МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №12

Специальность ПО-5(о)

Выполнили
Прокопчик Егор,
Бриштен Дмитрий,
студенты группы ПО-5
Проверил
А. А. Крощенко
ст. преп. каф. ИИТ

Брест 2022

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

Вариант 8

Задание. Игра «Крестики-нолики». Классическая игра для двух игроков на поле 3x3.

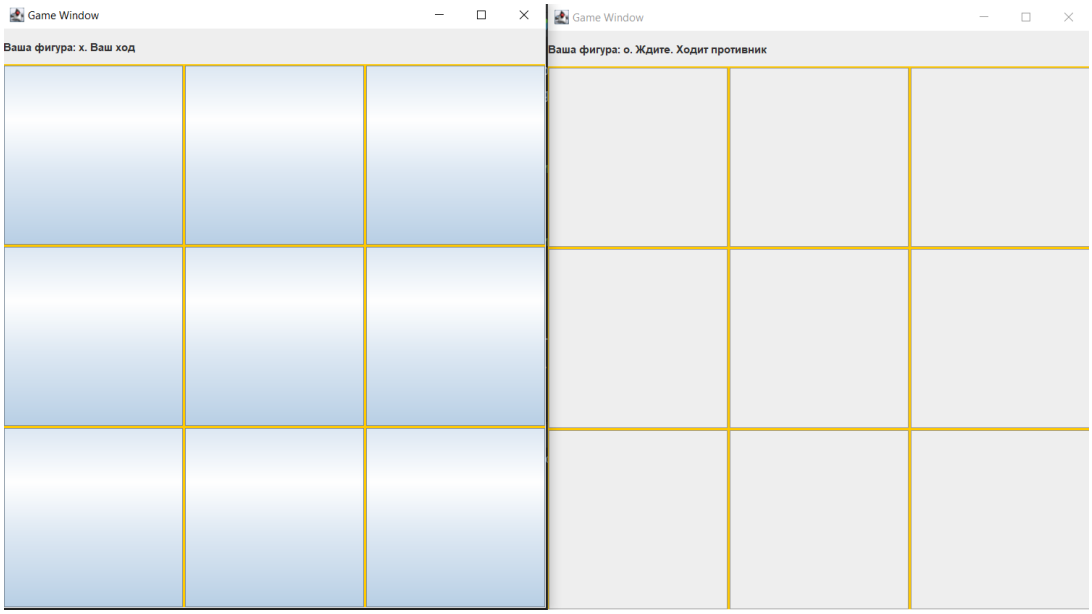

Рисунок 1 – Запуск игры

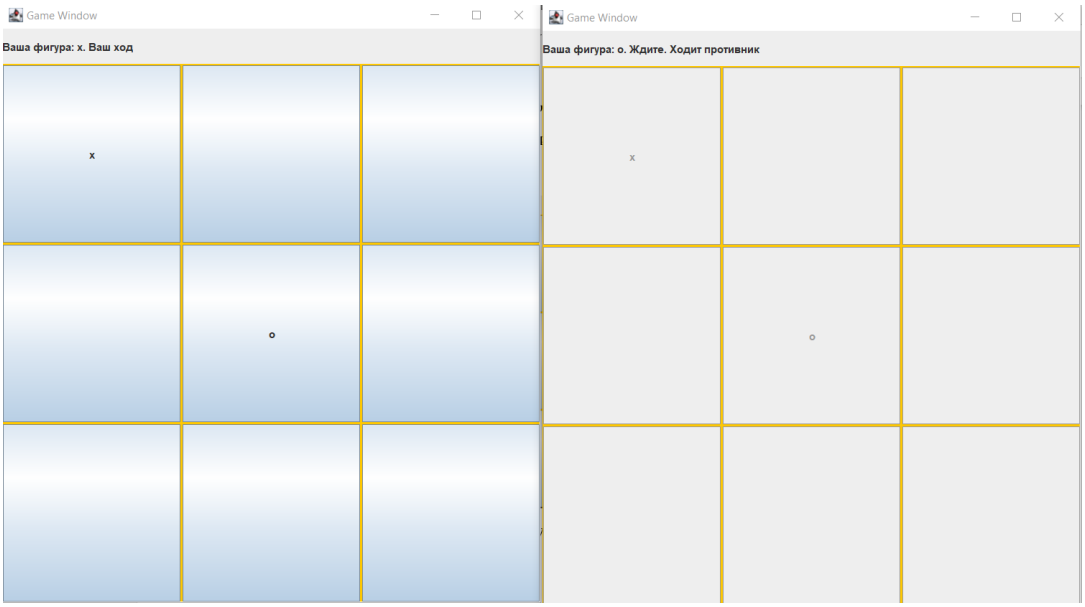

Рисунок 2 – Ходы игроков

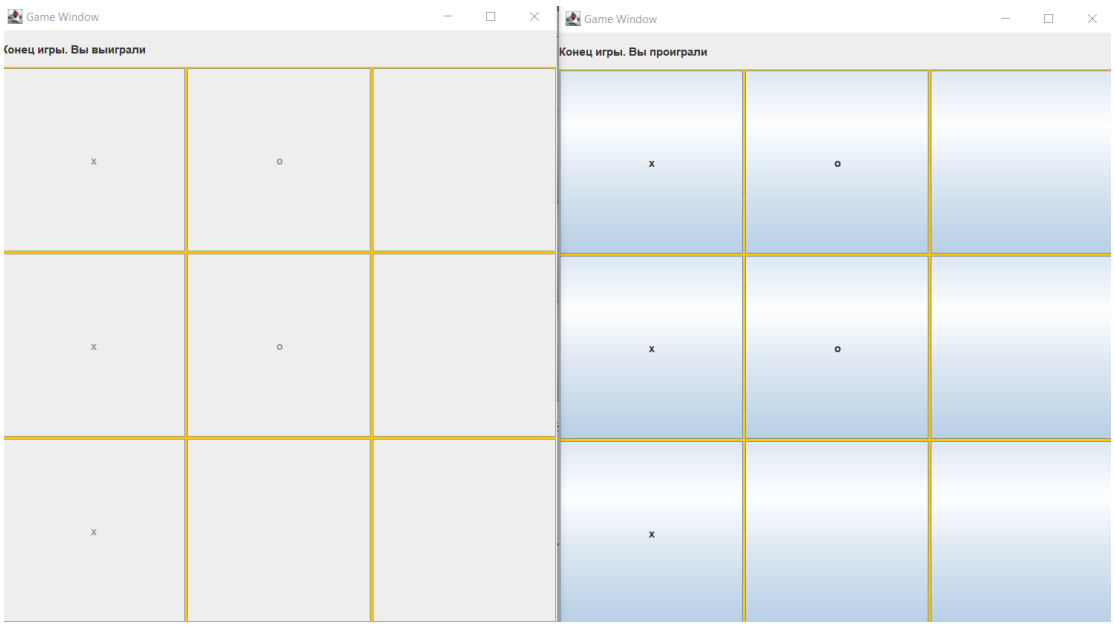

Рисунок 3 – Конец игры

Код программы:

Сервер

TcpServer.java

```
public class TcpServer {

    public static void main(String[] args) {

        int port = DEFAULT_PORT;
        if (args.length > 0) {
            port = Integer.parseInt(args[0]);
        }

        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(port);
        } catch (IOException e) {
            System.out.println("Порт занят: " + port);
            System.exit(-1);
        }

        try {
            Player players[] = new Player[2];

            for (int i = 0; i < 2; ++i) {
                players[i] = new Player();
                players[i].waitConnection(serverSocket);
            }

            System.out.println("Game started...\n");

            GameBoard gameboard = new GameBoard();
            generateRandomTypes(players);
```

```
            while (true) {
                System.out.println("Gameboard on server: " + gameboard.toString() + "\n");

                for (int i = 0; i < 2; ++i) {
                    players[i].send(gameboard);
                }

                if (gameboard.getWinner() != '_')
                    break;

                Messages.Move move = null;
                for (int i = 0; i < 2; ++i) {
                    Messages.Move currentMove = players[i].readMoveIfActive(gameboard.currentMove());
                    if (currentMove == null)
                        continue;
                    move = currentMove;
                }
                gameboard.process(move);
            }

        } catch (IOException e) {
            e.printStackTrace();
            System.exit(-1);
        }
    }

    static void generateRandomTypes(Player[] players) {
        Random rnd = new Random();
        int value = rnd.nextInt(1);

        if (value == 0) {
            players[0].set_type('x');
            players[1].set_type('o');
        }
        else {
            players[0].set_type('o');
            players[1].set_type('x');
        }
    }

    private static final int DEFAULT_PORT = 11122;
}
```

