МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №12

Специальность ПО5

Выполнили:
А.А. Игнатюк,
В.В. Крощук,
студенты группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2022 г.

Брест 2022

Цель работы: Освоить приемы разработки оконных клиент-серверных приложений на Java с использованием сокетов.

Вариант 5.

Задание.

Разработать клиент-серверное оконное приложение на Java с использованием сокетов и JavaFX. Можно сделать одну программу с сочетанием функций клиента и сервера либо две отдельных (клиентская часть и серверная часть).

Продемонстрировать работу разработанной программы в сети либо локально (127.0.0.1).

Лабораторную работу разрешается выполнять в команде из 2-х человек.

8) Игра Крестики-нолики. Классическая игра для двух игроков на поле 3x3.

Спецификация ввода: Взаимодействие пользователя с элементами графического интерфейса
Спецификация вывода: Отображение данных в элементах графического интерфейса

Код программы и результаты тестирования:

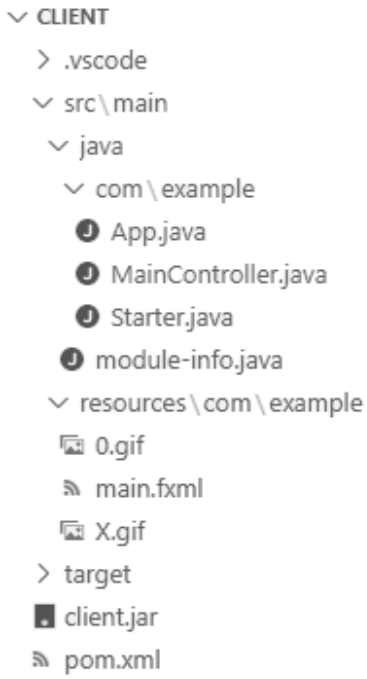

Рисунок 1 - Структура проекта клиента.

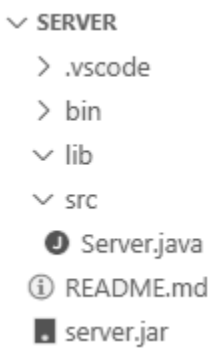

Рисунок 2 - Структура проекта сервера.

Server.java

src > Server.java > ...

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public final class Server {
    Run | Debug
    public final static void main(final String[] args) throws Exception {
        final Integer PORT = 9099;
        ServerSocket acceptor = null;

        while (true) {
            try {
                acceptor = new ServerSocket(PORT);
                System.out.println("Server is Running on port " + Integer.toString(PORT) + '.');

                System.out.println( x: "Waiting for players...");
                Player playerX = new Player(acceptor.accept(), role: "X", opponentRole: "0");
                System.out.println( x: "Player X connected.");
                Player player0 = new Player(acceptor.accept(), role: "0", opponentRole: "X");
                System.out.println( x: "Player 0 connected.");

                playerX.setOpponent(player0);
                player0.setOpponent(playerX);

                Game game = new Game(playerX);

                playerX.setGame(game);
                player0.setGame(game);

                playerX.start();
                player0.start();

                while (!game.isOver() && !game.isWin()) {
                }
            } finally {
                acceptor.close();
            }
        }
    }

    private static class Game {
        private Player current;

        private final Integer BOXES_COLUMNS_COUNT = 3;
        private final Integer BOXES_ROWS_COUNT = 3;
        private String[] board = new String[BOXES_COLUMNS_COUNT * BOXES_ROWS_COUNT];

        public Game(final Player current) {
            this.current = current;
        }

        public final synchronized Boolean move(final Player player, final Integer boxNumber) {
            if (this.current != player || this.board[boxNumber] != null) {
                return false;
            }

            System.out.println("Player " + this.current.getRole() + " move at index " + Integer.toString(boxNumber));

            this.board[boxNumber] = this.current.getRole();

            if (this.isWin()) {
                System.out.println("Player " + this.current.getRole() + " won!");
                System.out.println("Player " + this.current.getOpponent().getRole() + " lost!");

                this.current.reportWin(boxNumber);
                this.current.opponent.reportLose(boxNumber);
                return true;
            }

            if (this.isOver()) {
                System.out.println( x: "Draw!");

                this.current.reportDraw(boxNumber);
                this.current.getOpponent().reportDraw(boxNumber);
                return true;
            }

            this.current.reportMove(boxNumber);

            this.current = this.current.opponent;
            return true;
        }

```

Рисунок 3 - Исходный код файла Server.java.

Продолжение рисунка 3.

