МИНИСТЕРСТВО ОБРАЗОВАНИЯ

И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

Лабораторная работа № 5

по курсу «Объектно-ориентированное программирование»

«Классы, объекты, наследование в С#»

13 ВАРИАНТ

Выполнили:

студенты гр. КТбо2-6

Шевченко А. В.

Проверил:

Быстров Э. А.

Таганрог 2020

**Цель работы:** Ознакомление с основами объектно-ориентированного программирования на С#.

1. Вариант задания №13

Разработать программу, способную подобрать для человека по его данным пару из базы данных. Критериями должны быть соответствие по знаку зодиака, по совместимости имён, так же должен быть учёт местоположения и возраста. Базу данных необходимо заполнить с помощью интернет рессурсов.

1. Спецификация классов с комментариями.

Был написан базовый Human с полем для хранения пароля и свойством для логина, метод хеширования и метод проверяющий совпадения с паролем введённой строки с полем. Так же были перегружены сравнения. На основе это класса были разработаны два класса Admin и Client. Первый ничего не привносит, позволяя “создать” класс типа Human. Класс Client имеет свойства полное имя, пол, день рождения, возраст, город проживания, номер телефона, проверяемые с помощью статических методов класса StandartView. Поля класса Client полное имя, пол, знак зодиака и число, полученное комбинаторикой имени. Для получения данных клиента был разработан класс Result с соответствующими полями который используется в классе User который делает запрос на интернет-ресурсе и создаёт класс Result из которого извлекается массив строк. Класс DataBese имеет три приватных поля хранящих администраторов и клиентов, не сортированных и сортированных по городам и полу. Так же методы возвращающий лист клиентов, подходящих пользователю. Так же был разработан класс взаимодействующий с клиентом и админом.

1. Диаграмма классов

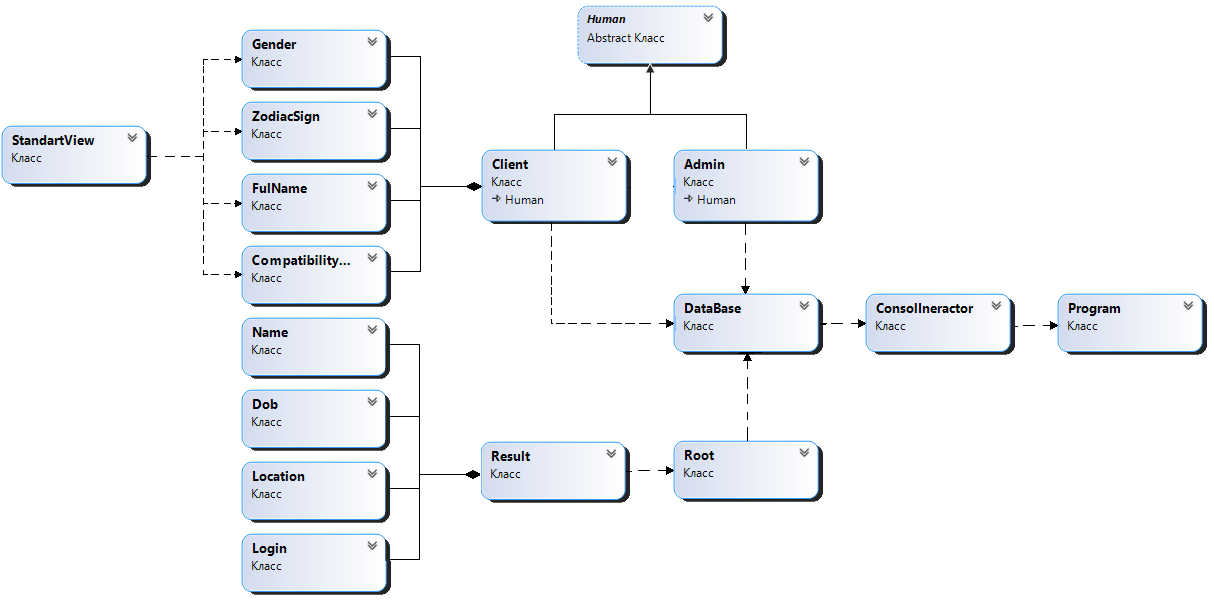


Рисунок 1 – диаграмма классов

1. Листинг прогграмы

Human.cs

using System.Linq;

using System.Security.Cryptography;

using System.Text;

namespace laba\_5

{

public abstract class Human

{

private readonly byte[] \_password;

public string Login { get; }

public Human(string login, string password, bool isStandertView = false)

