**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

СОГЛАСОВАНО  
Старший преподаватель департамента программной инженерии факультета компьютерных наук  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Максименкова  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

УТВЕРЖДАЮ  
Академический руководитель образовательной программы «Программная инженерия»  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**ИГРА "PIXEL WARS"**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.508830-01 12 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ173

**../../Снимок%20экрана%202018-05-21%20в%2010.23.26.png**/ С. И. Ройтман /

«17» мая 2018 г.

**Москва 2018**

**УТВЕРЖДЕН**

**RU.17701729.508830-01 12 01-1-ЛУ**

**ИГРА "PIXEL WARS"**

**Текст программы**

**RU.17701729.508830-01 12 01-1**

**Листов 21**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Москва 2018**

**СОДЕРЖАНИЕ**

1. ТЕКСТ ПРОГРАММЫ 3
2. **ТЕКСТ ПРОГРАММЫ**

**Клиентская часть: Java, Android Studio**

Класс EnterActivity:

**package** com.example.morrison.pixelwars;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.os.StrictMode;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.view.View;  
**import** android.view.Window;  
**import** android.widget.TextView;  
  
  
**public class** EnterActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 requestWindowFeature(Window.***FEATURE\_INDETERMINATE\_PROGRESS***);  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_enter***);  
  
 StrictMode.ThreadPolicy policy = **new** StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
  
 **try** {  
 TextView errorMessage = findViewById(R.id.***errorText***);  
 errorMessage.append(InternetConnection.*connectToServer*());  
 }  
 **catch** (Exception e){  
 TextView errorMessage = findViewById(R.id.***errorText***);  
 errorMessage.append(InternetConnection.*connectToServer*());  
 }  
 }  
  
 *//Login button is pressed* **public void** login(View view) {  
 **try** {  
 TextView errorMessage = findViewById(R.id.***errorText***);  
 errorMessage.setText(**""**);  
 TextView nickname = findViewById(R.id.***Nickname***);  
 TextView password = findViewById(R.id.***Password***);  
 String[] msg = **new** String[2];  
 msg[0] = (nickname.getText()).toString();  
 msg[1] = (password.getText()).toString();  
  
 InternetConnection.*sendMessageToServer*(**"loginUser:"** + msg[0] + **","** + msg[1]);  
 String ans = InternetConnection.*getMessageFromServer*();  
 **if**(ans.equals(**"Ok"**))  
 {  
 finish();  
 startActivity(**new** Intent(EnterActivity.**this**, GameActivity.**class**));  
 }  
 **else throw new** Exception(**"User is not registered!"**);  
 }  
 **catch** (Exception e){  
 TextView errorMessage = findViewById(R.id.***errorText***);  
 errorMessage.append(**"Error!"**);  
 errorMessage.append(e.getMessage());  
 }  
 }  
  
 *//Register button is clicked* **public void** registration(View view){  
 startActivity(**new** Intent(EnterActivity.**this**, RegistrationActivity.**class**));  
 }  
}

Класс RegistrationActivity:

**package** com.example.morrison.pixelwars;  
  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.TextView;  
  
**import** java.util.regex.Matcher;  
**import** java.util.regex.Pattern;  
  
**public class** RegistrationActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_registration***);  
 }  
  
 **public void** register\_user(View view){  
 TextView errorText = findViewById(R.id.***errorText***);  
 errorText.setText(**""**);  
 TextView nickname = findViewById(R.id.***Nickname***);  
 TextView password = findViewById(R.id.***Password***);  
 String text\_nickname = nickname.getText().toString();  
 String text\_password = password.getText().toString();  
  
 *//Check nickname and password* **if**((CheckPattern(text\_nickname)) && ((CheckPattern(text\_password)))) {  
 InternetConnection.*sendMessageToServer*(**"registerUser:"** + text\_nickname + **","** + text\_password);  
 String ans = InternetConnection.*getMessageFromServer*();  
 **if**(ans.equals(**"Ok"**))  
 finish();  
 **else** errorText.append(**"Failed to register! This nickname is registered"**);  
  
 }  
 **else** {  
 TextView errorMessage = findViewById(R.id.***errorText***);  
 errorMessage.append(**"Nickname and password should consist only of letters and numbers!"**);  
 }  
 }  
  
 **public boolean** CheckPattern(String s){  
 Pattern pattern = Pattern.*compile*(**"\\w+"**);  
 Matcher matcher = pattern.matcher(s);  
 **return** matcher.matches();  
 }  
}

