Міністерство освіти і наука України
Тернопільський національний технічний університет
Імені Івана Пулюя

Кафедра комп’ютерних систем та
мереж

Лабораторна робота № 8
З дисципліни “Програмування мовою JAVA”
на тему “Використання JAVA для створення мережевих програм”

Виконав(ла):
студент групи СІс-21
Підлатюк Денис Іванович

Перевірив(ла):
Луцків Андрій Мирославович

Тернопіль 2025

2. Теоретичні відомості

4-5. реалізує мережеву гру "хрестики-нулики", сервер реалізує логіку машини, клієнтом керує користувач;
(4 варіант через брак варіантів)

3. Виконання завдання

1. Створюємо два скрипта, один - сервер, інший - клієнт. На сервері будуть обрахунки, а клієнт лише надсилає команди, куди і як походити. Можна запускати програми з різних пристроїв, головне щоб вони були в одній мережі.

   код сервера -

```
import java.io.*;
import java.net.*;

public class TicTacToeServer {
    private static char[][] board = new char[3][3];
    private static PrintWriter out;

    public static void main(String[] args) throws
IOException {
        ServerSocket serverSocket = new
ServerSocket(5000);
        System.out.println("Сервер очікує
підключення...");
        Socket clientSocket = serverSocket.accept();
        System.out.println("Клієнт підключився");

        BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        out = new
PrintWriter(clientSocket.getOutputStream(), true);

        initBoard();

        String inputLine;
```

```
        while ((inputLine = in.readLine()) != null)
{
            String[] parts = inputLine.split(",");
            int row = Integer.parseInt(parts[0]);
            int col = Integer.parseInt(parts[1]);

            if (board[row][col] == '-') {
                board[row][col] = 'X';
                out.println("UPDATE,X," + row + ","
+ col);

                if (checkWin('X')) {
                    out.println("WIN");
                    break;
                }

                int[] move = bestMove();
                if (move != null) {
                    board[move[0]][move[1]] = 'O';
                    out.println("UPDATE,O," +
move[0] + "," + move[1]);
                    if (checkWin('O')) {
                        out.println("LOSE");
                        break;
                    }
                } else {
                    out.println("DRAW");
                    break;
                }
            }
        }

            clientSocket.close();
        serverSocket.close();
    }

    private static void initBoard() {
        for (int i = 0; i < 3; i++)
```

```
            for (int j = 0; j < 3; j++)
                board[i][j] = '-';
    }

    private static boolean checkWin(char player) {
        for (int i = 0; i < 3; i++)
            if (board[i][0] == player && board[i][1]
== player && board[i][2] == player)
                return true;

        for (int i = 0; i < 3; i++)
            if (board[0][i] == player && board[1][i]
== player && board[2][i] == player)
                return true;

        if (board[0][0] == player && board[1][1] ==
player && board[2][2] == player)
            return true;

        if (board[0][2] == player && board[1][1] ==
player && board[2][0] == player)
            return true;

        return false;
    }

    private static int[] bestMove() {
        // Спроба виграти
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                if (board[i][j] == '-') {
                    board[i][j] = 'O';
                    if (checkWin('O')) {
                        board[i][j] = '-';
                        return new int[]{i, j};
                    }
                    board[i][j] = '-';
                }
```

```
        // Спроба заблокувати 'X'
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                if (board[i][j] == '-') {
                    board[i][j] = 'X';
                    if (checkWin('X')) {
                        board[i][j] = '-';
                        return new int[]{i, j};
                    }
                    board[i][j] = '-';
                }

        // Перший вільний хід
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                if (board[i][j] == '-')
                    return new int[]{i, j};

        return null;
    }
}
```

