Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №12

По дисциплине «СПП» за 6 семестр

Выполнил:

Студент группы ПО-3

Ковалёва А. И.

Кабачук Д. С.

Проверил:

Крощенко А. А.

Брест 2021

**Вариант 12**

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

**Задание 1**

**Текст программы:**

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.

**Server**

**Main**

package com.company;

import java.io.\*;

import java.net.Socket;

public class Main {

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

NetworkConnection nc = new NetworkConnection();

while (!nc.areTwoPlayers()){

Socket s = nc.setUsers();

System.out.println("Player " + (nc.users.indexOf(s)+1) + " connected");

}

startNewGame(nc);

}

public static void startNewGame(NetworkConnection nc){

System.out.println("Start game");

int score = 0;

while (true){

String str = nc.getCommand(0);

score = Integer.parseInt(str);

if(score <= 0) {

System.out.println("Win player one");

nc.sendMessage("Win player one", 1);

break;

}

nc.sendMessage(str, 1);

String str1 = nc.getCommand(1);

score = Integer.parseInt(str1);

if(score <= 0) {

System.out.println("Win player two");

nc.sendMessage("Win player two", 0);

break;

}

System.out.println(str1);

nc.sendMessage(str1, 0);

}

}

}

**NetworkConnection**

package com.company;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.LinkedList;

import java.util.List;

public class NetworkConnection {

ServerSocket serverSocket;

int PORT = 55555;

List<Socket> users = new LinkedList<>();

public NetworkConnection(){

try {

serverSocket = new ServerSocket(PORT, 2);

} catch (IOException e) {

e.printStackTrace();

}

}

public Socket setUsers(){

Socket s = null;

try {

s = serverSocket.accept();

users.add(s);

DataOutputStream out = new DataOutputStream(s.getOutputStream());

out.writeUTF(String.valueOf(users.indexOf(s)));

out.flush();

} catch (IOException e) {

e.printStackTrace();

}

return s;

}

public String getCommand(int user){

String command = null;

try {

DataInputStream in = new DataInputStream(users.get(user).getInputStream());

command = in.readUTF();

} catch (IOException e) {

e.printStackTrace();

}

return command;

}

public int getStep(int user){

int command = 0;

try {

DataInputStream in = new DataInputStream(users.get(user).getInputStream());

command = Integer.parseInt(in.readUTF());

} catch (IOException e) {

e.printStackTrace();

}

return command;

}

public void sendMessage(String message, int user){

try {

DataOutputStream out = new DataOutputStream(users.get(user).getOutputStream());

out.writeUTF(message);

out.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

boolean areTwoPlayers(){

return users.size() == 2;

}

public void sendMessageForAll(String message) {

for (Socket soket : users) {

DataOutputStream out = null;

try {

out = new DataOutputStream(soket.getOutputStream());

out.writeUTF(String.valueOf(message));

out.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}  
  
**User**

**Main**

package sample;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class Main extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

primaryStage.setTitle("LABA 12");

primaryStage.setScene(new Scene(root, 520, 300));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

**View**

package sample;

import javafx.concurrent.Service;

import javafx.concurrent.Task;

import javafx.concurrent.WorkerStateEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

public class View {

private Service<String> service;

@FXML

Label label;

@FXML

private Label messageLabel;

@FXML

private Button threeButton;

@FXML

private Button startButton;

@FXML

private Button twoButton;

@FXML

private Button stopButton;

@FXML

private Button oneButton;

@FXML

Game game;

void displayStart() {

startButton.setDisable(true);

game.setScore("20");

game.startGame();

if (game.player == 0) {

game.setMessage("You are " + (game.player + 1) + " player. Your step");

}

if (game.player == 1) {

setDisableStepButton(true);

game.setMessage("You are " + (game.player + 1) + " player. Wait your step");

// ожидаем хода перврго игрока

waitAnswer();

}

}

void displayStep(int step) {

setDisableStepButton(true);

game.setMessage("Wait your step");

int i = Integer.parseInt(game.getScore()) - step;

game.setScore(String.valueOf(i));

service = new Service<String>() {

@Override

protected Task<String> createTask() {

return new Task<String>() {

@Override

protected String call() throws Exception {

String s = "";

game.step(i);

if(game.isWinner(i)){

s = game.connection.getMessage();

return s;

}else {

s = "You win";

return s;

}

}

};

}

};

service.setOnSucceeded(new EventHandler<WorkerStateEvent>() {

@Override

public void handle(WorkerStateEvent event) {

setDisableStepButton(false);

game.setMessage("your step");

game.setScore(service.getValue());

}

});

service.start();

