

85pt



CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS VII - UNIDADE TIMÓTEO - Engenharia da Computação

Trabalho Prático: Soquetes
André Malta, Elias Luiz, Nil Martins

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 endereço de rede de um outro usuário, 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.

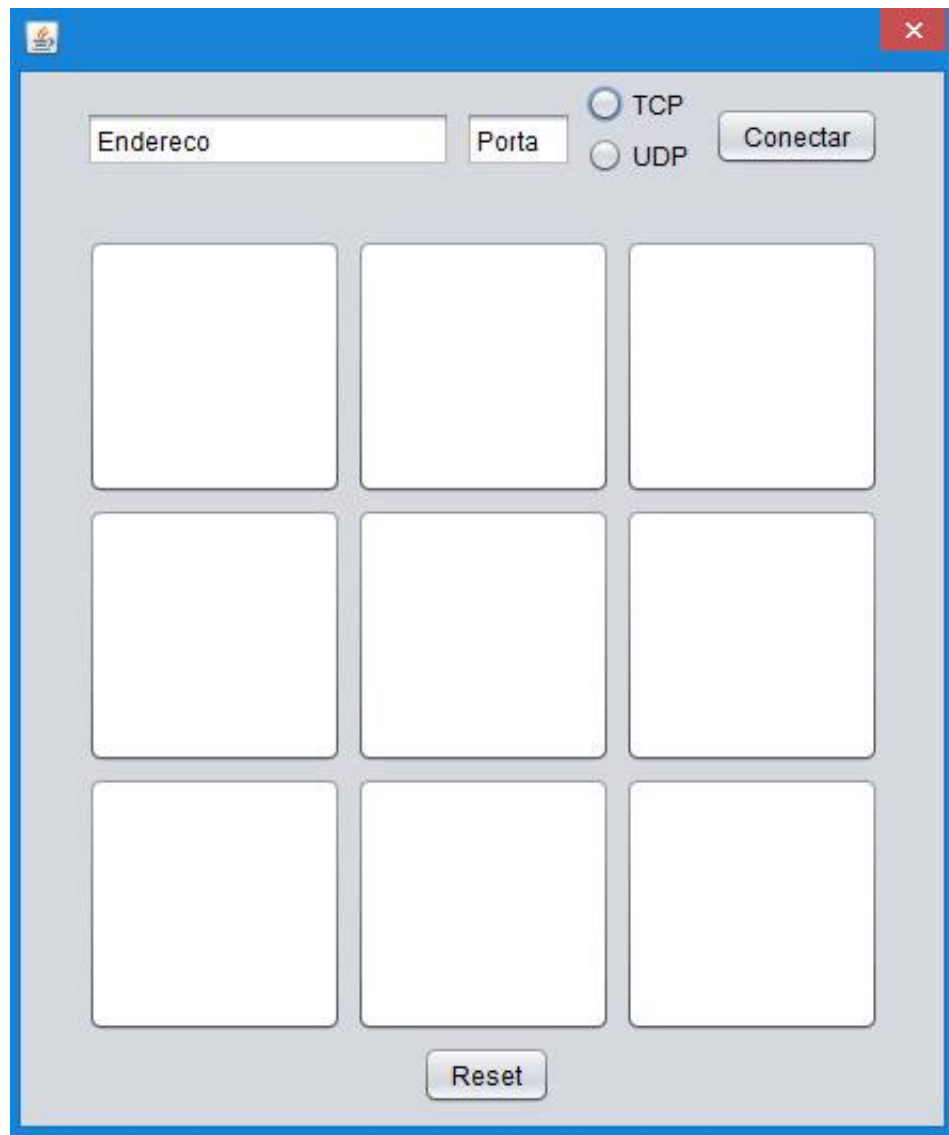


Figura 1: Interface do software

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 desenvolvimento despendido 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

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  }
```

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



