LAPORAN PEMROGRAMAN JARINGAN

UJIAN AKHIR SEMESTER

Oleh:

KINANTI PERMATA PUTRI NIM. 1841720022

TI-3D

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

DECEMBER 2020

Source Code Tic-Tac-Toe:

```
6   package Game;
7
8   import java.awt.BasicStroke;
9   import java.awt.Color;
10  import java.awt.Dimension;
11  import java.awt.Font;
12  import java.awt.Graphics;
13  import java.awt.Graphics2D;
14  import java.awt.RenderingHints;
15  import java.awt.Toolkit;
16  import java.awt.event.MouseEvent;
17  import java.awt.event.MouseListener;
18  import java.awt.image.BufferedImage;
19  import java.io.DataInputStream;
20  import java.io.DataOutputStream;
21  import java.io.IOException;
22  import java.net.InetAddress;
23  import java.net.ServerSocket;
24  import java.net.Socket;
25  import java.util.Scanner;
26
27  import javax.imageio.ImageIO;
28  import javax.swing.JFrame;
29  import javax.swing.JPanel;
30
31  public class TicTacToeGame implements Runnable {
32
33      private String ip = "localhost";
34      private int port = 22222;
        private Scanner scanner = new Scanner(System.in);
        private JFrame frame;
37      private final int WIDTH = 506;
38      private final int HEIGHT = 527;
        private Thread thread;
40
        private Painter painter;
42      private Socket socket;
43      private DataOutputStream dos;
44      private DataInputStream dis;
45
```

```
46      private ServerSocket serverSocket;
47
48      private BufferedImage board;
49      private BufferedImage redX;
50      private BufferedImage blueX;
51      private BufferedImage redO;
52      private BufferedImage blueO;
53
54      private String[] spaces = new String[9];
55
56      private boolean yourTurn = false;
57      private boolean circle = true;
58      private boolean accepted = false;
59      private boolean unableToCommunicateWithOpponent = false;
60      private boolean won = false;
61      private boolean enemyWon = false;
62      private boolean tie = false;
63
64      private int lengthOfSpace = 160;
65      private int errors = 0;
66      private int firstSpot = -1;
67      private int secondSpot = -1;
68
69      private Font font = new Font("Arial", Font.BOLD, 32);
70      private Font smallerFont = new Font("Arial", Font.BOLD, 20);
71      private Font largerFont = new Font("Arial", Font.BOLD, 50);
72
73      private String waitingString = "Menunggu player lain";
74      private String unableToCommunicateWithOpponentString = "Tidak bisa menghubungkan ke player lain";
75      private String wonString = "Kamu Menang!";
76      private String enemyWonString = "Kamu Kalah!";
77      private String tieString = "Game Seri";
78
79      private int[][] wins = new int[][] { { 0, 1, 2 }, { 3, 4, 5 }, { 6, 7, 8 }, { 0, 3, 6 }, { 1, 4, 7 }, { 2, 5, 8 }, { 0, 4, 8 }, { 2, 4, 6 } };
80
81      /**
82       * <pre>
83       * 0, 1, 2
84       * 3, 4, 5
85       * 6, 7, 8
86       * </pre>
87       */
88
```

```
89      public TicTacToeGame() {
90          System.out.println("Masukkan IP: ");
91          ip = scanner.nextLine();
92          System.out.println("Masukkan port: ");
93          port = scanner.nextInt();
94
95          while (port < 1 || port > 65535) {
96              System.out.println("port yang anda masukkan tidak valid, silakan masukkan port lain: ");
97              port = scanner.nextInt();
98          }
99
100         loadImages();
101
102         painter = new Painter();
103         painter.setPreferredSize(new Dimension(WIDTH, HEIGHT));
104
105         if (!connect()) initializeServer();
106
107         frame = new JFrame();
108         frame.setTitle("Tic Tac Toe");
109         frame.setContentPane(painter);
110         frame.setSize(WIDTH, HEIGHT);
111         frame.setLocationRelativeTo(null);
112         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
113         frame.setResizable(false);
114         frame.setVisible(true);
115
116         thread = new Thread(this, "TicTacToe");
            thread.start();
118     }
119
        public void run() {
121         while (true) {
122             tick();
123             painter.repaint();
124
125             if (!circle && !accepted) {
126                 listenForServerRequest();
127             }
128         }
129     }
130
```

```
131     private void render(Graphics g) {
132         g.drawImage(board, 0, 0, null);
133 
134         if (unableToCommunicateWithOpponent) {
135             g.setColor(Color.RED);
136             g.setFont(smallerFont);
137             Graphics2D g2 = (Graphics2D) g;
138             g2.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING, RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
139             int stringWidth = g2.getFontMetrics().stringWidth(unableToCommunicateWithOpponentString);
140             g.drawString(unableToCommunicateWithOpponentString, WIDTH / 2 - stringWidth / 2, HEIGHT / 2);
141             return;
142         }
143         if (accepted) {
144             for (int i = 0; i < spaces.length; i++) {