}

void waitAnswer() {

service = new Service<String>() {

@Override

protected Task<String> createTask() {

return new Task<String>() {

@Override

protected String call() throws Exception {

String s = game.connection.getMessage();

return s;

}

};

}

};

service.setOnSucceeded(new EventHandler<WorkerStateEvent>() {

@Override

public void handle(WorkerStateEvent event) {

setDisableStepButton(false);

game.setMessage("Your step");

game.setScore(service.getValue());

}

});

service.start();

}

void setDisableStepButton(Boolean flag) {

oneButton.setDisable(flag);

twoButton.setDisable(flag);

threeButton.setDisable(flag);

}

}

**Game**

package sample;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.StringProperty;

import javafx.concurrent.Service;

public class Game {

private Service<String> service;

Connection connection;

int player;

public Game() {

score.set("20");

connection = new Connection();

}

private StringProperty score = new SimpleStringProperty();

public final String getScore() {

return score.get();

}

public final void setScore(String str) {

score.set(str);

}

public StringProperty scoreProperty() {

return score;

}

private StringProperty message = new SimpleStringProperty();

public final String getMessage() {

return message.get();

}

public final void setMessage(String str) {

message.set(str);

}

public StringProperty messageProperty() {

return message;

}

public int startGame() {

player = Integer.parseInt(connection.getMessage());

if (player == 0) {

return 0;

}

return 1;

}

public void step(int i) {

connection.sendMessage(String.valueOf(i));

}

public boolean isWinner(int i) {

if (i <= 0) {

return false;

}

return true;

}

}

**Controller**

package sample;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

public class Controller extends View implements Initializable {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private Button threeButton;

@FXML

private Button startButton;

@FXML

private Button twoButton;

@FXML

private Button oneButton;

@FXML

private Label label;

@FXML

private Label messageLabel;

@FXML

Game game = new Game();

@FXML

void startClick(ActionEvent event) {

displayStart();

}

@FXML

void oneClick(ActionEvent event) {

displayStep(1);

}

@FXML

void twoClick(ActionEvent event) {

displayStep(2);

}

@FXML

void threeClick(ActionEvent event) {

displayStep(3);

}

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

super.game = game;

messageLabel.setText("Click \"Start\"");

label.setText("20");

game.scoreProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observableValue, String s, String t1) {

label.setText(t1);

}

});

game.messageProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observableValue, String s, String t1) {

messageLabel.setText(t1);

}

});

}

}

**Connection**

package sample;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.Socket;

public class Connection {

String URL = "localhost";

int PORT = 55555;

Socket socket;

DataOutputStream out;

DataInputStream in;

public Connection() {

try {

socket = new Socket(URL, PORT);

out = new DataOutputStream(socket.getOutputStream());

in = new DataInputStream(socket.getInputStream());

} catch (IOException e) {

e.printStackTrace();

}

}

public String getMessage() {

String temp = null;

try {

temp = in.readUTF();

} catch (IOException e) {

e.printStackTrace();

}

return temp;

}

public void sendMessage(String message) {

try {

out.writeUTF(message);

out.flush();