Player.java

```
public class Player {
    Player() {
        m_clientSocket = null;
        m_in = null;
        m_out = null;
        m_playerType = '_';
    }

    void waitConnection(ServerSocket serverSocket) throws IOException {
        m_clientSocket = serverSocket.accept();
        System.out.print("Connection accepted.\n");
```

```
        m_in = m_clientSocket.getInputStream();
        m_out = m_clientSocket.getOutputStream();
    }

    void send(GameBoard board) {
        Gson gson = new Gson();

        Messages.Board boardMessage = new Messages.Board();
        boardMessage.gameboard = board.toString();

        boardMessage.your_type = m_playerType;
        boardMessage.move = board.currentMove();
        boardMessage.winner = board.getWinner();

        Common.writeBytes(m_out, gson.toJson(boardMessage));
    }

    void set_type(char type) {
        m_playerType = type;
    }

    Messages.Move readMoveIfActive(char currentPlayerType) {
        if (m_playerType != currentPlayerType)
            return null;

        String buffer = Common.readBytes(m_in);

        Gson gson = new Gson();
        return gson.fromJson(buffer, Messages.Move.class);
    }

    private OutputStream m_out;
    private InputStream m_in;
    private Socket m_clientSocket;
    private char m_playerType;
}
```

Messages.java

```
public class Messages {
    public static class Move {
        public int x, y;
    }

    public static class Board {
        public String gameboard;
        public char your_type;
        public char move;
        public char winner;
    }
}
```

GameBoard.java

```
public class GameBoard {
    GameBoard() {
```

```
    m_board = new char[Size][Size];

    for (int i = 0; i < Size; i++) {
        for (int j = 0; j < Size; j++) {
            m_board[i][j] = '_';
        }
    }

    m_currentMove = 'x';
}

public String toString() {
    String buffer = "";
    for (int i = 0; i < Size; i++) {
        for (int j = 0; j < Size; j++) {
            buffer += m_board[i][j];
        }
    }
    return buffer;
}

public char currentMove() {
    return m_currentMove;
}

private void changeActivePlayer() {
    if (m_currentMove == 'x')
        m_currentMove = 'o';
    else
        m_currentMove = 'x';
}

public void process(Messages.Move move) {
    if (m_board[move.x][move.y] != '_')
        return;
    m_board[move.x][move.y] = m_currentMove;
    changeActivePlayer();
}

public char getWinner() {
    for (int i = 0; i < Size; i++) {
        for (int j = 0; j < Size; j++) {
            if (checkFrom(i, j) == true)
                return m_board[i][j];
        }
    }
    return '_';
}

private boolean checkFrom(int x, int y) {
    char player = m_board[x][y];
    if (player == '_')
        return false;

    boolean xWin = true, yWin = true, xyWin = true;
    for (int i = 0; i < 3; ++i) {
        if (i + x >= Size || m_board[i+x][y] != player)
```

```
            xWin = false;
         if (i + y >= Size || m_board[x][i+y] != player)
            yWin = false;
         if (i + y >= Size || i + x >= Size || m_board[x+i][i+y] != player)
            xyWin = false;
      }

      return xyWin || xWin || yWin;
   }

   private char[][] m_board;
   private char m_currentMove;
   public final int Size = 3;
}
```

Common.java

```
public class Common {

   public static String readBytes(java.io.InputStream stream) {
      try {
         BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
         int length = reader.read();

         String string = "";
         for (int i = 0; i < length; ++i) {
            string += (char)reader.read();
         }

         return string;
      }
      catch (IOException ex) {
         System.out.println("I/O Error!");
      }
      return null;
   }

   public static void writeBytes(java.io.OutputStream stream, String string) {
      try {
         BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));
         writer.write(string.length());
         writer.write(string);
         writer.flush();
      }
      catch (IOException ex) {
         System.out.println("I/O Error!");
      }
   }
}
```

Клиент

TcpClient.java

```
public class TcpClient {
   public static void main(String[] args) {
      String host = DEFAULT_HOST;
      int port = DEFAULT_PORT;
```

```
        if (args.length > 0) {
            host = args[0];
        }
        if (args.length > 1) {
            port = Integer.parseInt(args[1]);
        }

        try {
            Socket socket = new Socket(host, port);

            System.out.println("connected.\n");

            OutputStream out = socket.getOutputStream();
            InputStream in = socket.getInputStream();

            GameWindow gamewindow = new GameWindow(out);
            gamewindow.pack();
            gamewindow.setVisible(true);

            while (true) {
                String fromServer = Common.readBytes(in);
                Gson gson = new Gson();

                Messages.Board boardMessage = gson.fromJson(fromServer, Messages.Board.class);

                if (boardMessage == null)
                    break;

                gamewindow.load(boardMessage);

                if (boardMessage.winner != '_')
                    break;
            }
        } catch (UnknownHostException e) {
            System.out.println("Неизвестный хост: " + host);
            System.exit(-1);
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(-1);
        }
    }

    private static final String DEFAULT_HOST = "localhost";
    private static final int DEFAULT_PORT = 11122;
}
```