```
156         .addGroup(jPanel1Layout.createParallelGroup(GroupLayout.Alignment.LEADING)  
157             .addGroup(jPanel1Layout.createSequentialGroup()  
158                 .addComponent(btn1, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)  
159                 .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
160                 .addComponent(btn2, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)  
161                 .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
162                 .addComponent(btn3, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE))  
163             .addGroup(jPanel1Layout.createSequentialGroup()  
164                 .addComponent(btn4, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)  
165                 .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
166                 .addComponent(btn5, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)  
167                 .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
168                 .addComponent(btn6, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE))  
169             .addGroup(jPanel1Layout.createSequentialGroup()  
170                 .addComponent(btn7, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)  
171                 .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
172                 .addComponent(btn8, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)  
173                 .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
174                 .addComponent(btn9, GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE)))  
175         .addGroup(jPanel1Layout.createSequentialGroup()  
176             .addGap(30, 30, 30)  
177             .addComponent(txtAddress, GroupLayout.PREFERRED_SIZE, 172, GroupLayout.PREFERRED_SIZE)  
178             .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
179             .addComponent(txtPort)  
180             .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)  
181             .addGroup(jPanel1Layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)  
182                 .addComponent(rdUDP, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)  
183                 .addComponent(rdTCP, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))  
184             .addPreferredGap(GroupLayoutStyle.ComponentPlacement.UNRELATED)  
185             .addComponent(btnConnect))  
186         .addContainerGap(30, Short.MAX_VALUE)  
187         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  
188             .createSequentialGroup()  
189             .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
199         .LEADING)
200         .addGroup(jPanel1Layout.createSequentialGroup())
201         .addContainerGap()
202         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
203             Alignment.LEADING)
204             .addGroup(jPanel1Layout.createSequentialGroup())
205             .addComponent(rdTCP)
206             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
207                 RELATED)
208             .addComponent(rdUDP))
209             .addGroup(jPanel1Layout.createSequentialGroup())
210             .addGap(10, 10, 10)
211             .addComponent(btnConnect))))
212     .addGroup(jPanel1Layout.createSequentialGroup())
213     .addGap(18, 18, 18)
214     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
215         Alignment.BASELINE)
216         .addComponent(txtAddress, javax.swing.GroupLayout.PREFERRED_SIZE,
217             27, javax.swing.GroupLayout.PREFERRED_SIZE)
218         .addComponent(txtPort, javax.swing.GroupLayout.PREFERRED_SIZE,
219             27, javax.swing.GroupLayout.PREFERRED_SIZE)))
220     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 30,
221         Short.MAX_VALUE)
222     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
223         .BASELINE)
224         .addComponent(btn2, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
225             swing.GroupLayout.PREFERRED_SIZE)
226         .addComponent(btn3, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
227             swing.GroupLayout.PREFERRED_SIZE)
228         .addComponent(btn1, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
229             swing.GroupLayout.PREFERRED_SIZE)
230         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
231         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
232             .BASELINE)
233             .addComponent(btn5, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
234                 swing.GroupLayout.PREFERRED_SIZE)
235             .addComponent(btn6, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
236                 swing.GroupLayout.PREFERRED_SIZE)
237             .addComponent(btn4, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
238                 swing.GroupLayout.PREFERRED_SIZE)
239             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
240             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
241                 .BASELINE)
242                 .addComponent(btn8, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
243                     swing.GroupLayout.PREFERRED_SIZE)
244                 .addComponent(btn9, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
245                     swing.GroupLayout.PREFERRED_SIZE)
246                 .addComponent(btn7, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.
247                     swing.GroupLayout.PREFERRED_SIZE)
248                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
249                 .addComponent(btnReset)
250                 .addGap(10, 10, 10))
251             .addComponent(btnReset)
252             .addGap(10, 10, 10))
253     );
254
255 jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
256     {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 :
287     event_rdTCPActionPerformed
288     isTcp = true ;
289 } //GEN-LAST : event_rdTCPActionPerformed
290
291 private void rdUDPActionPerformed ( java . awt . event . ActionEvent evt ) { //GEN-FIRST :
292     event_rdUDPActionPerformed
293     isTcp = false ;
294 } //GEN-LAST : event_rdUDPActionPerformed
295
296 private void btnResetActionPerformed ( java . awt . event . ActionEvent evt ) { //GEN-FIRST :
297     event_btnResetActionPerformed
298     game . reset () ;
299 } //GEN-LAST : event_btnResetActionPerformed
300
301 private void btnConnectActionPerformed ( java . awt . event . ActionEvent evt ) { //GEN-FIRST :
302     event_btnConnectActionPerformed
303     game . connect ( txtAddress . getText () , txtPort . getText () , isTcp ) ;
304 } //GEN-LAST : event_btnConnectActionPerformed
305
306 public void changeButtonColor ( int player , int button ) {
307     switch ( button ) {
308         case 1 :
309             btn1 . setBackground ( playerColor [ player ] ) ;
310             break ;
311         case 2 :
312             btn2 . setBackground ( playerColor [ player ] ) ;
313             break ;
314         case 3 :
315             btn3 . setBackground ( playerColor [ player ] ) ;
316             break ;
317         case 4 :
318             btn4 . setBackground ( playerColor [ player ] ) ;
319             break ;
320         case 5 :
321             btn5 . setBackground ( playerColor [ player ] ) ;
322             break ;
323         case 6 :
324             btn6 . setBackground ( playerColor [ player ] ) ;
325             break ;
326         case 7 :
327             btn7 . setBackground ( playerColor [ player ] ) ;
328             break ;
329         case 8 :
330             btn8 . setBackground ( playerColor [ player ] ) ;
331             break ;
332         case 9 :
333             btn9 . setBackground ( playerColor [ player ] ) ;
334     }
335 }
336
337 public void createDialog ( String message , String title ) {
338     JOptionPane . showMessageDialog ( jPanel1 , message , title , JOptionPane .
339         INFORMATION_MESSAGE ) ;
340 }
341
342 // Variables declaration — do not modify //GEN-BEGIN : variables
343 private javax . swing . ButtonGroup bgTipo ;
344 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