145                 if (spaces[i] != null) {
146                     if (spaces[i].equals("X")) {
147                         if (circle) {
148                             g.drawImage(redX, (i % 3) * lengthOfSpace + 10 * (i % 3), (int) (i / 3) * lengthOfSpace + 10 * (int) (i / 3), null);
149                         } else {
150                             g.drawImage(blueX, (i % 3) * lengthOfSpace + 10 * (i % 3), (int) (i / 3) * lengthOfSpace + 10 * (int) (i / 3), null);
151                         }
152                     } else if (spaces[i].equals("O")) {
153                         if (circle) {
154                             g.drawImage(blueO, (i % 3) * lengthOfSpace + 10 * (i % 3), (int) (i / 3) * lengthOfSpace + 10 * (int) (i / 3), null);
155                         } else {
156                             g.drawImage(redO, (i % 3) * lengthOfSpace + 10 * (i % 3), (int) (i / 3) * lengthOfSpace + 10 * (int) (i / 3), null);
157                         }
158                     }
159                 }
160             }
161             if (won || enemyWon) {
162                 Graphics2D g2 = (Graphics2D) g;
163                 g2.setStroke(new BasicStroke(10));
164                 g.setColor(Color.BLACK);
165                 g.drawLine(firstSpot % 3 * lengthOfSpace + 10 * firstSpot % 3 + lengthOfSpace / 2, (int) (firstSpot / 3) * lengthOfSpace + 10 * (int) (firstSpot / 3) + l
166                 g.setColor(Color.RED);
167                 g.setFont(largerFont);
168 
169                 if (won) {
170                     int stringWidth = g2.getFontMetrics().stringWidth(wonString);
171                     g.drawString(wonString, WIDTH / 2 - stringWidth / 2, HEIGHT / 2);
172                 } else if (enemyWon) {
173                     int stringWidth = g2.getFontMetrics().stringWidth(enemyWonString);
174                     g.drawString(enemyWonString, WIDTH / 2 - stringWidth / 2, HEIGHT / 2);
175                 }
```

```
176                 }
177                 if (tie) {
178                     Graphics2D g2 = (Graphics2D) g;
179                     g.setColor(Color.BLACK);
180                     g.setFont(largerFont);
181                     int stringWidth = g2.getFontMetrics().stringWidth(tieString);
182                     g.drawString(tieString, WIDTH / 2 - stringWidth / 2, HEIGHT / 2);
183                 }
184             } else {
185                 g.setColor(Color.RED);
186                 g.setFont(font);
187                 Graphics2D g2 = (Graphics2D) g;
188                 g2.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING, RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
189                 int stringWidth = g2.getFontMetrics().stringWidth(waitingString);
190                 g.drawString(waitingString, WIDTH / 2 - stringWidth / 2, HEIGHT / 2);
191             }
192         }
193 
194         private void tick() {
195             if (errors >= 10) unableToCommunicateWithOpponent = true;
196 
197             if (!yourTurn && !unableToCommunicateWithOpponent) {
198                 try {
199                     int space = dis.readInt();
200                     if (circle) spaces[space] = "X";
201                     else spaces[space] = "O";
202                     checkForEnemyWin();
203                     checkForTie();
204                     yourTurn = true;
205                 } catch (IOException e) {
                        e.printStackTrace();
207                     errors++;
208                 }
209             }
210         }
211 
```

```
212     private void checkForWin() {
            for (int i = 0; i < wins.length; i++) {
214             if (circle) {
                    if (spaces[wins[i][0]] == "O" && spaces[wins[i][1]] == "O" && spaces[wins[i][2]] == "O") {
216                     firstSpot = wins[i][0];
217                     secondSpot = wins[i][2];
218                     won = true;
219                 }
220             } else {
                    if (spaces[wins[i][0]] == "X" && spaces[wins[i][1]] == "X" && spaces[wins[i][2]] == "X") {
222                     firstSpot = wins[i][0];
223                     secondSpot = wins[i][2];
224                     won = true;
225                 }
226             }
227         }
228     }
229
230
231     private void checkForEnemyWin() {
            for (int i = 0; i < wins.length; i++) {
233             if (circle) {
                    if (spaces[wins[i][0]] == "X" && spaces[wins[i][1]] == "X" && spaces[wins[i][2]] == "X") {
235                     firstSpot = wins[i][0];
236                     secondSpot = wins[i][2];
237                     enemyWon = true;
238                 }
239             } else {
                    if (spaces[wins[i][0]] == "O" && spaces[wins[i][1]] == "O" && spaces[wins[i][2]] == "O") {
241                     firstSpot = wins[i][0];
242                     secondSpot = wins[i][2];
243                     enemyWon = true;
244                 }
245             }
246         }
247     }
248
249
```

