Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №12

Выполнил:

Студент группы ПО-5

Пищик А. В.

Проверил:

Крощенко А. А.

Брест 2022

**Вариант 11**

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

**Задание:**

Игра «Палочки» (с вытягиванием палочек). На игровом поле находятся двадцать деревянных палочек. Игроки, по очереди, берут одну, две или три палочки (сколько именно брать — решать игроку). Взявший последнюю палочку проигрывает, поэтому цель игры заключается в том, чтобы оставить эту палочку оппоненту.

**Client**

**Helloaplication.java**

package com.example.client;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {

@Override

public void start(Stage stage) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));

Scene scene = new Scene(fxmlLoader.load(), 720, 540);

stage.setTitle("Палочки!");

stage.setResizable(false);

stage.setScene(scene);

stage.show();

}

public static void main(String[] args) {

launch();

}

}

**Client.java**

package com.example.client;

import java.io.\*;

import java.net.Socket;

import java.util.Scanner;

public class Client {

private Socket socket;

private BufferedReader bufferedReader;

private BufferedWriter bufferedWriter;

private String username;

private int countWand;

private ClientGUI clientGUI;

private boolean isRun;

public Client(Socket socket, String username, ClientGUI clientGUI) {

try {

this.clientGUI = clientGUI;

this.socket = socket;

this.username = username;

this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

this.countWand = 20;

this.isRun = false;

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

}

public void sendName() {

try {

while (socket.isConnected()) {

bufferedWriter.write(username);

bufferedWriter.newLine();

bufferedWriter.flush();

return;

}

} catch (IOException e) {}

}

public void send(boolean isResult) {

try {

while (socket.isConnected()) {

if(isResult) {

return;

}

else {

String messageToSend = clientGUI.textNum.getText();

bufferedWriter.write(username + ": " + messageToSend);

bufferedWriter.newLine();

bufferedWriter.flush();

int sendValue = Integer.parseInt(messageToSend.substring(messageToSend.length() - 1));

if(isRun && sendValue > 0 && sendValue < 4) {

countWand -= sendValue;

String place = "";

for (int i = 0; i < countWand; i++) {

place += "| ";

}

clientGUI.place.setText(place);

clientGUI.size.setText(String.valueOf(countWand));

if(countWand < 1) {

return;

}

clientGUI.result.setText("Left " + countWand + " wands");

}

clientGUI.textNum.clear();

return;

}

}

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

}

public void listenForMessage() {

new Thread(new Runnable() {

@Override

public void run() {

String msgFromGame;

while (socket.isConnected()) {

try {

//GUI

clientGUI.textIp.setDisable(true);

clientGUI.textPort.setDisable(true);

clientGUI.connected.setDisable(true);

clientGUI.name.setDisable(true);

msgFromGame = bufferedReader.readLine();

if(msgFromGame.equals("server fill")) {

isRun = false;

socket.close();

clientGUI.result.setText("Server is Fill!");

//GUI

clientGUI.textIp.setDisable(false);

clientGUI.textPort.setDisable(false);

clientGUI.connected.setDisable(false);

clientGUI.name.setDisable(false);

Thread.currentThread().stop();

send(true);

return;

}

if(msgFromGame.equals("Next step You")) {

clientGUI.take.setDisable(false);

clientGUI.textNum.setDisable(false);

}

if(msgFromGame.equals("Next step Enemy")) {

clientGUI.take.setDisable(true);

clientGUI.textNum.setDisable(true);

}

try {

if(msgFromGame.substring(0,6).equals("SERVER")) msgFromGame = "";

} catch (Exception e) {}

if(msgFromGame.equals("S-R: You Lose!!!")) {

isRun = false;

socket.close();

clientGUI.result.setText(msgFromGame);

//GUI

clientGUI.textIp.setDisable(false);

clientGUI.textPort.setDisable(false);

clientGUI.connected.setDisable(false);

clientGUI.name.setDisable(false);

clientGUI.take.setDisable(false);

clientGUI.textNum.setDisable(false);

clientGUI.size.setText("20");

String place = "";

for (int i = 0; i < 20; i++) {

place += "| ";

}

clientGUI.place.setText(place);

countWand = 20;

Thread.currentThread().stop();

send(true);

return;

}

if(msgFromGame.equals("S-R: You Winner!!!")) {

isRun = false;

socket.close();

clientGUI.result.setText(msgFromGame);

//GUI

clientGUI.textIp.setDisable(false);

clientGUI.textPort.setDisable(false);

clientGUI.connected.setDisable(false);

clientGUI.name.setDisable(false);

clientGUI.take.setDisable(false);

clientGUI.textNum.setDisable(false);

clientGUI.size.setText("20");