Класс GameActivity:

**package** com.example.morrison.pixelwars;  
  
**import** android.os.AsyncTask;  
**import** android.os.Bundle;  
**import** android.os.CountDownTimer;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.Log;  
**import** android.graphics.Color;  
**import** android.widget.TextView;  
**import** com.android.colorpicker.ColorPickerDialog;  
**import** com.android.colorpicker.ColorPickerSwatch;  
**import** com.bumptech.glide.Glide;  
**import** java.util.Timer;  
**import** java.util.TimerTask;  
  
**public class** GameActivity **extends** AppCompatActivity{  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 setContentView(R.layout.***activity\_game***);  
 **super**.onCreate(savedInstanceState);  
  
 **final** TouchImageView Field = findViewById(R.id.***Field***);  
 *//Request new field* **new** RefreshFieldTask().execute();  
  
 Field.setMaxZoom(18);  
  
 *//Add single tap listener* Responder responder = **new** Responder();  
 Field.addListener(responder);  
  
 getTimeLimit();  
 startRefreshFieldTimer(10);  
 }  
  
 *//Start timer with current time limit* **public void** getTimeLimit(){  
 InternetConnection.*sendMessageToServer*(**"getTimeLimit:"**);  
 String ans = InternetConnection.*getMessageFromServer*();  
 **if**(!ans.equals(**"No"**))  
 startTimer(5 - Integer.*parseInt*(ans));  
 }  
  
 *//Timer which request new field once in seconds* **public void** startRefreshFieldTimer(**int** seconds){  
 *//Declare the timer* Timer t = **new** Timer();  
 *//Set the schedule function and rate* t.scheduleAtFixedRate(**new** TimerTask() {  
 @Override  
 **public void** run() {  
 **new** RefreshFieldTask().execute();  
 }  
  
 }, seconds\*1000, seconds\*1000);  
 }  
  
 */\*//Calls menu when back is pressed  
 @Override  
 public boolean onKeyDown(int keyCode, KeyEvent event) {  
 if (keyCode == KeyEvent.KEYCODE\_BACK) {  
 startActivity(new Intent(GameActivity.this, MenuActivity.class));  
 return true;  
 }  
 return super.onKeyDown(keyCode, event);  
 }\*/  
  
 //Timer which shows users setPixel time limit* **public void** startTimer(**int** seconds){  
 **final** TextView mTimer = findViewById(R.id.***tv***);  
  
 **new** CountDownTimer(seconds\*1000, 1000) {  
 **public void** onTick(**long** millisUntilFinished) {  
 mTimer.setText(**"Осталось: "** + millisUntilFinished / 1000);  
 }  
 **public void** onFinish() {  
 mTimer.setText(**""**);  
 }  
 }  
 .start();  
 }  
  
 *//Load new field into TouchImageView* **public void** RefreshField(**byte**[] data){  
 **final** TouchImageView Field = findViewById(R.id.***Field***);  
 Glide  
  
 .*with*(GameActivity.**this**)  
 .asBitmap()  
 .load(data)  
 .into(Field);  
 }  
  
 *//Field single tap handler* **class** Responder **implements** SingleTapListener {  
 @Override  
 **public void** singleTapHandler(**float** x, **float** y) {  
 *//Check time limit* TextView mTimer = findViewById(R.id.***tv***);  
 String time = mTimer.getText().toString();  
 time = time.substring(time.indexOf(**':'**) + 1);  
 **if**(time.equals(**""**)) {  
 *//No limit* Log.*println*(Log.***ASSERT***, **"Click coordinates"**, x + **", "** + y);  
 **int** cell\_x = (**int**) (x / 11.0);  
 **int** cell\_y = (**int**) (y / 11.0);  
 Log.*println*(Log.***ASSERT***, **"Click cell: "**, + cell\_x + **", "** + cell\_y);  
 *//Pixel color picker* showColourPicker(cell\_x, cell\_y);  
 }  
 }  
 }  
  