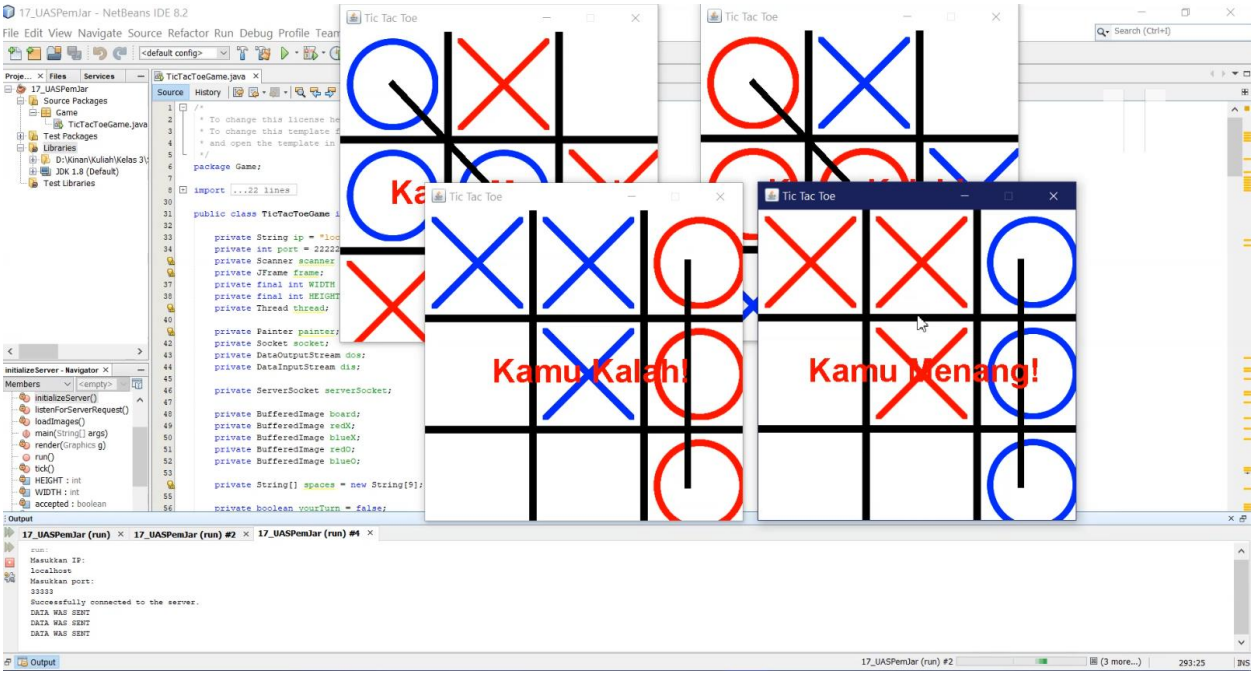
```
250     private void checkForTie() {
            for (int i = 0; i < spaces.length; i++) {
252             if (spaces[i] == null) {
253                 return;
254             }
255         }
256         tie = true;
257     }
258
259     private void listenForServerRequest() {
            Socket socket = null;
261         try {
262             socket = serverSocket.accept();
263             dos = new DataOutputStream(socket.getOutputStream());
264             dis = new DataInputStream(socket.getInputStream());
265             accepted = true;
266             System.out.println("CLIENT HAS REQUESTED TO JOIN, AND WE HAVE ACCEPTED");
267         } catch (IOException e) {
                e.printStackTrace();
269         }
270     }
271
272     private boolean connect() {
273
274         try {
275             socket = new Socket(ip, port);
276             dos = new DataOutputStream(socket.getOutputStream());
277             dis = new DataInputStream(socket.getInputStream());
278             accepted = true;
279         } catch (IOException e) {
280             System.out.println("Unable to connect to the address: " + ip + ":" + port + " | Starting a server");
281             return false;
282         }
283         System.out.println("Successfully connected to the server.");
284         return true;
285     }
286
```

```
287     private void initializeServer() {
288         try {
289             serverSocket = new ServerSocket(port, 8, InetAddress.getByName(ip));
            } catch (Exception e) {
                e.printStackTrace();
292         }
293         yourTurn = true;
294         circle = false;
295     }
296
297     private void loadImages() {
298
299         try {
300             board = ImageIO.read(getClass().getResourceAsStream("/board.png"));
301             redX = ImageIO.read(getClass().getResourceAsStream("/redX.png"));
302             redO = ImageIO.read(getClass().getResourceAsStream("/redCircle.png"));
303             blueX = ImageIO.read(getClass().getResourceAsStream("/blueX.png"));
304             blueO = ImageIO.read(getClass().getResourceAsStream("/blueCircle.png"));
305         } catch (IOException e) {
                e.printStackTrace();
307         }
308     }
309
310     @SuppressWarnings("unused")
311     public static void main(String[] args) {
312         TicTacToeGame ticTacToe = new TicTacToeGame();
313     }
314
315     private class Painter extends JPanel implements MouseListener {