6.2 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 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                ||
34                (grid[0][2] == grid[1][1] && grid[0][2] == grid[2][0] && grid[0][2] == player)
35            )
36                return player;
37            for(int i = 0; i < 3; i++)
38                if((grid[i][0] == grid[i][1] && grid[i][0] == grid[i][2] && grid[i][0] ==
39                    player) ||
40                    (grid[0][i] == grid[1][i] && grid[0][i] == grid[2][i] && grid[0][i] ==
41                        player))
42                    return player;
43        }
44        return 0;
45    }
46
47    public void makePlay(int player, int pos) {
48        if(grid[pos / 3][pos % 3] == 0 && !isOver() && lastPlay != player){
49            lastPlay = player;
50            grid[pos / 3][pos % 3] = player;
51            screen.changeButtonColor(player, pos+1);
52            if(player == 1){
53                sender.writePlay(pos);
54                sender.readPlay();
55            }
56        }
57        isOver();
58    }
59 }
```

```
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

6.3 Pacote “net”

```
1 package net;
2
3 import ctrl.TicTacToe;
4
5 public class Sender {
6
7     private final TicTacToe game;
8     private TcpServer tcp;
9     private int tcpPort;
10    private UdpServer udp;
11    private int udpPort;
12    private boolean isTcp;
13    private boolean connected;
14
15    public Sender(TicTacToe t){
16        game = t;
17        tcpPort = (int) (Math.random() * 100 + 1000);
18        System.out.println("porta tcp: " + tcpPort);
19        udpPort = (int) (Math.random() * 100 + 1000);
20        System.out.println("porta udp: " + udpPort);
21        game.createDialog("TCP: " + tcpPort + "\nUDP: " + udpPort, "Escutando nas portas");
22        tcp = new TcpServer(this, tcpPort);
23        udp = new UdpServer(this, udpPort);
24    }
25
26    public void connect(String address, String port, boolean isTcp) {
27        this.isTcp = isTcp;
28
29        if(isTcp)
30            tcp.connect(address, port);
31        else
32            udp.connect(address, port, null);
33
34        connected = true;
35    }
36
37    public void writePlay(int pos) {
38        if(isTcp)
39            tcp.write(pos + "");
40        else
41            udp.write(pos + "");
42    }
43
44    public void readPlay() {
45        PlayReader reader = new PlayReader(tcp, udp, game, isTcp);
46        reader.start();
47    }
48
49    public void connected(boolean isTcp) {
50        this.isTcp = isTcp;
51        if(!connected){
52            connected = true;
53            game.connected();
54        }
55    }
56
57    public void makePlay(int player, int pos){
58        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
```

```
7 public class UdpListener extends Thread {
8
9     private DatagramSocket socket;
10    private UdpServer server;
11
12    public UdpListener(UdpServer server, DatagramSocket socket){
13        this.server = server;
14        this.socket = socket;
15    }
16
17    @Override
18    public void run(){
19        try {
20            byte[] bMsg = new byte[256];
21            DatagramPacket pkg = new DatagramPacket(bMsg, bMsg.length);
22            socket.receive(pkg);
23            server.connect(pkg.getAddress().getHostAddress(), pkg.getPort() + "", pkg);
24        } catch (IOException ex) {
25            ex.printStackTrace();
26        }
27    }
28 }
```

net/UdpListener.java

```
1 package net;
2
3 import java.io.IOException;
4 import java.net.DatagramPacket;
5 import java.net.DatagramSocket;
6 import java.net.InetAddress;
7
8 public class UdpServer {
9
10    private final Sender sender;
11    UdpListener listener;
12    private DatagramSocket serverSocket;
13    private String address;
14    private int port;
15    private int listeningPort;
16
17    public UdpServer(Sender sender, int listeningPort) {
18        this.sender = sender;
19        this.listeningPort = listeningPort;
20        clear();
21    }
22
23    public final void clear() {
24        try {
25            serverSocket = new DatagramSocket(this.listeningPort);
26            listener = new UdpListener(this, serverSocket);
27            listener.start();
28        } catch (IOException ex) {
29            ex.printStackTrace();
30        }
31    }
32
33    public void connect(String address, String port, DatagramPacket pkg) {
34        this.address = address;
35        this.port = Integer.parseInt(port);
36    }
37 }
```

```
36         if(pkg != null){
37             sender.connected(false);
38             sender.makePlay(2, Integer.parseInt(new String(pkg.getData(), 0, pkg.getLength())
39                 .trim()));
40         }
41         listener = null;
42     }
43     public void write(String data) {
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
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