```
88          public final Boolean isWin() {
89              return (this.board[0] != null && this.board[0] == this.board[1] && this.board[0] == this.board[2])
90                      || (this.board[3] != null && this.board[3] == this.board[4] && this.board[3] == this.board[5])
91                      || (this.board[6] != null && this.board[6] == this.board[7] && this.board[6] == this.board[8])
92                      || (this.board[0] != null && this.board[0] == this.board[3] && this.board[0] == this.board[6])
93                      || (this.board[1] != null && this.board[1] == this.board[4] && this.board[1] == this.board[7])
94                      || (this.board[2] != null && this.board[2] == this.board[5] && this.board[2] == this.board[8])
95                      || (this.board[0] != null && this.board[0] == this.board[4] && this.board[0] == this.board[8])
96                      || (this.board[2] != null && this.board[2] == this.board[4] && this.board[2] == this.board[6]);
97          }
98
99          public final Boolean isOver() {
100             for (final String player : this.board) {
101                 if (player == null) {
102                     return false;
103                 }
104             }
105
106             return true;
107         }
108     }
109
110     private static class Player extends Thread {
111         private final static String GREETING_REQUEST = "GREETING";
112         private final static String START_REQUEST = "START";
113         private final static String MOVE_REQUEST = "MOVE";
114
115         private Socket socket;
116         private String role;
117         private String opponentRole;
118
119         private BufferedReader input;
120         private PrintWriter output;
121
122         private Player opponent = null;
123         private Game game = null;
124
125         public Player(final Socket socket, final String role, final String opponentRole) {
126             this.socket = socket;
127             this.role = role;
128             this.opponentRole = opponentRole;
129
130             try {
131                 this.input = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));
132                 this.output = new PrintWriter(socket.getOutputStream(), autoFlush: true);
133                 this.output.println(Player.GREETING_REQUEST + this.role + this.opponentRole);
134             } catch (final IOException exception) {
135                 System.out.println(x: "END");
136             }
137         }
138
139         public final String getRole() {
140             return this.role;
141         }
142
143         public final void setOpponent(final Player opponent) {
144             this.opponent = opponent;
145         }
146
147         public final Player getOpponent() {
148             return this.opponent;
149         }
150
151         public final void setGame(final Game game) {
152             this.game = game;
153         }
154
```

Продолжение рисунка 3.

```
155         public final void reportMove(final Integer boxNumber) {
156             this.opponent.output.println("MOVE" + boxNumber);
157         }
158
159         public final void reportWin(final Integer boxNumber) {
160             this.output.println("WIN" + boxNumber);
161         }
162
163         public final void reportLose(final Integer boxNumber) {
164             this.output.println("LOSE" + boxNumber);
165         }
166
167         public final void reportDraw(final Integer boxNumber) {
168             this.output.println("DRAW" + boxNumber);
169         }
170
171         @Override
172         public final void run() {
173             try {
174                 this.output.println(Player.START_REQUEST);
175
176                 while (true) {
177                     String request = this.input.readLine();
178
179                     if (request == null || request.equals( anObject: "")) {
180                         continue;
181                     }
182
183                     if (request.startsWith(MOVE_REQUEST)) {
184                         Integer boxNumber = request.charAt(MOVE_REQUEST.length()) - '0';
185                         this.game.move(this, boxNumber);
186                     }
187
188                     if (this.game.isOver()) {
189                         return;
190                     }
191                 }
192             } catch (final IOException exception) {
193                 Logger.getLogger(Server.class.getName()).log(Level.SEVERE, msg: null, exception);
194             } finally {
195                 try {
196                     socket.close();
197                 } catch (final IOException exception) {
198                     Logger.getLogger(Server.class.getName()).log(Level.SEVERE, msg: null, exception);
199                 }
200             }
201         }
202     }
203 }
204
```

Starter.java ×

src > main > java > com > example > Starter.java > ...

```
1   package com.example;
2
3   public final class Starter {
        Run | Debug
4       public final static void main(final String[] args) {
5           App.main(args);
6       }
7   }
8
```

Рисунок 4 - Исходный код файла Starter.java.

App.java 3 ×

src > main > java > com > example > App.java > ...