Код клієнта -

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class TicTacToeClient extends JFrame {
    private JButton[][] buttons = new JButton[3][3];
    private BufferedReader in;
    private PrintWriter out;

    public TicTacToeClient() {
        setTitle("Хрестики-нулики");
        setSize(300, 300);
```

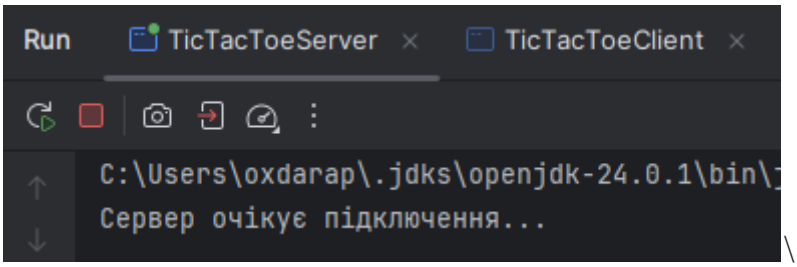
```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(3, 3));

        try {
            Socket socket = new Socket("localhost",
5000);
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new
PrintWriter(socket.getOutputStream(), true);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
"Помилка підключення: " + e.getMessage());
            System.exit(1);
        }

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                JButton btn = new JButton("");
                btn.setFont(new
Font(Font.SANS_SERIF, Font.BOLD, 40));
                final int row = i, col = j;
                btn.addActionListener(e ->
handleClick(btn, row, col));
                buttons[i][j] = btn;
                add(btn);
            }
        }

        setVisible(true);
    }

    private void handleClick(JButton btn, int row,
int col) {
        if (!btn.getText().equals("")) return;

        out.println(row + "," + col);
```

```
        // Окремий потік для обробки відповідей
        new Thread(() -> {
            try {
                while (true) {
                    String line = in.readLine();
                    if (line == null) break;

                    if (line.startsWith("UPDATE")) {
                        String[] parts =
line.split(",");
                        String symbol = parts[1];
                        int r =
Integer.parseInt(parts[2]);
                        int c =
Integer.parseInt(parts[3]);

SwingUtilities.invokeLater(() -> {

buttons[r][c].setText(symbol);

buttons[r][c].setEnabled(false);
                        });
                    } else if (line.equals("WIN")) {

SwingUtilities.invokeLater(() -> {

JOptionPane.showMessageDialog(this, "Ви
перемогли!");
                            System.exit(0);
                        });
                        break;
                    } else if (line.equals("LOSE"))
{

SwingUtilities.invokeLater(() -> {

JOptionPane.showMessageDialog(this, "Ви програли!");
```

```
                                System.exit(0);
                            });
                            break;
                        } else if (line.equals("DRAW"))
{

SwingUtilities.invokeLater(() -> {

JOptionPane.showMessageDialog(this, "Нічия!");
                                System.exit(0);
                            });
                            break;
                        } else {
                            break;
                        }
                    }
                } catch (IOException e) {
                    SwingUtilities.invokeLater(() ->

JOptionPane.showMessageDialog(this, "Помилка
зв'язку: " + e.getMessage())
                    );
                }
            }).start();
    }


