Mihai Tuhari

FMI Mate-Info ID, anul 2

Rețele de calculatoare

1. Cerințe proiect

Un program Java ce trebuie sa contina cat mai multe dintre subiectele prezentate:

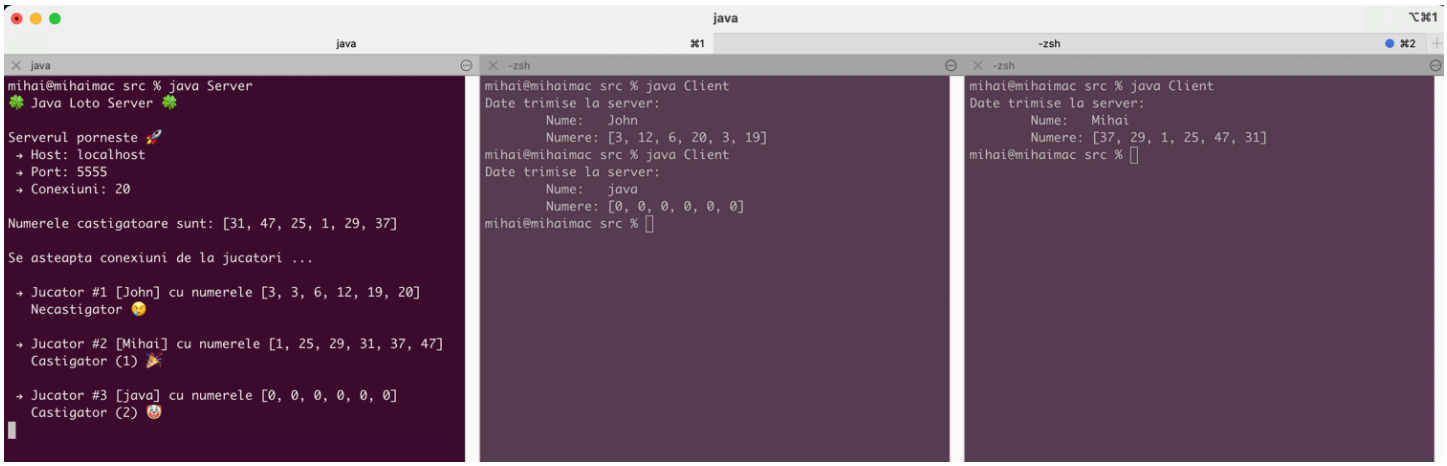
- Fluxuri, serializare
- Fire de executare
- Clienti - Server
- Interfete grafice, evenimente
- Resurse comune pe server accesate "simulatan" de client
- Metode de rezolvare a problemelor de concurență: monitoare, semafoare

---

2. Descriere proiect

A fost implementat un joc simplu de **Loto 6/49** 🍀
Se regaseste si in **Github** la 📎 https://github.com/mihaituhari/fmi-retele

Serverul generează numerele câștigătoare la instanțiere și apoi deschide conexiuni pentru **clienți**.



Clientii, prin interfata grafica, introduc numele și aleg cele 6 numere.

Aceste numere sunt trimise la server, unde se verifica dacă sunt castigatoare.

🥚 *Easter egg - folosește numele `java` pentru a castiga.*

3. Implementare proiect

Proiectul consta în 3 fișiere distincte, cu mai multe clase:

A. `Config.java` - pentru constante
B. `Server.java` - serverul care genereaza numere castigatoare si asteapta conexiuni de la clienți
C. `Client.java` - clientul cu interfata grafica, de unde se introduc numele și numerele jucate

3A. Config

```
public class Config {

   // Connection
   public static final String HOST = "localhost";
   public static final int PORT = 5555;

   public static final int ALLOWED_CONNECTIONS = 20;

   // Loto game
   public static final int LOTO_CHOICES = 6;
   public static final int LOTO_MAX = 49;

   public static final String CHEAT_WINNER_NAME = "java";

   // UI
   public static final int WINDOW_WIDTH = 480;
   public static final int WINDOW_HEIGHT = 300;
}
```