```java
package com.example;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

/**
 * JavaFX App
 */
public final class App extends Application {
    private static Scene scene;

    @Override
    public final void start(final Stage stage) throws IOException {
        App.scene = new Scene(loadFXML(fxml: "main"), 320, 480);
        stage.setResizable(false);
        stage.setScene(App.scene);
        stage.show();
    }

    private static final void setRoot(final String fxml) throws IOException {
        App.scene.setRoot(loadFXML(fxml));
    }

    public final static Parent loadFXML(final String fxml) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource(fxml + ".fxml"));
        return fxmlLoader.load();
    }

    Run | Debug
    public final static void main(final String[] args) {
        launch();
    }
}

```

Рисунок 5 - Исходный код файла App.java.

MainController.java 1 ×

src > main > java > com > example > MainController.java > ...

```java
package com.example;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.URL;
import java.net.UnknownHostException;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;

import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

```

Рисунок 6 - Исходный код файла MainController.java.

Продолжение рисунка 6.

```
24  public final class MainController implements Initializable {
25      @FXML
26      private Button connectButton = new Button();
27
28      @FXML
29      private TextField serverAddressTextField = new TextField();
30      @FXML
31      private TextField serverPortTextField = new TextField();
32      @FXML
33      private TextField playerTextField = new TextField();
34      @FXML
35      private TextField statusTextField = new TextField();
36
37      @FXML
38      private Button button0 = new Button();
39      @FXML
40      private Button button1 = new Button();
41      @FXML
42      private Button button2 = new Button();
43      @FXML
44      private Button button3 = new Button();
45      @FXML
46      private Button button4 = new Button();
47      @FXML
48      private Button button5 = new Button();
49      @FXML
50      private Button button6 = new Button();
51      @FXML
52      private Button button7 = new Button();
53      @FXML
54      private Button button8 = new Button();
55
56      private Button currentButton = null;
57
58      private final static String GREETING_RESPONSE = "GREETING";
59      private final static String START_RESPONSE = "START";
60      private final static String MOVE_RESPONSE = "MOVE";
61      private final static String WIN_RESPONSE = "WIN";
62      private final static String LOSE_RESPONSE = "LOSE";
63      private final static String DRAW_RESPONSE = "DRAW";
64
65      private final static String WAINTING_STATUS = "Wait opponent...";
66      private final static String YOUR_TURN_STATUS = "It's your turn!";
67      private final static String WIN_STATUS = "You won!";
68      private final static String LOSE_STATUS = "You lost!";
69      private final static String DRAW_STATUS = "Draw!";
70
71      private final static String ROLE_X = "X";
72      private final static String ROLE_0 = "0";
73
74      private String myRole = "";
75      private String opponentRole = "";
76
77      private Image myImage;
78      private Image opponentImage;
79
80      private Socket socket;
81      private BufferedReader input;
82      private PrintWriter output;
83
```

Продолжение рисунка 6.