 *//Color picker* **public void** showColourPicker(**final int** cell\_x, **final int** cell\_y) {  
 **final** ColorPickerDialog colorPickerDialog = **new** ColorPickerDialog();  
 colorPickerDialog.initialize(R.string.***color\_picker\_default\_title***,  
 **new int**[] {  
 Color.***BLACK***,  
 Color.***BLUE***,  
 Color.***CYAN***,  
 Color.***GRAY***,  
 Color.***GREEN***,  
 Color.***MAGENTA***,  
 Color.***RED***,  
 Color.***WHITE***,  
 Color.***YELLOW***,  
 }, Color.***BLACK***, 3, 2);  
 colorPickerDialog.setOnColorSelectedListener(**new** ColorPickerSwatch.OnColorSelectedListener() {  
 @Override  
 **public void** onColorSelected(**int** colour) {  
 InternetConnection.*sendMessageToServer*(**"setPixel:"** + cell\_x + **","** + cell\_y + **","** + colour);  
 String ans = InternetConnection.*getMessageFromServer*();  
 **if**(ans.equals(**"Ok"**)) {  
 startTimer(5);  
 **new** RefreshFieldTask().execute();  
 }  
 }  
 });  
  
 android.app.FragmentManager fm = **this**.getFragmentManager();  
 colorPickerDialog.setSelectedColor(0);  
 colorPickerDialog.show(fm, **"colorpicker"**);  
 }  
  
 *//AsyncTask which refresh field in background* **class** RefreshFieldTask **extends** AsyncTask<Void, Void, **byte**[]> {  
  
 @Override  
 **protected byte**[] doInBackground(Void... params) {  
 **try** {  
 **return** InternetConnection.*getFieldFromServer*();  
 } **catch** (Exception e) {  
 Log.*println*(Log.***ASSERT***, **"AsyncTask: RefreshField"**, e.getMessage());  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(**byte**[] result)  
 {  
 *// Call activity method with results* RefreshField(result);  
 }  
 }  
}

Класс InternetConnection:

**package** com.example.morrison.pixelwars;  
  
**import** android.util.Log;  
**import** java.io.BufferedReader;  
**import** java.io.DataInputStream;  
**import** java.io.DataOutputStream;  
**import** java.io.InputStreamReader;  
**import** java.net.InetAddress;  
**import** java.net.InetSocketAddress;  
**import** java.net.Socket;  
**import** java.net.SocketAddress;  
**import** java.nio.charset.Charset;  
  
**public class** InternetConnection {  
  
 **public static** BufferedReader *inFromUser*;  
 **public static** DataInputStream *inFromServer*;  
 **public static** DataOutputStream *outToServer*;  
  
 **public static** String connectToServer(){  
 **try** {  
 *//create input stream  
 inFromUser* = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 *//create client socket, connect to server* **int** port = 11198;  
 String host = **"0.tcp.ngrok.io"**;  
 Socket clientSocket = **new** Socket();  
 InetAddress addr = InetAddress.*getByName*(host);  
 SocketAddress sockaddr = **new** InetSocketAddress(addr, port);  
 clientSocket.connect(sockaddr);  
 *//create output stream attached to socket  
 outToServer* = **new** DataOutputStream(clientSocket.getOutputStream());  
 *//create input stream attached to socket  
 inFromServer* = **new** DataInputStream(clientSocket.getInputStream());  
 **return "Connected!"**;  
 }  
 **catch** (Exception e){  
 **return "Connection error!"**;  
 }  
 }  
  
 **public static void** sendMessageToServer(String msg){  
 **try** {  
 Thread.*sleep*(250);  
 **byte**[] data = msg.getBytes(Charset.*forName*(**"Unicode"**));  
 *outToServer*.write(data);  
 *outToServer*.flush();  
 }  
 **catch**(Exception e){  
  
 }  
 }  
  
 **public static** String getMessageFromServer(){  
 String ans = **""**;  
 **try** {  
 **byte**[] tmp = **new byte**[32];  
 **int** lenght = *inFromServer*.read(tmp);  
 String str = **new** String(tmp, 0, lenght);  
 ans = str;  
 }  
 **catch** (Exception e){  
 ans = **"Error"**;  
 }  
 **return** ans;  
 }  
  
 **public static byte**[] getFieldFromServer(){  
 **byte**[] bytes;  
 **try** {  
 *//Посылаем запрос на сервер  
 sendMessageToServer*(**"getField:"**);  
 Thread.*sleep*(100);  
 Log.*println*(Log.***ASSERT***, **"getFieldFromServer"**, **"Message sent"**);  
 String size = *getMessageFromServer*();  
 Log.*println*(Log.***ASSERT***, **"Image size"**, size);  
 bytes = **new byte**[Integer.*parseInt*(size)];  
 *//Thread.sleep(200);  
 inFromServer*.readFully(bytes);  
 Log.*println*(Log.***ASSERT***, **"getFieldFromServer"**, **"All bytes read"**);  
 }  
 **catch** (Exception e){  
 Log.*println*(Log.***ASSERT***, **"getFieldFromServer"**, **"Exception!"**);  
 bytes = **new byte**[1];  
 }  
 **return** bytes;  
 }  
}