String place = "";

for (int i = 0; i < 20; i++) {

place += "| ";

}

clientGUI.place.setText(place);

countWand = 20;

Thread.currentThread().stop();

send(true);

return;

}

clientGUI.result.setText(msgFromGame);

if(msgFromGame.contains("Server say:")) {

isRun = false;

socket.close();

clientGUI.result.setText("S-R: You Winner!!!");

//GUI

clientGUI.textIp.setDisable(false);

clientGUI.textPort.setDisable(false);

clientGUI.connected.setDisable(false);

clientGUI.name.setDisable(false);

clientGUI.take.setDisable(false);

clientGUI.textNum.setDisable(false);

clientGUI.size.setText("20");

String place = "";

for (int i = 0; i < 20; i++) {

place += "| ";

}

clientGUI.place.setText(place);

countWand = 20;

Thread.currentThread().stop();

send(true);

return;

}

if(isRun && !msgFromGame.equals("") && !msgFromGame.contains("left the game")

&& !msgFromGame.contains("Next step You") && !msgFromGame.contains("Next step Enemy")

&& !msgFromGame.equals("server fill")) {

String EnemyName = msgFromGame.substring(0,msgFromGame.length() - 3);

int minusValue = Integer.parseInt(msgFromGame.substring(msgFromGame.length() - 1));

clientGUI.result.setText(EnemyName + " take " + minusValue + " wands");

countWand -= minusValue;

String place = "";

for (int i = 0; i < countWand; i++) {

place += "| ";

}

//GUI

clientGUI.place.setText(place);

clientGUI.size.setText(String.valueOf(countWand));

clientGUI.result.setText("Left " + countWand + " wands");

}

try {

if(msgFromGame.substring(0, 6).equals("Starts")) isRun = true;

} catch (Exception e) {}

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

}

}

}).start();

}

public void closeEverything(Socket socket, BufferedReader bufferedReader, BufferedWriter bufferedWriter) {

try {

if(bufferedReader != null) {

bufferedReader.close();

}

if(bufferedWriter != null) {

bufferedWriter.close();

}

if(socket != null) {

socket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**ClientGUI.java**

package com.example.client;

import javafx.concurrent.Service;

import javafx.concurrent.Task;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.scene.text.Text;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.net.Socket;

import java.net.SocketTimeoutException;

public class ClientGUI {

private String ip;

private int port;

private Client client;

private String username;

@FXML

public TextField name;

@FXML

public TextField textNum;

@FXML

public TextField textIp;

@FXML

public TextField textPort;

@FXML

public Button take;

@FXML

public Button connected;

@FXML

public Text result;

@FXML

public Text place;

@FXML

public Text size;

@FXML

protected void takeClick() {

Service<Void> service = new Service<Void>() {

@Override

protected Task<Void> createTask() {

return new Task<Void>() {

@Override

protected Void call() throws Exception {

client.send(false);

return null;

}

};

}

};

service.start();

}

@FXML

protected void connectedClick() throws IOException {

if(name.getText().equals("") || textIp.getText().equals("")

|| textPort.getText().equals("")) {

return;

}

ip = textIp.getText();

try {

port = Integer.parseInt(textPort.getText());

} catch (NumberFormatException e) {

result.setText("Not correct port!");

}

try {

Socket socket = new Socket(ip, port);

username = name.getText();

client = new Client(socket, username, this);

Service<Void> service = new Service<Void>() {

@Override

protected Task<Void> createTask() {

return new Task<Void>() {

@Override

protected Void call() throws Exception {

client.listenForMessage();

client.sendName();

return null;

}

};

}

};

service.start();

} catch (IOException e) {

result.setText("Connected Failed!");

}

}

}

**Server**

**Server.java**

package org.example;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

public class Server {

private ServerSocket serverSocket;

public Server(ServerSocket serverSocket) {

this.serverSocket = serverSocket;

}

public void startServer() {

try {

while(!serverSocket.isClosed()) {

Socket socket = serverSocket.accept();

System.out.println("A new player has connected");

ClientHandler clientHandler = new ClientHandler(socket);

Thread thread = new Thread(clientHandler);

thread.start();

}

} catch (IOException e) {}

}

public void closeServerSocket() {

try {

if(serverSocket != null) {

serverSocket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args) throws IOException {

ServerSocket serverSocket = new ServerSocket(Integer.parseInt(args[0]));