```
84      @Override
85      public final void initialize(final URL arg0, final ResourceBundle arg1) {
86          try {
87              if (this.serverAddressTextField.getText().equals( anObject: "")
88                      || this.serverPortTextField.getText().equals( anObject: "")) {
89                  return;
90              }
91
92              this.button0.setDisable(false);
93              this.button1.setDisable(false);
94              this.button2.setDisable(false);
95              this.button3.setDisable(false);
96              this.button4.setDisable(false);
97              this.button5.setDisable(false);
98              this.button6.setDisable(false);
99              this.button7.setDisable(false);
100             this.button8.setDisable(false);
101
102             this.button0.setGraphic(null);
103             this.button1.setGraphic(null);
104             this.button2.setGraphic(null);
105             this.button3.setGraphic(null);
106             this.button4.setGraphic(null);
107             this.button5.setGraphic(null);
108             this.button6.setGraphic(null);
109             this.button7.setGraphic(null);
110             this.button8.setGraphic(null);
111
112             this.myRole = "";
113             this.opponentRole = "";
114
115             this.statusTextField.setText(MainController.WAINTING_STATUS);
116             this.playerTextField.setText(null);
117
118             this.socket = new Socket(this.serverAddressTextField.getText(),
119                     Integer.parseInt(this.serverPortTextField.getText()));
120
121             this.input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
122             this.output = new PrintWriter(this.socket.getOutputStream(), autoFlush: true);
123
124             String response = this.input.readLine();
125
126             if (!response.startsWith(MainController.GREETING_RESPONSE)) {
127                 this.statusTextField.setText("Cannot connect!");
128                 return;
129             }
130
131             this.myRole += response.charAt(MainController.GREETING_RESPONSE.length());
132             this.opponentRole += response.charAt(MainController.GREETING_RESPONSE.length() + 1);
133             this.playerTextField.setText(this.myRole);
134
135             this.myImage = new Image(getClass().getResourceAsStream(this.myRole + ".gif"));
136             this.opponentImage = new Image(getClass().getResourceAsStream(this.opponentRole + ".gif"));
137
138             EventHandler<ActionEvent> onButtonAction = new EventHandler<ActionEvent>() {
139                 @Override
140                 public final void handle(ActionEvent event) {
141                     if (statusTextField.getText().equals(MainController.WAINTING_STATUS)
142                             || statusTextField.getText().equals(MainController.WIN_STATUS)
143                             || statusTextField.getText().equals(MainController.LOSE_STATUS)
144                             || statusTextField.getText().equals(MainController.DRAW_STATUS)) {
145                         return;
146                     }
147
148                     statusTextField.setText(MainController.WAINTING_STATUS);
149
150                     Button button = (Button) event.getSource();
151                     String buttonId = button.getId();
152                     button.setDisable(true);
153                     button.setGraphic(new ImageView(myImage));
154                     output.println("MOVE" + buttonId.charAt(buttonId.length() - 1));
155
156                     Thread thread = new Thread(new Runnable() {
157                         @Override
158                         public final void run() {
159                             waitForOpponent();
160                         }
161                     });
162
163                     thread.start();
164                 }
165             };
```

Продолжение рисунка 6.

```
            this.button0.setOnAction(onButtonAction);
            this.button1.setOnAction(onButtonAction);
            this.button2.setOnAction(onButtonAction);
            this.button3.setOnAction(onButtonAction);
            this.button4.setOnAction(onButtonAction);
            this.button5.setOnAction(onButtonAction);
            this.button6.setOnAction(onButtonAction);
            this.button7.setOnAction(onButtonAction);
            this.button8.setOnAction(onButtonAction);

            Thread thread = new Thread(new Runnable() {
                @Override
                public final void run() {
                    waitForOpponent();
                }
            });

            thread.start();
            Thread.sleep( millis: 500);
        } catch (final UnknownHostException exception) {
            Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
        } catch (final IOException exception) {
            Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
        } catch (final Exception exception) {
            Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
        }
    }

    @FXML
    private final void connect() {
        if (this.statusTextField.getText().equals(MainController.WAINTING_STATUS)
                || this.statusTextField.getText().equals(MainController.YOUR_TURN_STATUS)) {
            return;
        }

        if (this.output != null) {
            this.output.close();
        }

        if (this.input != null) {
            try {
                this.input.close();
            } catch (final IOException exception) {
                Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
            }
        }

        if (this.socket != null) {
            try {
                this.socket.close();
            } catch (final IOException exception) {
                Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
            }
        }

        this.initialize( arg0: null, arg1: null);
    }

```

Продолжение рисунка 6.