Класс TouchImageView:

Импортирован

**Серверная часть: C# .NET**

Класс Program:

using System;  
using MyLib;  
using System.Threading;  
  
namespace TCPServer  
{  
    class Program  
    {  
        static Server server; // сервер  
        static Thread listenThread; // потока для прослушивания  
        static void Main(string[] args)  
        {  
            try  
            {  
                server = new Server();  
                listenThread = new Thread(new ThreadStart(server.Listen));  
                listenThread.Start(); //старт потока  
            }  
            catch (Exception ex)  
            {  
                server.Disconnect();  
                Console.WriteLine(ex.Message);  
            }  
        }  
    }  
}

Класс Server:

using System;  
using System.Collections.Generic;  
using System.Net;  
using System.Net.Sockets;  
using System.Text;  
using System.Threading;  
using System.Drawing;  
using System.IO;  
using System.Drawing.Imaging;  
using System.Xml.Serialization;  
  
namespace MyLib  
{  
    public class Server  
    {  
        static TcpListener tcpListener; // сервер для прослушивания  
        List<Client> clients = new List<Client>(); // все подключения  
  
        //Конструктор  
        public Server()  
        {  
            CreateField();  
        }  
  
        //Обработка запросов клиентов  
        public void Processing(int id, string msg)  
        {  
            int pos\_cmd = msg.IndexOf(':');  
            string cmd = msg.Remove(pos\_cmd);  
            string data = msg.Remove(0, pos\_cmd+1);  
            //Обрабатываем варианты запросов  
            switch (cmd)  
            {  
                case "getField":  
                    //Посылаем пользователю поле  
                    SendFieldToUser(id);  
                    break;  
                case "setPixel":  
                    int limit = GetTimeLimit(id);  
                    if ((limit >= 5) || (limit < 0))  
                    {  
                        SendMessageToUser("Ok", id);  
                        UpdateUserTimeLimit(id);  
                        //Получаем координаты пикселя  
                        string[] coord = data.Split(',');  
                        int x = int.Parse(coord[0]) \* 10;  
                        int y = int.Parse(coord[1]) \* 10;  
                        int col = int.Parse(coord[2]);  
                        SetPixelColor(x, y, Color.FromArgb(col));  
                    }  
                    else  
                        Console.WriteLine($"{clients[id].UserName}: timeLimit!");  
                    break;  
                case "registerUser":  
                    string[] user\_information = data.Split(',');  
                    //Если пользователь с таким именем не зарегистрирован  
                    if (!File.Exists(@"users\" + user\_information[0] + ".xml"))  
                    {  
                        clients[id].AddUserInfo(user\_information[0], user\_information[1]);  
                        Tools.SerializeClient(clients[id]);  
                        SendMessageToUser("Ok", id);  
                    }  
                    else  
                        SendMessageToUser("No", id);  
                    break;  
                case "loginUser":  
                    string[] login\_information = data.Split(',');  
                    if (LoginUser(login\_information[0], login\_information[1]))  
                    {  
                        SendMessageToUser("Ok", id);  
                        clients[id].AddUserInfo(login\_information[0], login\_information[1]);  
                    }  
                    else  
                        SendMessageToUser("No", id);  
                    break;  
                case "getTimeLimit":  
                    int timeLimit = GetTimeLimit(id);  
                    if ((timeLimit > 0) && (timeLimit < 5))  
                        SendMessageToUser(timeLimit.ToString(), id);  
                    else  
                        SendMessageToUser("No", id);  
                        break;  
            }  
        }  
  
        //Создание поля   
        private void CreateField()  
        {  
            int size = 1000;  
            int block\_size = 10;  
            Bitmap b = new Bitmap(size, size);  
            using (Graphics g = Graphics.FromImage(b))  
            {  
                g.Clear(Color.White);  
                Pen pen = new Pen(Color.Black);  
                pen.Width = 1;  
                for (int i = 0; i <= size/block\_size; i++)  
                {  
                    g.DrawLine(pen, i \* block\_size, 0, i \* block\_size, size);  
                    g.DrawLine(pen, 0, i \* block\_size, size, i \* block\_size);  
                }  
            }  
            b.Save(@"field.png", ImageFormat.Png);  
            b.Dispose();  
        }  
  