    public static void main(String[] args) {

SwingUtilities.invokeLater(TicTacToeClient::new);
    }
}
```

Під час виконання лабораторної роботи виникли певні труднощі з тим, як правильно передавати команди, і в фінальній версії вирішено роздробити програму на декілька потоків, після чого проблему було усунено

Приклад виконання -



\

Рисунок 1 - Початковий стан сервера

При запуску сервера він очікує підключення на порт, який ми вказували всередині програми, після чого продовжує виконання.

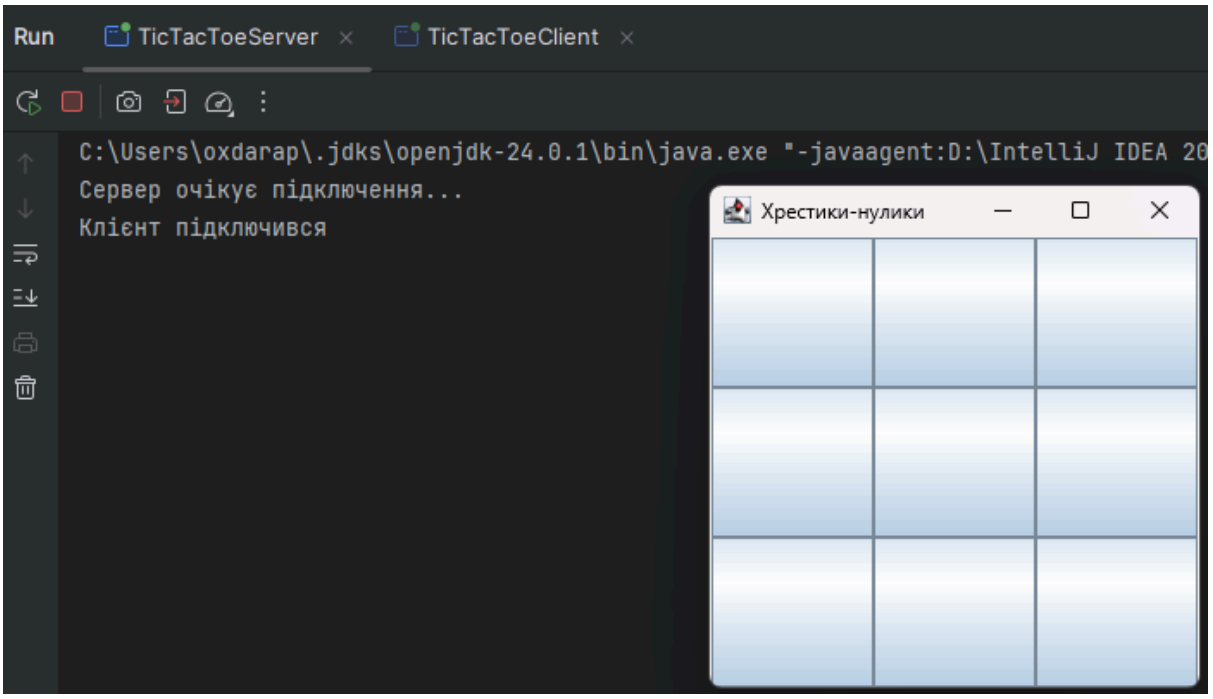


Рисунок 2 - Меню програми