```
225     private final void waitForOpponent() {
226         try {
227             while (true) {
228                 String response = this.input.readLine();
229 
230                 if (response == null || response.equals( anObject: "")) {
231                     continue;
232                 }
233 
234                 if (response.startsWith(MainController.START_RESPONSE)) {
235                     if (this.myRole.equals(MainController.ROLE_X)) {
236                         this.statusTextField.setText(MainController.YOUR_TURN_STATUS);
237                         return;
238                     }
239 
240                     continue;
241                 }
242 
243                 Integer boxNumber = null;
244 
245                 if (response.startsWith(MainController.MOVE_RESPONSE)) {
246                     Platform.runLater(() -> {
247                         this.statusTextField.setText(MainController.YOUR_TURN_STATUS);
248                     });
249 
250                     boxNumber = response.charAt(MOVE_RESPONSE.length()) - '0';
251                 }
252 
253                 if (response.startsWith(MainController.WIN_RESPONSE)) {
254                     Platform.runLater(() -> {
255                         this.statusTextField.setText(MainController.WIN_STATUS);
256                     });
257 
258                     boxNumber = response.charAt(WIN_RESPONSE.length()) - '0';
259                 }
260 
261                 if (response.startsWith(MainController.LOSE_RESPONSE)) {
262                     Platform.runLater(() -> {
263                         this.statusTextField.setText(MainController.LOSE_STATUS);
264                     });
265 
266                     boxNumber = response.charAt(LOSE_RESPONSE.length()) - '0';
267                 }
268 
269                 if (response.startsWith(MainController.DRAW_RESPONSE)) {
270                     Platform.runLater(() -> {
271                         this.statusTextField.setText(MainController.DRAW_STATUS);
272                     });
273 
274                     boxNumber = response.charAt(DRAW_RESPONSE.length()) - '0';
275                 }
276 
277                 if (boxNumber == null) {
278                     return;
279                 }
280 
281                 switch (boxNumber) {
282                     case 0:
283                         this.currentButton = this.button0;
284                         break;
285                     case 1:
286                         this.currentButton = this.button1;
287                         break;
288                     case 2:
289                         this.currentButton = this.button2;
290                         break;
291                     case 3:
292                         this.currentButton = this.button3;
293                         break;
294                     case 4:
295                         this.currentButton = this.button4;
296                         break;
297                     case 5:
298                         this.currentButton = this.button5;
299                         break;
300                     case 6:
301                         this.currentButton = this.button6;
302                         break;
303                     case 7:
304                         this.currentButton = this.button7;
305                         break;
306                     case 8:
307                         this.currentButton = this.button8;
308                         break;
309 
310                 }
311 
```

Продолжение рисунка 6.

```
312                    Platform.runLater(() -> {
313                        if (this.currentButton.isDisable()) {
314                            return;
315                        }
316
317                        this.currentButton.setDisable(true);
318                        this.currentButton.setGraphic(new ImageView(this.opponentImage));
319                    });
320                }
321            } catch (final IOException exception) {
322                Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
323            } catch (final Exception exception) {
324                Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, msg: null, exception);
325            }
326        }
327    };
328
```

Рисунок 7 - Графический интерфейс клиента.

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
```