3B. Server

```
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.util.*;

class Server {
   static List<ObjectOutputStream> clientOutputStreams = new ArrayList<>();

   public static void main(String[] sss) throws Exception {
       System.out.println("🍀 Java Loto Server 🍀");

       System.out.println("\nServerul porneste 🚀");
       System.out.println(" → Host: " + Config.HOST);
       System.out.println(" → Port: " + Config.PORT);
       System.out.println(" → Conexiuni: " + Config.ALLOWED_CONNECTIONS);

       Game.generateWinningNumbers();
       System.out.println("\nNumerele castigatoare sunt: " + Game.winningNumbers);

       System.out.println("\nSe asteapta conexiuni de la jucatori ...");

       Game game = new Game();
       ServerSocket ss = new ServerSocket(Config.PORT);
       Socket cs;
```

```java
        for (int i = 0; i < Config.ALLOWED_CONNECTIONS; i++) {
            cs = ss.accept();
            new Connection(cs, game);

            try {
                ObjectOutputStream oos = new ObjectOutputStream(cs.getOutputStream());
                clientOutputStreams.add(oos);
                oos.writeObject(Game.participants);
                oos.flush();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        ss.close();
    }
}

class Connection extends Thread {
    Game game;
    Socket cs;

    Connection(Socket cs, Game game) throws Exception {
        this.cs = cs;
        this.game = game;

        start();
    }

    public void run() {
        try {
            InputStream is = cs.getInputStream();
            ObjectInputStream ois = new ObjectInputStream(is);

            PlayData data = (PlayData) ois.readObject();
            game.saveData(data);

            cs.close();
            is.close();
            ois.close();
        } catch (ClassNotFoundException | IOException e) {
            throw new RuntimeException(e);
        } finally {
            try {
                cs.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

class Game {
    public static List<Integer> winningNumbers = new ArrayList<>();

    public static int winners = 0;

    public static int participants = 0;

    public static void generateWinningNumbers() {
        Random rand = new Random();
        while (winningNumbers.size() < Config.LOTO_CHOICES) {
            int num = rand.nextInt(Config.LOTO_MAX) + 1;

            if (!winningNumbers.contains(num)) {
                winningNumbers.add(num);
            }
        }
```

```java
    }

    synchronized void saveData(PlayData data) {
        participants++;

        Collections.sort(data.numbers);
        Collections.sort(winningNumbers);

        boolean isWinner = data.numbers.equals(winningNumbers);
        boolean isCheater = Objects.equals(data.name, Config.CHEAT_WINNER_NAME);

        System.out.println("\n → Jucator #" + participants + " [" + data.name + "] cu numerele " +
data.numbers);

        if (isWinner || isCheater) {
            winners++;
            System.out.println("   Castigator (" + winners + ") " + (isCheater ? "🤡" : "🎉"));
        } else {
            System.out.println("   Necastigator 😢");
        }

        // Notify all clients of the new participant count
        for (ObjectOutputStream oos : Server.clientOutputStreams) {
            try {
                oos.writeObject(participants);
                oos.flush();
            } catch (SocketException e) {
                // Client closed connection
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

3C. Client

```java
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

class PlayModel extends Frame implements Serializable {
    boolean submitted = false;

    Choice[] numberChoices = new Choice[Config.LOTO_CHOICES];
    TextField nameField;
    Label participantsCounter;

    Button button;

    PlayModel() {
        setLayout(new GridLayout(5, 1));

        // Heading
        Panel headingPanel = new Panel();
        Label headingLabel = new Label("Bine ai venit la Java Loto 🍀");
        headingLabel.setFont(new Font("Arial", Font.BOLD, 24));
        headingPanel.add(headingLabel);
```

```java
        add(headingPanel);

        // Participants
        Panel participantsPanel = new Panel();
        participantsCounter = new Label("Participanti: ------");
        participantsCounter.setFont(new Font("Arial", Font.BOLD, 18));
        participantsCounter.setForeground(new Color(11, 161, 11));
        participantsPanel.add(participantsCounter);
        add(participantsPanel);

        // Name
        Panel namePanel = new Panel();
        Label nameLabel = new Label("💁‍♀️ Nume jucator: ");
        namePanel.add(nameLabel);
        nameField = new TextField(25);
        nameField.requestFocus();
        namePanel.add(nameField);
        add(namePanel);

        // Numbers choices
        Panel choicesPanel = new Panel(new FlowLayout());
        for (int i = 0; i < Config.LOTO_CHOICES; i++) {
            numberChoices[i] = new Choice();

            for (int j = 0; j <= Config.LOTO_MAX; j++) { // Add 0 as option
                numberChoices[i].addItem(Integer.toString(j));
            }

            choicesPanel.add(numberChoices[i]);
        }
        add(choicesPanel);

        // Button
        button = new Button("Inscrie numerele! SUCCES! 🍀");
        button.setBackground(new Color(11, 161, 11));
        add(button);

        // Handlers
        ClientHandler H = new ClientHandler(this);
        button.addActionListener(H);
    }
}

class ClientHandler implements ActionListener, Serializable {
    PlayModel play;

    ClientHandler(PlayModel play) {
        this.play = play;
    }

    public void actionPerformed(ActionEvent e) {
        play.submitted = true;
    }
}

class PlayData implements Serializable {
    List<Integer> numbers = new ArrayList<>();

    String name;

    PlayData(PlayModel play) {
        for (int i = 0; i < Config.LOTO_CHOICES; i++) {
            this.numbers.add(Integer.parseInt(play.numberChoices[i].getSelectedItem()));
        }

        this.name = play.nameField.getText();
    }
}
```

```java
class Client {
   private static volatile boolean running = true;

   public static void main(String[] sss) throws Exception {
       PlayModel play = new PlayModel();

       play.setSize(Config.WINDOW_WIDTH, Config.WINDOW_HEIGHT);
       play.setVisible(true);

       Socket cs = new Socket(Config.HOST, Config.PORT);

       new Thread(() -> {
           try {
               ObjectInputStream ois = new ObjectInputStream(cs.getInputStream());

               while (running) {
                   int participants = (int) ois.readObject();
                   play.participantsCounter.setText("Participanti: " + participants);
               }
           } catch (IOException | ClassNotFoundException e) {
               // Do nothing, connection closed
           }
       }).start();

       while (!play.submitted) {
           Thread.sleep(10);
       }

       OutputStream os = cs.getOutputStream();
       ObjectOutputStream oos = new ObjectOutputStream(os);

       PlayData sPlay = new PlayData(play);
       oos.writeObject(sPlay);

       System.out.println("Date trimise la server:");
       System.out.println("\tNume:\t" + sPlay.name);
       System.out.println("\tNumere:\t" + sPlay.numbers);

       running = false;

       play.dispose();
       os.close();
       oos.close();
       cs.close();

       System.exit(0);
   }
}
```