} catch (IOException e) {

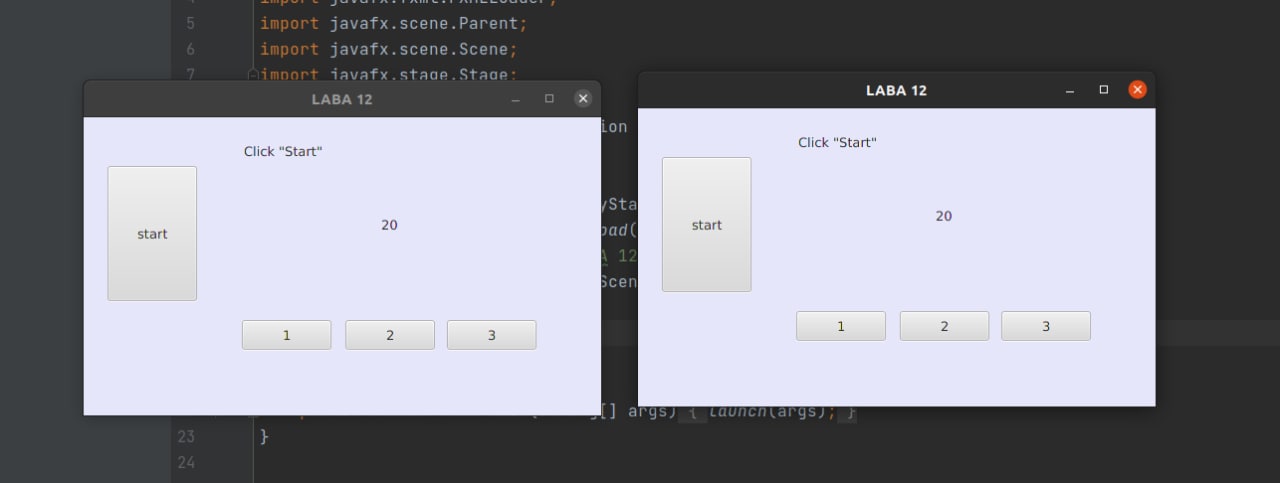
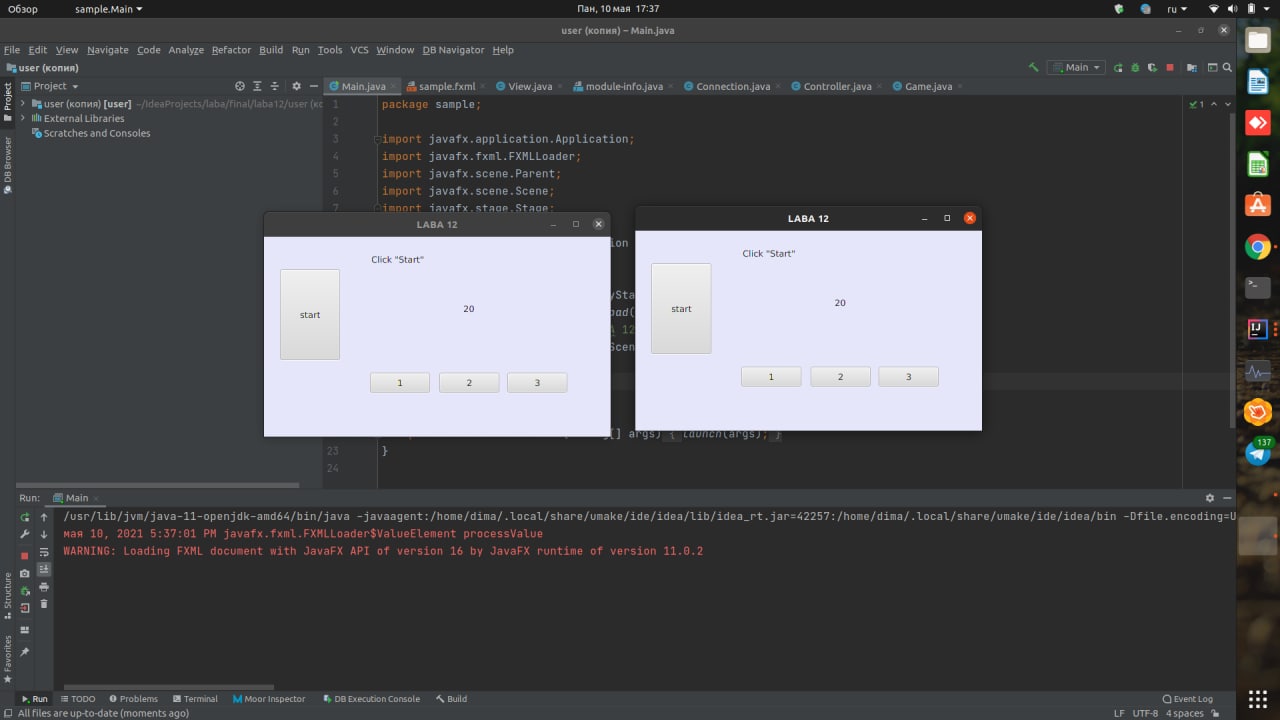
e.printStackTrace();