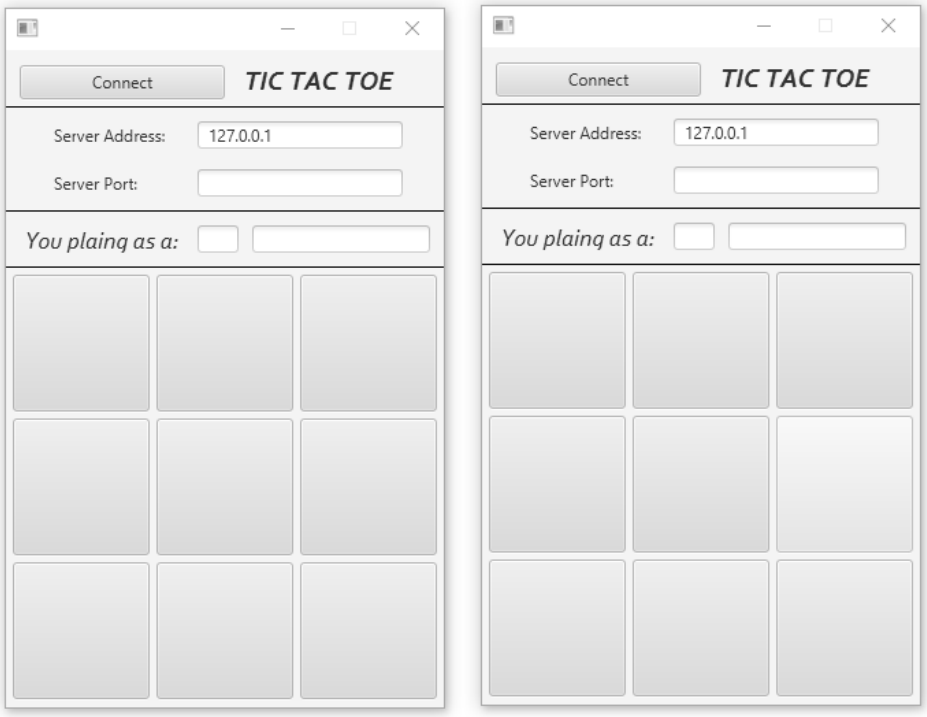

Рисунок 8 - Тестирование работы программ (запуск).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
```

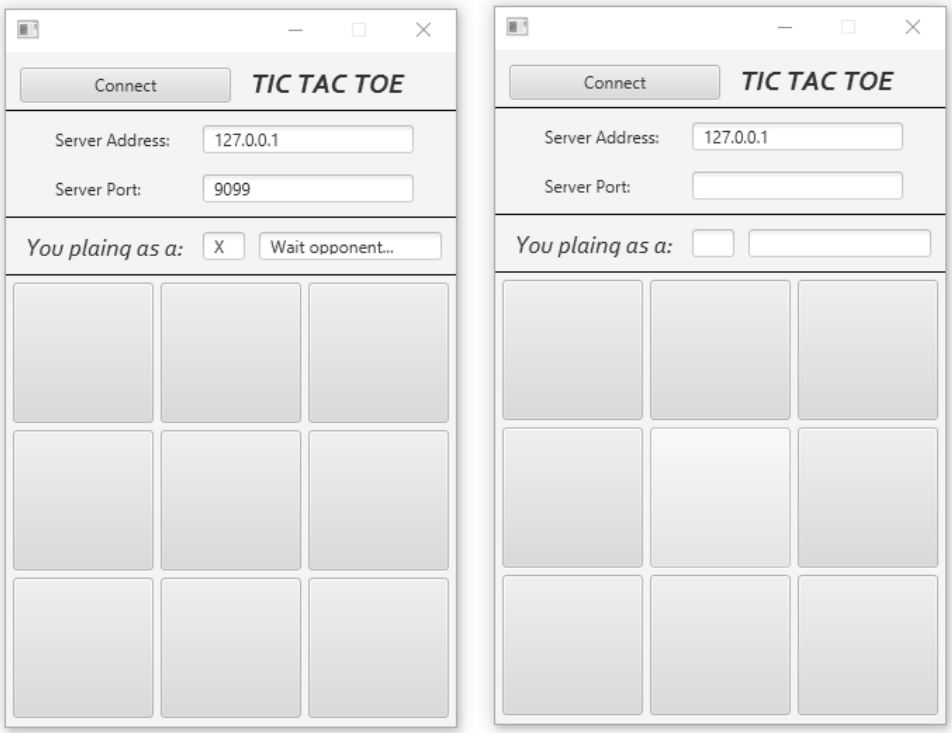

Рисунок 9 - Тестирование работы программ (первый игрок подключен).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
```

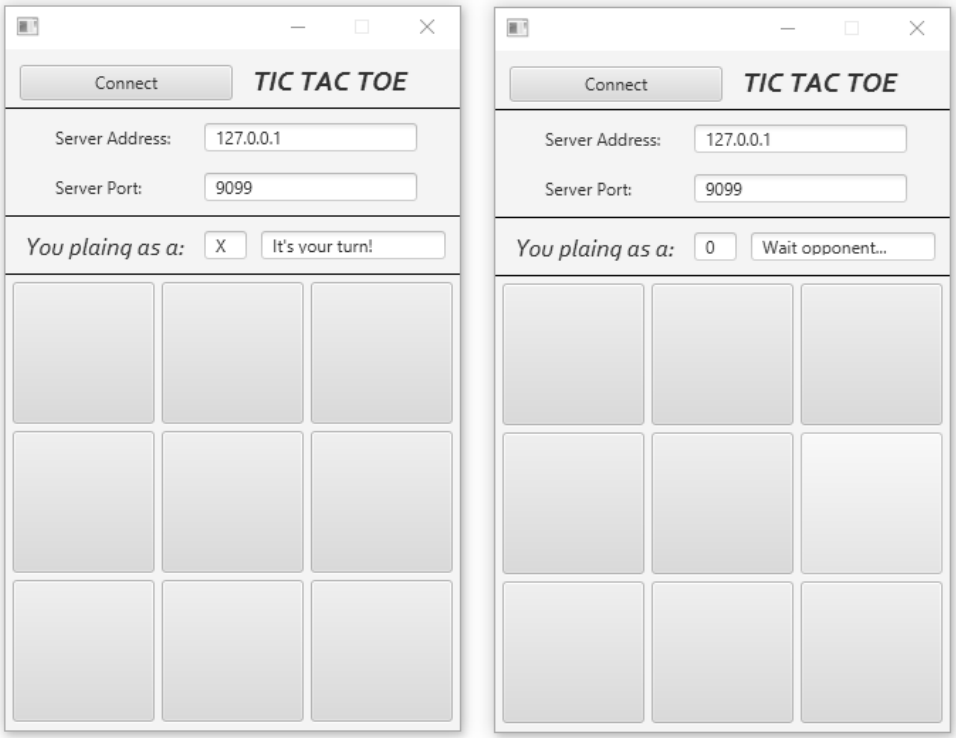

Рисунок 10 - Тестирование работы программ (второй игрок подключен).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
Player X move at index 0
```

Рисунок 11 - Тестирование работы программ (ход первого игрока).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
Player X move at index 0
Player 0 move at index 8
```