Server server = new Server(serverSocket);

System.out.println("Server started...");

server.startServer();

}

}

**ClientHandler.java**

package org.example;

import java.io.\*;

import java.net.Socket;

import java.util.ArrayList;

import java.util.Random;

public class ClientHandler implements Runnable {

public static ArrayList<ClientHandler> clientHandlers = new ArrayList<>(2);

private Socket socket;

private BufferedReader bufferedReader;

private BufferedWriter bufferedWriter;

private String clientUsername;

private int countWand;

private boolean isFlag;

private int startedPlayer;

private int secondPlayer;

int currentWands;

public ClientHandler(Socket socket) {

try {

this.socket = socket;

this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

this.clientUsername = bufferedReader.readLine();

this.countWand = 20;

this.isFlag = false;

this.startedPlayer = 2;

this.secondPlayer = 2;

clientHandlers.add(this);

if(clientHandlers.size() > 2) {

broadcastMessageToOnePlayers(clientHandlers.get(2), "server fill");

clientHandlers.remove(2);

return;

}

System.out.println(clientHandlers.size() + "/2 players");

broadcastAllPlayers ("Server: " + clientUsername + " has entered the game!");

if(clientHandlers.size() == 2) {

System.out.println("NEW GAME INIT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

int started = Random();

int second;

broadcastAllPlayers("Starts the game " + clientHandlers.get(started).clientUsername);

clientHandlers.get(started).isFlag = true;

for (ClientHandler clientHand : clientHandlers) {

if(started == 0) second = 1;

else second = 0;

broadcastMessageToOnePlayers(clientHandlers.get(second), "Next step Enemy");

clientHand.startedPlayer = started;

clientHand.secondPlayer = second;

}

}

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

}

@Override

public void run() {

String messageFromClient;

while(socket.isConnected()) {

try {

messageFromClient = bufferedReader.readLine();

if(messageFromClient == null) {

removeClientHandler();

return;

}

else broadcastMessage(messageFromClient);

} catch (IOException e) {}

}

}

public void broadcastAllPlayers(String Message) {

try {

for (ClientHandler clientHandler : clientHandlers) {

clientHandler.bufferedWriter.write(Message);

clientHandler.bufferedWriter.newLine();

clientHandler.bufferedWriter.flush();

}

} catch (IOException e) {}

System.out.println(Message);

}

public void minusValue(String messageToSend) {

for (ClientHandler clientHand : clientHandlers) {

switch (messageToSend.substring(messageToSend.length() - 1)) {

case "1":

clientHand.countWand -= 1;

break;

case "2":

clientHand.countWand -= 2;

break;

case "3":

clientHand.countWand -= 3;

break;

default:

currentWands = clientHand.countWand;

clientHand.countWand = 999;

}

}

}

public int send(ClientHandler clientHandler, ClientHandler sec, String currentName, String messageToSend) {

if(clientHandler.countWand == 999) {

clientHandler.countWand = currentWands;

sec.countWand = currentWands;

return 0;

}

if(clientHandler.countWand == 1) {

try {

clientHandler.bufferedWriter.write("S-R: You Lose!!!");

clientHandler.bufferedWriter.newLine();

clientHandler.bufferedWriter.flush();

sec.bufferedWriter.write("S-R: You Winner!!!");

sec.bufferedWriter.newLine();

sec.bufferedWriter.flush();

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

System.out.println(clientHandler.clientUsername + " Lose!!!");

clientHandlers.clear();

System.out.println("Server clean!");

return 2;

}

if(clientHandler.countWand < 1) {

try {

sec.bufferedWriter.write("S-R: You Lose!!!");

sec.bufferedWriter.newLine();

sec.bufferedWriter.flush();

clientHandler.bufferedWriter.write("S-R: You Winner!!!");

clientHandler.bufferedWriter.newLine();

clientHandler.bufferedWriter.flush();

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

System.out.println(sec.clientUsername + " Lose!!!");

clientHandlers.clear();

System.out.println("Server clean!");

return 2;

}

else {

try {

clientHandler.bufferedWriter.write(currentName + ": " + messageToSend.substring(messageToSend.length() - 1));

clientHandler.bufferedWriter.newLine();

clientHandler.bufferedWriter.flush();

} catch (IOException e) {

closeEverything(socket, bufferedReader, bufferedWriter);

}

System.out.println("Left " + clientHandler.countWand + " wands");

return 1;

}

}

public void broadcastMessageToOnePlayers(ClientHandler clientHandler, String Message) {

try {

clientHandler.bufferedWriter.write(Message);

clientHandler.bufferedWriter.newLine();

clientHandler.bufferedWriter.flush();