        //Изменение цвета пикселя  
        private void SetPixelColor(int x, int y, Color color)  
        {  
            try  
            {  
                int block\_size = 10;  
                Bitmap b = new Bitmap(@"field.png");  
                using (Graphics g = Graphics.FromImage(b))  
                {  
                    Pen colPen = new Pen(color);  
                    Rectangle rect = new Rectangle(x+1, y+1, block\_size-1, block\_size-1);  
                    SolidBrush brush = new SolidBrush(color);  
                    g.FillRectangle(brush, rect);  
                    g.Dispose();  
                }  
                b.Save(@"tmp.png", ImageFormat.Png);  
                b.Dispose();  
                File.Delete(@"field.png");  
                File.Move(@"tmp.png", @"field.png");  
            }  
            catch (Exception e)  
            {  
                Console.WriteLine(e);  
            }  
        }  
  
        //Вход в игру  
        private bool LoginUser(string nickname, string password)  
        {  
            Client client;  
            XmlSerializer serializer = new XmlSerializer(typeof(Client));  
  
            if (File.Exists(@"users\" + nickname + ".xml"))  
            {  
                using (StreamReader reader = new StreamReader(@"users\" + nickname + ".xml"))  
                {  
                    client = (Client)serializer.Deserialize(reader);  
                }  
                return (client.Password.Equals(password));  
            }  
            return false;  
        }  
  
        //Обновление ограничения  
        private void UpdateUserTimeLimit(int id)  
        {  
            clients[id].timeLimit = DateTime.Now;  
            Tools.SerializeClient(clients[id]);  
        }  
  
        //Возвращает кол-во секунд прошеднее с последнего изменения цвета пикселя пользователем  
        private int GetTimeLimit(int id)  
        {  
            Client client = Tools.DeserializeClient(clients[id].UserName);  
            var diff = (int)DateTime.Now.Subtract(client.timeLimit).TotalSeconds;  
            return diff;  
        }  
  
        //Отправка сообщения клиенту  
        private void SendMessageToUser(string message, int id)  
        {  
            byte[] data = Encoding.UTF8.GetBytes(message);  
            clients[id].Stream.Write(data, 0, data.Length);  
        }  
  
        // Отправка текущего состояния поля клиенту  
        private void SendFieldToUser(int id)  
        {  
            Bitmap b = (Bitmap)Image.FromFile(@"field.png");  
            byte[] bytes = Tools.ImageToByteArray(b, ImageFormat.Png);  
            Console.WriteLine("Field size: " + bytes.Length);  
            //Отправляем размер поля  
            SendMessageToUser((bytes.Length).ToString(), id);  
            //Отправляем поле  
            clients[id].Stream.Write(bytes, 0, bytes.Length);  
            b.Dispose();  
        }  
  
        // Прослушивание входящих подключений  
        public void Listen()  
        {  
            try  
            {  
                tcpListener = new TcpListener(IPAddress.Any, 8888);  
                tcpListener.Start();  
                Console.WriteLine("Сервер запущен. Ожидание подключений...");  
  
                while (true)  
                {  
                    TcpClient tcpClient = tcpListener.AcceptTcpClient();  
  
                    Client client = new Client(tcpClient, this);  
                    Thread clientThread = new Thread(new ThreadStart(client.Process));  
                    clientThread.Start();  
                }  
            }  
            catch (Exception ex)  
            {  
                Console.WriteLine(ex.Message);  
                Disconnect();  
            }  
        }  
  
        //Добавить клиента  
        internal void AddConnection(Client client)  
        {  
            clients.Add(client);  
            client.Id = clients.Count - 1;  
        }  
  
        // Удалить клиента  
        internal void RemoveConnection(int id)  
        {  
            // получаем по id закрытое подключение  
            // и удаляем его из списка подключений  
            if (clients[id] != null)  
                clients.RemoveAt(id);  
        }  
  
        // Отключение всех клиентов  
        public void Disconnect()  
        {  
            //остановка сервера  
            tcpListener.Stop();  
  
            //отключение клиентов  
            for (int i = 0; i < clients.Count; i++)  
                clients[i].Close();   
            //завершение процесса  
            Environment.Exit(0);  
        }  
    }  
}