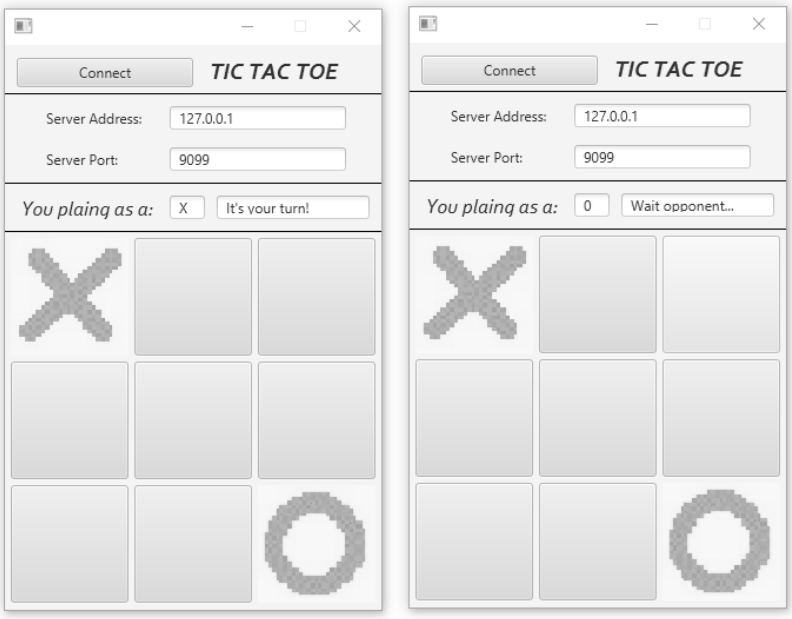

Рисунок 12 - Тестирование работы программ (ход второго игрока).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
Player X move at index 0
Player 0 move at index 8
Player X move at index 2
Player 0 move at index 1
Player X move at index 6
Player 0 move at index 3
Player X move at index 4
Player X won!
Player 0 lost!
Server is Running on port 9099.
Waiting for players...
```

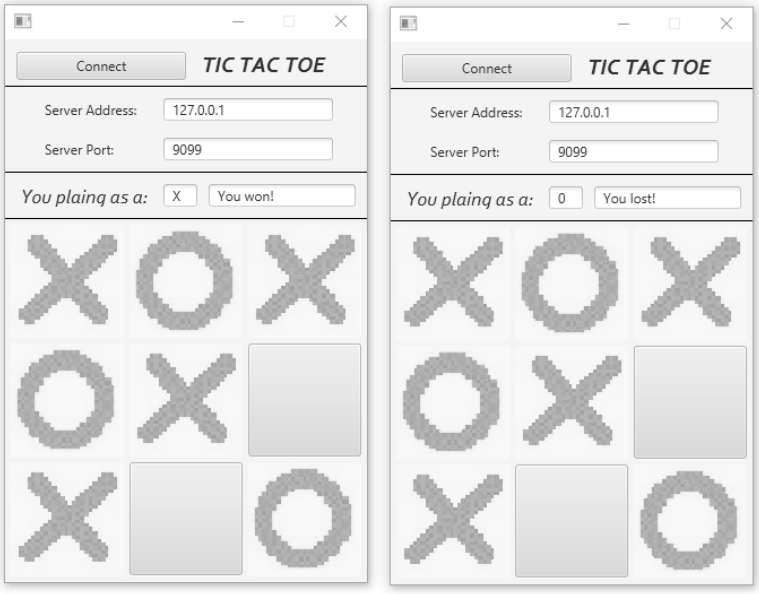

Рисунок 13 - Тестирование работы программ (окончание игры).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
Player X move at index 0
Player 0 move at index 8
Player X move at index 2
Player 0 move at index 1
Player X move at index 6
Player 0 move at index 3
Player X move at index 4
Player X won!
Player 0 lost!
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
```