} catch (IOException e) {}

}

public void broadcastMessage(String messageToSend) {

try {

if(messageToSend == null) {

return;

}

System.out.println(messageToSend); // out in server

if(clientHandlers.size() == 2) {

String currentName = messageToSend.substring(0,messageToSend.length() - 3);

ClientHandler clientHandler; // revievPlayer

ClientHandler sender; // senderPlayer

if(clientHandlers.get(startedPlayer).isFlag && currentName.equals(clientHandlers.get(startedPlayer).clientUsername)) {

int result;

minusValue(messageToSend);

sender = clientHandlers.get(startedPlayer);

clientHandler = clientHandlers.get(secondPlayer);

result = send(clientHandler, sender, currentName, messageToSend);

if(result == 1) {

System.out.println(currentName + " sends " + clientHandler.clientUsername);

broadcastMessageToOnePlayers(clientHandler, "Next step You");

broadcastMessageToOnePlayers(sender, "Next step Enemy");

clientHandlers.get(startedPlayer).isFlag = false;

clientHandlers.get(secondPlayer).isFlag = true;

}

if(result == 2) {

removeClientHandler();

}

}

if(clientHandlers.get(secondPlayer).isFlag && currentName.equals(clientHandlers.get(secondPlayer).clientUsername)) {

int result;

minusValue(messageToSend);

sender = clientHandlers.get(secondPlayer);

clientHandler = clientHandlers.get(startedPlayer);

result = send(clientHandler, sender, currentName, messageToSend);

if(result == 1) {

System.out.println(currentName + " sends " + clientHandler.clientUsername);

broadcastMessageToOnePlayers(clientHandler, "Next step You");

broadcastMessageToOnePlayers(sender, "Next step Enemy");

clientHandlers.get(secondPlayer).isFlag = false;

clientHandlers.get(startedPlayer).isFlag = true;

}

if(result == 2) {

removeClientHandler();

}

}

}

} catch (Exception e) {}

}

public void removeClientHandler() {

int n = clientHandlers.size();

clientHandlers.remove(this);

if(n != clientHandlers.size()) broadcastAllPlayers("Server say: " + clientUsername + " has left the game!");

}

public void closeEverything(Socket socket, BufferedReader bufferedReader, BufferedWriter bufferedWriter) {

removeClientHandler();

try {

if(bufferedReader != null) {

bufferedReader.close();

}

if(bufferedWriter != null) {

bufferedWriter.close();

}

if(socket != null) {

socket.close();