Після підключення клієнта відкривається вікно з грою, натискання на кнопки надсилає команду на сервер, після чого бот повертає свій хід на клієнт. В даному випадку було запущено і сервер і клієнт з одного пристрою, але це не має бути проблемою, інтерфейс і команди обробляє сам клієнт.

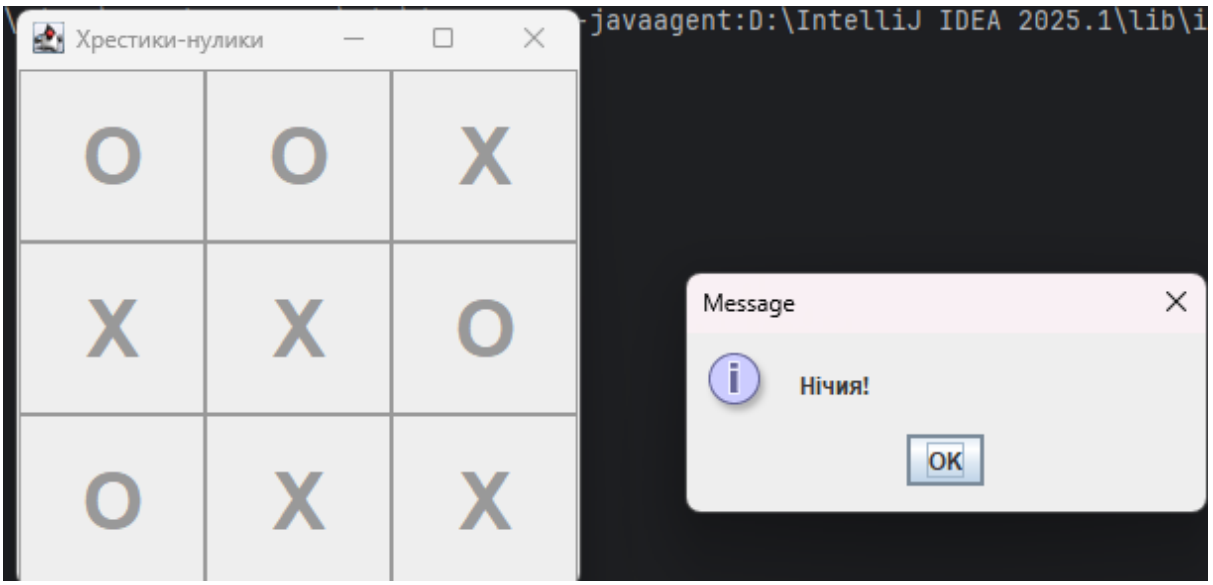


Рисунок 3 - Приклад гри

Бот написаний так, що він буде пробувати перемогти, а якщо це неможливо, то блокувати гравця. Такий підхід робить з нього хорошого суперника, але все-ще його можливо обіграти.

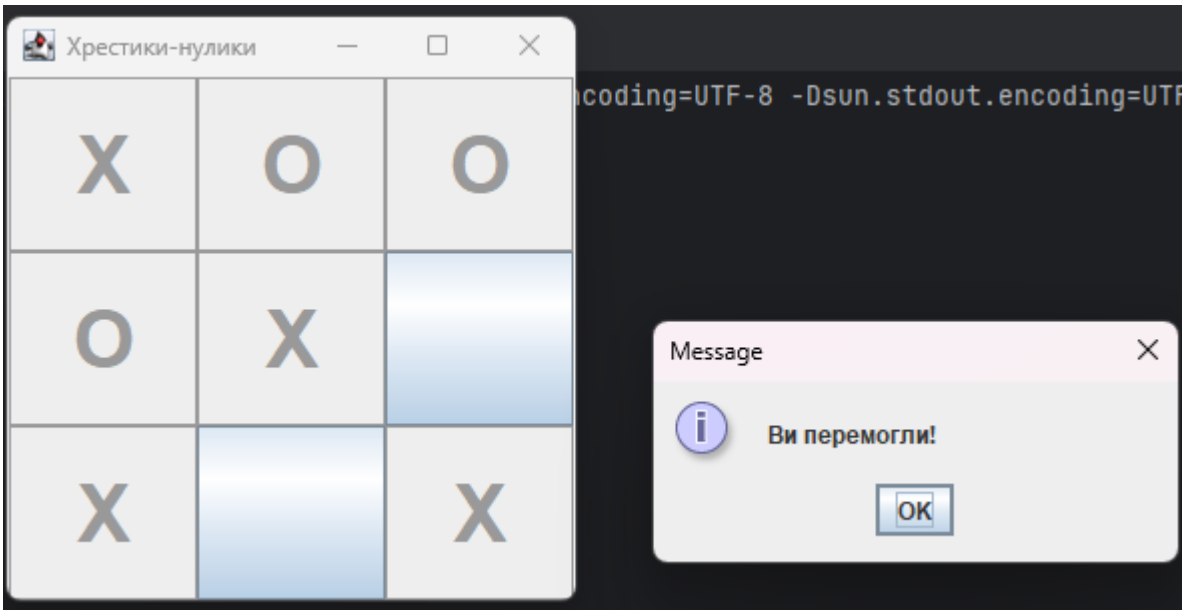


Рисунок 4 - Приклад перемоги

Висновок

Під час даної лабораторної роботи було освоєно спосіб роботи із сокетами та потоками у java.