316         private static final long serialVersionUID = 1L;
317         public Painter() {
                setFocusable(true);
                requestFocus();
                setBackground(Color.WHITE);
                addMouseListener(this);
322         }
323
324         @Override
            public void paintComponent(Graphics g) {
326             super.paintComponent(g);
327             render(g);
328         }
```

```
330         @Override
            public void mouseClicked(MouseEvent e) {
332             if (accepted) {
333                 if (yourTurn && !unableToCommunicateWithOpponent && !won && !enemyWon) {
334                     int x = e.getX() / lengthOfSpace;
335                     int y = e.getY() / lengthOfSpace;
336                     y *= 3;
337                     int position = x + y;
338
339                     if (spaces[position] == null) {
340                         if (!circle) spaces[position] = "X";
341                         else spaces[position] = "O";
342                         yourTurn = false;
343                         repaint();
344                         Toolkit.getDefaultToolkit().sync();
345
346                         try {
347                             dos.writeInt(position);
348                             dos.flush();
349                         } catch (IOException e1) {
350                             errors++;
                            e1.printStackTrace();
352                         }
353
354                         System.out.println("DATA WAS SENT");
355                         checkForWin();
356                         checkForTie();
357
358                     }
359                 }
360             }
361         }
362
363         @Override
            public void mousePressed(MouseEvent e) {
365
366         }
367
368
369         @Override
            public void mouseReleased(MouseEvent e) {
371
372         }
```

```
369         @Override
            public void mouseReleased(MouseEvent e) {
371
372         }
373
374         @Override
            public void mouseEntered(MouseEvent e) {
376
377         }
378
379
380         @Override
            public void mouseExited(MouseEvent e) {
382
383         }
384
385     }
386
387 }
```

Output:


Tic Tac Toe
Kamu Kalah!
Tic Tac Toe
Kamu Menang!
17_UASPemJar - NetBeans IDE 8.2
Masukkan IP:
localhost
Masukkan port:
33333
Successfully connected to the server.
DATA WAS SENT
DATA WAS SENT
DATA WAS SENT