}

} catch (IOException e) {}

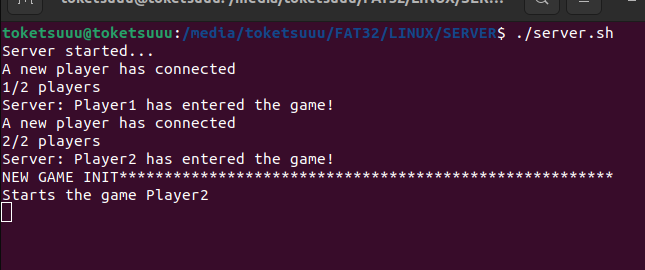
}

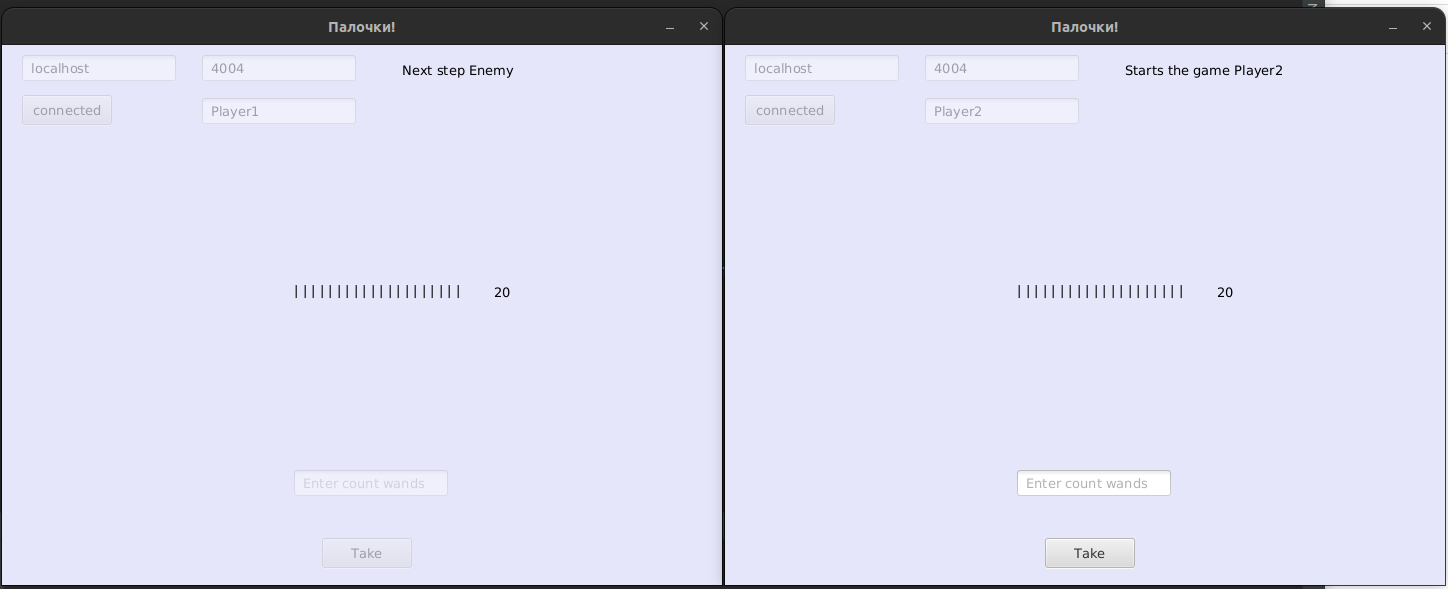
private int Random() {

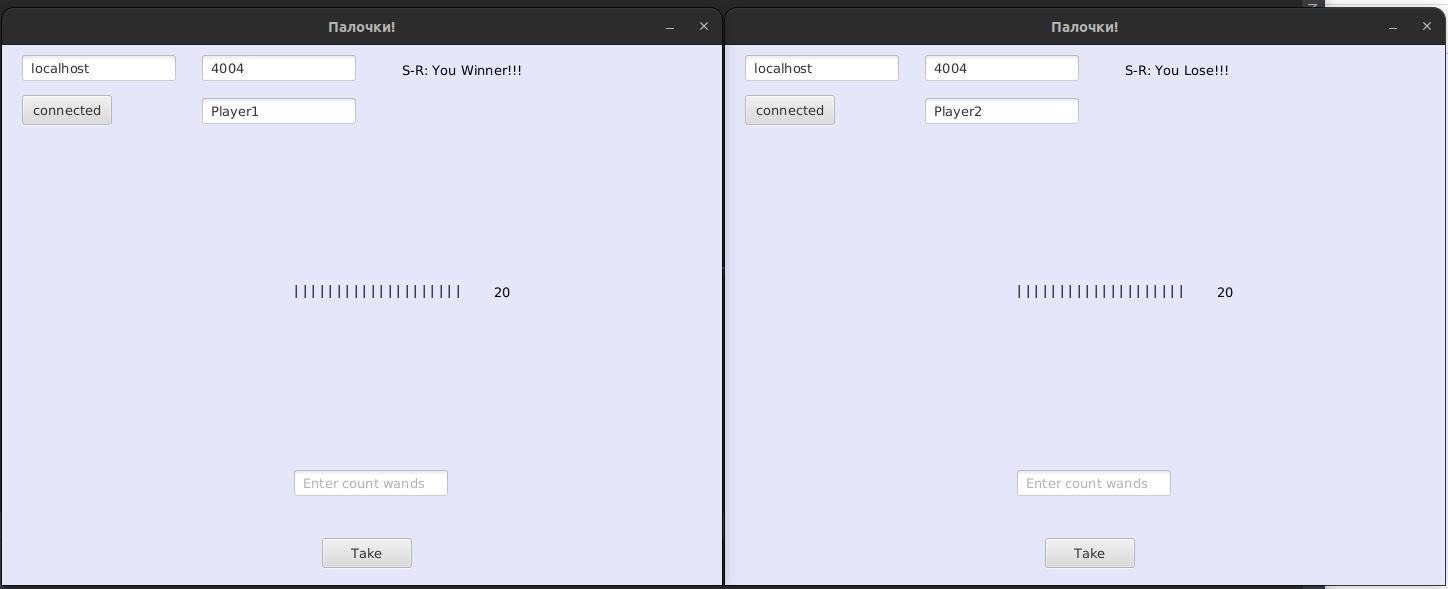
return new Random().nextInt(2);

}

}

****

****

****

****

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