Messages.java

```
public class Messages {
    public static class Move {
        public int x, y;
    }

    public static class Board {
        public String gameboard;
        public char your_type;
        public char move;
```

```
        public char winner;
    }
}
```

GameWindow.java

```
public class GameWindow extends JFrame {
    public GameWindow(OutputStream socketOut) {
        super("Game Window");
        createGUI(socketOut);
    }

    private void createGUI(OutputStream socketOut) {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel mainPanel = new JPanel(new BorderLayout());
        add(mainPanel);

        JPanel gridPanel = new JPanel(new GridLayout(Size, Size, 2, 2));
        gridPanel.setBackground(Color.orange);

        m_message = new JLabel("Ожидайте подключения противника...");
        m_message.setPreferredSize(new Dimension(600, 40));
        m_message.setMinimumSize(new Dimension(100, 20));

        gridPanel.setPreferredSize(new Dimension(600, 600));

        mainPanel.add(m_message, BorderLayout.NORTH);
        mainPanel.add(gridPanel, BorderLayout.SOUTH);

        m_buttons = new GameButton[3][];
        for (int i = 0; i < 3; i++) {
            m_buttons[i] = new GameButton[3];
            for (int j = 0; j < 3; j++) {
                m_buttons[i][j] = new GameButton(i, j, socketOut);
                m_buttons[i][j].setMargin(new Insets(0, 0, 0, 0));
                m_buttons[i][j].setEnabled(false);

                gridPanel.add(m_buttons[i][j]);
            }
        }
        setSize(600, 600);
    }

    public void load(Messages.Board boardMessage) {
        boolean isActive = boardMessage.move == boardMessage.your_type;

        if (isActive == true) {
            m_message.setText("Ваша фигура: " + boardMessage.your_type + ". Ваш ход");
        }
        else {
            m_message.setText("Ваша фигура: " + boardMessage.your_type + ". Ждите. Ходит
противник");
        }

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

```
                char c = boardMessage.gameboard.charAt(i*Size + j);
                if (c == '_')
                    c = ' ';
                m_buttons[i][j].setText("" + c);
                m_buttons[i][j].setEnabled(isActive);
            }
        }

        if (boardMessage.winner != '_') {
            if (boardMessage.winner == boardMessage.your_type)
                m_message.setText("Конец игры. Вы выиграли");
            else {
                m_message.setText("Конец игры. Вы проиграли");
            }
            return;
        }
    }

    private GameButton[][] m_buttons;
    private JLabel m_message;
    public final int Size = 3;
}
```

GameButton.java

```
public class GameButton extends JButton {
    public GameButton(int x, int y, OutputStream socketOut) {
        super("");
        m_x = x;
        m_y = y;
        m_socketOut = socketOut;

        addActionListener(new MoveActionListener());
    }

    public class MoveActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Gson gson = new Gson();

            Messages.Move moveMessage = new Messages.Move();
            moveMessage.x = m_x;
            moveMessage.y = m_y;

            Common.writeBytes(m_socketOut, gson.toJson(moveMessage));
        }
    }

    private int m_x, m_y;
    private OutputStream m_socketOut;
}
```

Common.java

```
public class Common {
    public static String readBytes(java.io.InputStream stream) {
        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
```

```
            int length = reader.read();

            String string = "";
            for (int i = 0; i < length; ++i) {
               string += (char)reader.read();
            }

            return string;
         }
         catch (IOException ex) {
            System.out.println("I/O Error!");
         }
         return null;
      }

      public static void writeBytes(java.io.OutputStream stream, String string) {
         try {
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));
            writer.write(string.length());
            writer.write(string);
            writer.flush();
         }
         catch (IOException ex) {
            System.out.println("I/O Error!");
         }
      }

      public static String readString(java.io.InputStream stream) {
         try {
            BufferedReader br = new BufferedReader(new InputStreamReader(stream));
            return br.readLine();
         }
         catch (IOException ex) {
            System.out.println("I/O Error!");
         }
         return null;
      }

      public static void writeString(java.io.OutputStream stream, String string) {
         try {
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stream));
            writer.write(string + "\n");
            writer.flush();
         }
         catch (IOException ex) {
            System.out.println("I/O Error!");
         }
      }
   }
```

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