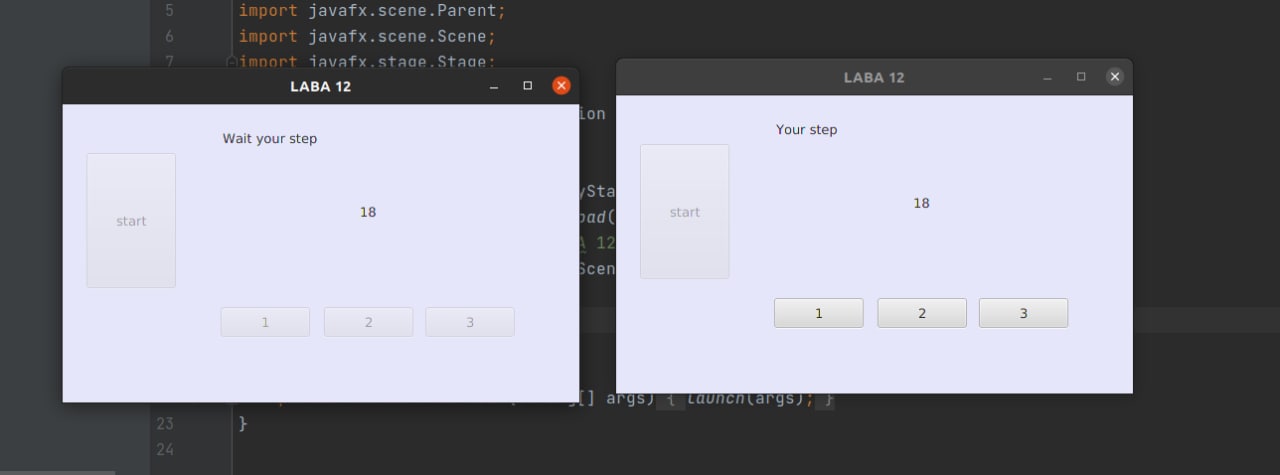
}

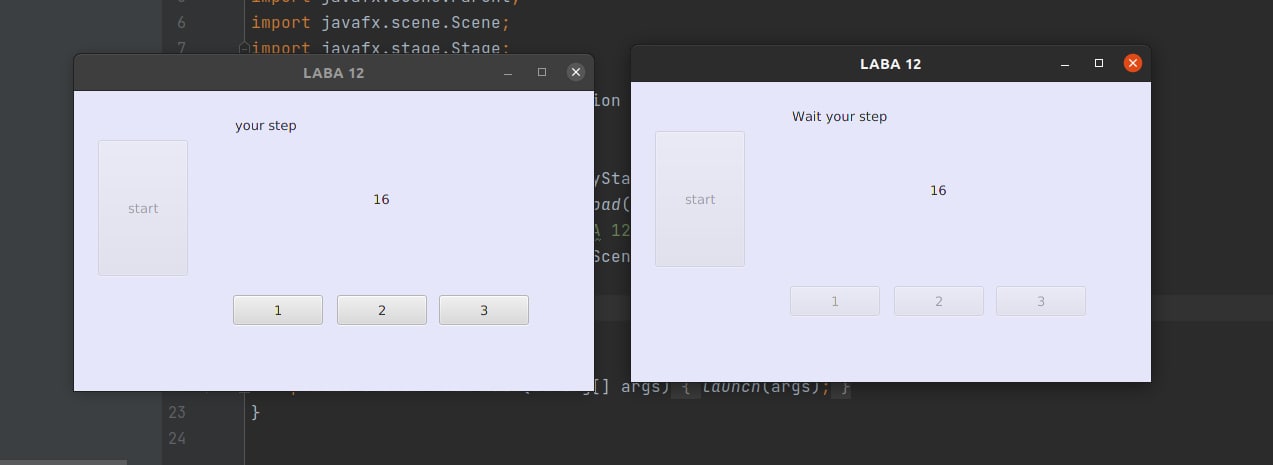
}

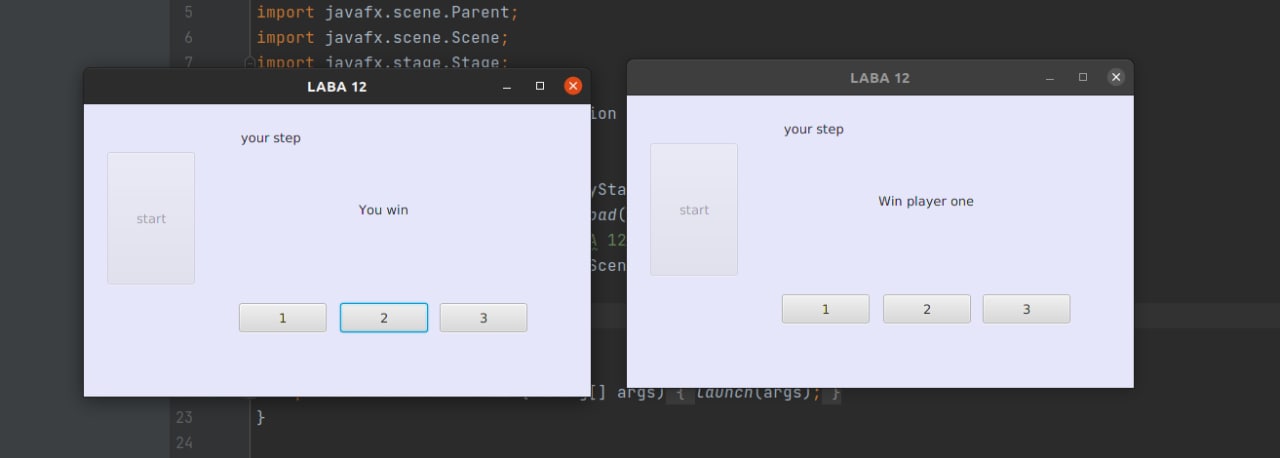
}

Результат выполнения:



****

****

****

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