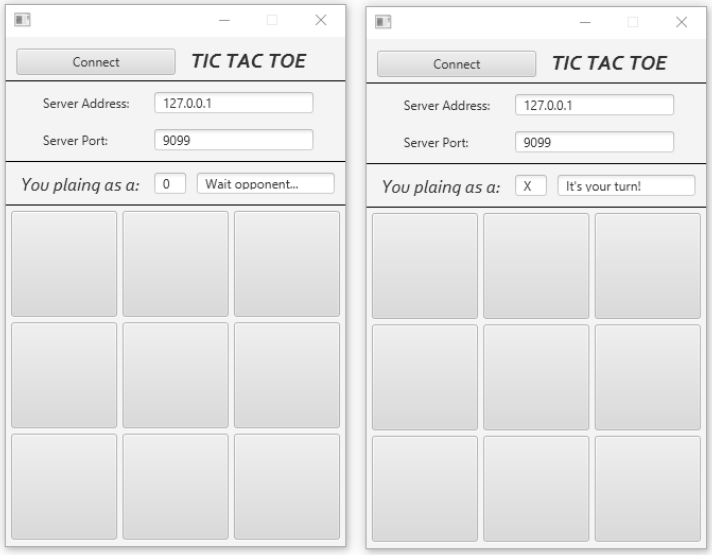

Рисунок 14 - Тестирование работы программ (переподключение).

```
PS D:\Documents\Visual Studio Code\Java\server> java -jar .\server.jar
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
Player X move at index 0
Player 0 move at index 8
Player X move at index 2
Player 0 move at index 1
Player X move at index 6
Player 0 move at index 3
Player X move at index 4
Player X won!
Player 0 lost!
Server is Running on port 9099.
Waiting for players...
Player X connected.
Player 0 connected.
Player X move at index 2
Player 0 move at index 6
Player X move at index 0
Player 0 move at index 1
Player X move at index 7
Player 0 move at index 4
Player X move at index 5
Player 0 move at index 8
Player X move at index 3
Draw!
Server is Running on port 9099.
Waiting for players...
```

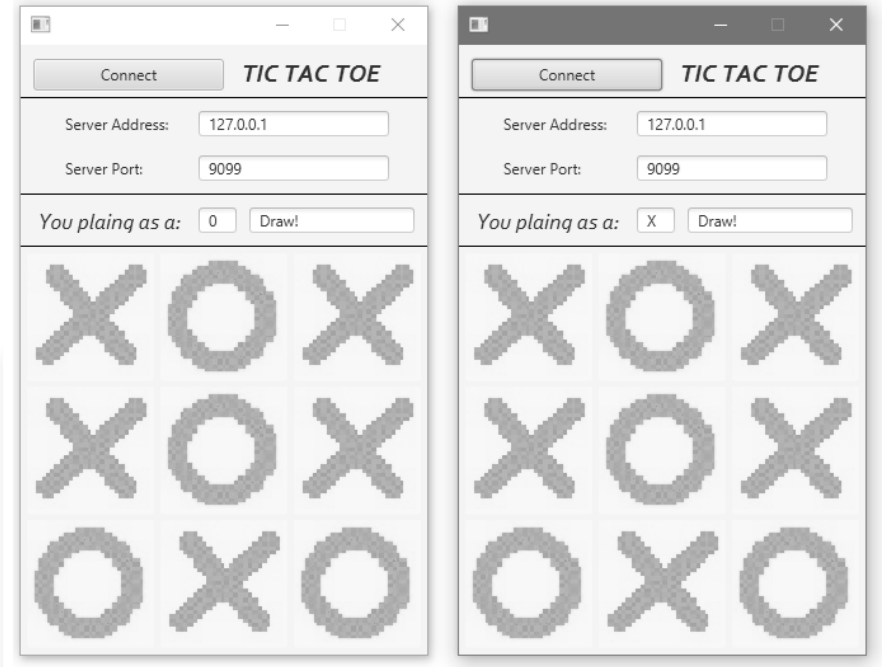

Рисунок 15 - Тестирование работы программ (ничья).

Вывод: Освоил приемы разработки оконных клиент-серверных приложений на Java с использованием сокетов.