Класс Client:

using System;  
using System.Net.Sockets;  
using System.Text;  
using System.Xml;  
using System.Xml.Schema;  
using System.Xml.Serialization;  
  
namespace MyLib  
{  
    public class Client: IXmlSerializable  
    {  
        //XML сериализация  
        public void WriteXml(XmlWriter writer)  
        {  
            writer.WriteAttributeString("password", password);  
            writer.WriteAttributeString("timeLimit", timeLimit.ToString());  
        }  
  
        //XML десериализация  
        public void ReadXml(XmlReader reader)  
        {  
            password = reader.GetAttribute("password");  
            string tmp = reader.GetAttribute("timeLimit");  
            timeLimit = Convert.ToDateTime(reader.GetAttribute("timeLimit"));  
        }  
  
        public XmlSchema GetSchema()  
        {  
            return (null);  
        }  
  
  
        protected internal int Id { get; set; }  
        protected internal NetworkStream Stream { get; private set; }  
        public string UserName { get { return userName; } }  
        string userName;  
        public string Password { get { return password; } }  
  
        string password;  
        public DateTime timeLimit;  
        protected internal TcpClient client;  
        protected internal Server server; // объект сервера  
  
        public Client() { }  
        //Конструктор  
        public Client(TcpClient tcpClient, Server server)  
        {  
            userName = "Client" + Id;  
            client = tcpClient;  
            this.server = server;  
            server.AddConnection(this);  
        }  
  
        //Добавляем информацию о пользователе  
        public void AddUserInfo(string userName, string password)  
        {  
            this.userName = userName;  
            this.password = password;  
        }  
  
        //Обработка клиента  
        public void Process()  
        {  
            try  
            {  
                Stream = client.GetStream();  
                string message = UserName + " вошел в сеть";  
                Console.WriteLine(message);  
                // в бесконечном цикле получаем сообщения от клиента  
                while (true)  
                {  
                    try  
                    {  
                        message = GetMessage();  
                        Console.WriteLine(String.Format("{0}: {1}", UserName, message));  
                        server.Processing(Id, message);  
                    }  
                    catch(Exception e)  
                    {  
                        Console.WriteLine(e.Message);  
                        message = String.Format("{0}: покинул чат", UserName);  
                        Console.WriteLine(message);  
                        break;  
                    }  
                }  
            }  
            catch (Exception e)  
            {  
                Console.WriteLine(e.Message);  
            }  
            finally  
            {  
                // в случае выхода из цикла закрываем ресурсы  
                server.RemoveConnection(this.Id);  
                Close();  
            }  
        }  
  
        // чтение входящего сообщения и преобразование в строку  
        private string GetMessage()  
        {  
            byte[] data = new byte[64]; // буфер для получаемых данных  
            StringBuilder builder = new StringBuilder();  
            int bytes = 0;  
            do  
            {  
                bytes = Stream.Read(data, 0, data.Length);  
                builder.Append(Encoding.Unicode.GetString(data, 0, bytes));  
            }  
            while (Stream.DataAvailable);  
            string s = (builder.ToString()).Substring(1);  
            return s;  
        }  
  
        // закрытие подключения  
        protected internal void Close()  
        {  
            if (Stream != null)  
                Stream.Close();  
            if (client != null)  
                client.Close();  
        }  
    }  
}

Класс Tools:

using System.Drawing;  
using System.Drawing.Imaging;  
using System.IO;  
using System.Xml.Serialization;  
  
namespace MyLib  
{  
    public static class Tools  
    {  
        //Конвертируем картинку в набор байт  
        public static byte[] ImageToByteArray(Image image, ImageFormat format)  
        {  
            using (MemoryStream ms = new MemoryStream())  
            {  
                image.Save(ms, format);  
                return ms.ToArray();  
            }  
        }  
  
        //Записываем данные пользователя в XML файл  
        public static void SerializeClient(Client client)  
        {  
            XmlSerializer serializer = new XmlSerializer(typeof(Client));  
  
            using (StreamWriter writer = new StreamWriter(@"users\" + client.UserName + ".xml"))  
            {  
                serializer.Serialize(writer, client);  
            }  
        }  
  
        //Считываем данные пользователя из XML файла  
        public static Client DeserializeClient(string userName)  
        {  
            Client cl = new Client();  
            XmlSerializer serializer = new XmlSerializer(typeof(Client));  
            string path = @"users\" + userName + ".xml";  
  
            if (File.Exists(path))  
            {  
                using (StreamReader reader = new StreamReader(path))  
                {  
                    cl = (Client)serializer.Deserialize(reader);  
                }  
            }  
            return cl;  
        }  
    }  
}

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |