javax.swing.JButton btn8;
348         private javax.swing.JButton btn9;
349         private javax.swing.JButton btnConnect;
350         private javax.swing.JButton btnReset;
351         private javax.swing.JPanel jPanel1;
352         private javax.swing.JRadioButton rdTCP;
353         private javax.swing.JRadioButton rdUDP;
354         private javax.swing.JTextField txtAddress;
355         private javax.swing.JTextField txtPort;
356         // End of variables declaration//GEN-END:variables
357 }
```

gui/Screen.java

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

```
55
56      public void connect(String address, String port, boolean isTcp) {
57          sender.connect(address, port, isTcp);
58          initPlay = 2;
59          lastPlay = 2;
60      }
61
62      public void connected(){
63          initPlay = 1;
64          lastPlay = 1;
65          sender.readPlay();
66          screen.createDialog("Jogador conectado", "");
67      }
68
69      public boolean isOver() {
70          if(winner() != 0){
71              screen.createDialog("Jogador " + winner() + " venceu!", "Resultado");
72              reset();
73              return true;
74          }
75
76          for(int i = 0; i < 3; i++)
77              for(int j = 0; j < 3; j++)
78                  if(grid[i][j] == 0)
79                      return false;
80
81          screen.createDialog("Deu velha!", "Resultado");
82          reset();
83          return true;
84      }
85
86      public void setLastPlay(int lastPlay) {
87          this.lastPlay = lastPlay;
88      }
89
90      public void createDialog(String message, String title) {
91          screen.createDialog(message, title);
92      }
93  }
```

ctrl/TicTacToe.java

6.4 Pacote “net”

```java
package net;

import ctrl.TicTacToe;

public class Sender {

    private final TicTacToe game;
    private TcpServer tcp;
    private int tcpPort;
    private UdpServer udp;
    private int udpPort;
    private boolean isTcp;
    private boolean connected;

    public Sender(TicTacToe t){
        game = t;
        tcpPort = (int) (Math.random() * 100 + 1000);
        System.out.println("porta tcp: " + tcpPort);
        udpPort = (int) (Math.random() * 100 + 1000);
        System.out.println("porta udp: " + udpPort);
        game.createDialog("TCP: " + tcpPort + "\nUDP: " + udpPort, "Escutando nas portas");
        tcp = new TcpServer(this, tcpPort);
        udp = new UdpServer(this, udpPort);
    }

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

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

        connected = true;
    }

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

    public void readPlay() {
        PlayReader reader = new PlayReader(tcp, udp, game, isTcp);
        reader.start();
    }

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

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

```
59          game.makePlay(2, pos);
60      }
61
62  }
```

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 int listeningPort;
16      private BufferedReader in;
17      private BufferedWriter out;
18
19      public TcpServer(Sender sender, int listeningPort) {
20          this.sender = sender;
21          this.listeningPort = listeningPort;
22          clear();
23      }
```

```
24
25      public final void clear(){
26          try {
27              tcp = new TcpListener(this, listeningPort);
28              tcp.start();
29          } catch (Exception ex) {
30              ex.printStackTrace();
31          }
32          socket = null;
33      }
34
35      public void setBuffers(Socket socket){
36          this.socket = socket;
37          try {
38              in =  new BufferedReader(new InputStreamReader(socket.getInputStream()));
39              out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
40          } catch (IOException ex) {
41              ex.printStackTrace();
42          }
43      }
44
45      public void connect(String address, String port) {
46          try {
47              socket = new Socket(address, Integer.parseInt(port));
48          } catch (IOException ex) {
49              ex.printStackTrace();
50          }
51          tcp = null;
52          setBuffers(socket);
53      }
54
55      public void connected(Socket socket) {
56          setBuffers(socket);
57          tcp = null;
58          sender.connected(true);
59      }
60
61      public void write(String data){
62          try {
63              out.write(data);
64              out.flush();
65          } catch (IOException ex) {
66              ex.printStackTrace();
67          }
68      }
69
70      public String read() throws IOException {
71          return (in.read() - 48) + "";
72      }
73
74  }
```

net/TcpServer.java

```
1   package net;
2
3   import java.io.IOException;
4   import java.net.DatagramPacket;
5   import java.net.DatagramSocket;
6
```

```java
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

```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;
    private int listeningPort;

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

    public final void clear() {
        try {
            serverSocket = new DatagramSocket(this.listeningPort);
            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);
```

```
36          if(pkg != null){
37              sender.connected(false);
38              sender.makePlay(2, Integer.parseInt(new String(pkg.getData(), 0, pkg.getLength())
                    .trim()));
39          }
40          listener = null;
41      }
42
43      public void write(String data) {
44
45          try {
46              InetAddress addr = InetAddress.getByName(this.address);
47              String s = data;
48              byte[] bMsg = s.getBytes();
49              DatagramPacket pkg = new DatagramPacket(bMsg, bMsg.length, addr, this.port);
50              serverSocket.send(pkg);
51          } catch (Exception ex) {
52              ex.printStackTrace();
53          }
54      }
55
56      public String read() throws IOException {
57          byte[] bMsg = new byte[256];
58          String s = "";
59          while ("".equals(s)) {
60              DatagramPacket pkg = new DatagramPacket(bMsg, bMsg.length);
61              serverSocket.receive(pkg);
62              s = new String(pkg.getData());
63          }
64          return s;
65      }
66
67  }
```

net/UdpServer.java

```
1   package net;
2
3   import ctrl.TicTacToe;
4   import java.io.IOException;
5   import java.util.OptionalInt;
6   import java.util.stream.IntStream;
7
8   public class PlayReader extends Thread {
9
10      private final TcpServer tcp;
11      private final UdpServer udp;
12      private final TicTacToe game;
13      private final boolean isTcp;
14
15      public PlayReader(TcpServer tcp, UdpServer udp, TicTacToe game, boolean isTcp) {
16          this.tcp = tcp;
17          this.udp = udp;
18          this.game = game;
19          this.isTcp = isTcp;
20      }
21
22      @Override
23      public void run(){
24          String read = "";
```

```
25          if(isTcp)
26              while("".equals(read) || read == null){
27                  try { read = tcp.read(); }
28                  catch (IOException ex) { ex.printStackTrace(); }
29              }
30          else
31              while("".equals(read) || read == null){
32                  try { read = udp.read(); }
33                  catch (IOException ex) { ex.printStackTrace(); }
34              }
35          int p;
36          try { p = Integer.parseInt(read); }
37          catch (NumberFormatException e) {
38              //Fazendo o que o Java por algum motivo faz errado as vezes
39              p = read.chars().findFirst().getAsInt() - 48;
40          }
41          game.makePlay(2, p);
42      }
43
44
45  }
```

net/PlayReader.java