{

if(!isStandertView) StandartView.LoginEr(login);

Login = login;

\_password = HachPassword(password);

}

public bool IsMyPassword(string password)

{

var temp = HachPassword(password);

return \_password.SequenceEqual(temp);

}

// он сказал сделать статическим

private static byte[] HachPassword(string password)

{

byte[] tmpSource;

tmpSource = ASCIIEncoding.ASCII.GetBytes(password);

return new MD5CryptoServiceProvider().ComputeHash(tmpSource);

}

public static bool operator ==(Human c1, Human c2) => c1.Login == c2.Login && c1.\_password.SequenceEqual(c2.\_password);

public static bool operator !=(Human c1, Human c2) => !(c1 == c2);

public override bool Equals(object obj)

{

var c2 = obj as Human;

return this == c2;

}

// не понимаю - зачем перегружать

public override int GetHashCode() => System.HashCode.Combine(Login + \_password);

}

}

Admin.cs:

namespace laba\_5

{

public class Admin : Human

{

public Admin(string login, string passward, bool isStandertView = false) : base(login, passward, isStandertView) { }

}

}

Client.cs:

using System;

namespace laba\_5

{

public class Client : Human

{

public FulName MyFulName { get; }

public DateTime MyBirthday { get; }

public Gender MySex { get; }

public string MyCity { get; }

public string MyPhoneNumber { get; }

public ZodiacSign MyZodiacSign { get; }

public СompatibilityByName MyCompatibilityByName { get; }

public int MyAge

{

get

{

// там оказывается ещё сложнее из-за 29 февраля

var age = DateTime.Now.Year - MyBirthday.Year;

if (DateTime.Now.Month < MyBirthday.Month || (DateTime.Now.Month == MyBirthday.Month && DateTime.Now.Day < MyBirthday.Day))

{

age--;

}

if (age < 0)

{

throw new System.Exception("Человек ещё не родился");

}

return age;

}

}

public Client(string login, string password, string myFulName, string sex, string myBirthday,

string myCity, string myPhoneNumber, bool isStandartView = false) : base(login, password, isStandartView)

{

MyFulName = new FulName(myFulName, isStandartView);

MyBirthday = StandartView.ConverteStringToDate(myBirthday);

MyCity = (isStandartView) ? myCity : StandartView.ConverteToStandartString(myCity);

MyPhoneNumber = (isStandartView) ? myPhoneNumber : StandartView.ConverteToStandartPhoneNumber(myPhoneNumber);

MySex = new Gender(sex);

MyZodiacSign = new ZodiacSign(MyBirthday);

MyCompatibilityByName = new СompatibilityByName(MyFulName.FirstName);

}

public Client(string[] str, bool isStandartView = false) : this(str[0], str[1], str[2], str[3], str[4], str[5], str[6], isStandartView) { }

public int ScoreIsPaer(in Client person) => (MySex != person.MySex && MyCity == person.MyCity) ? ScoreIsPaerWithoutSexAndLocalization(person) : 0;

public int ScoreIsPaerWithoutSexAndLocalization(in Client person, int maxAgeDiferent = 5)

{

if (Math.Abs(MyAge - person.MyAge) > maxAgeDiferent)

{

return 0;

}

var result = MyCompatibilityByName.IsAPaer(person.MyCompatibilityByName);

result += MyZodiacSign.IsAPaer(person.MyZodiacSign);

return result\*5;

}

}

}

StandartView.cs:

using System;

using System.Collections.Generic;

using System.Linq;

namespace laba\_5

{

public class StandartView

{

public static string[] ToStringArray(string str, char separator = ' ')

=> str.Split(new char[] { separator }, StringSplitOptions.RemoveEmptyEntries);

// ох уж эти многосоставные имена да фамилии и города...

public static string ConverteToStandartString(string str)

{

var strArr = ToStringArray(str, '-');

str = null;

for (var i = 0; i < strArr.Length; i++)

{

strArr[i] = ConverteToStandartWord(strArr[i]);

str += (strArr.Length - i == 1) ? strArr[i] : strArr[i] + '-';

}

return str;

}

public static string ConverteToStandartWord(string word)

{

WordEr(word);

// ну это его идея использовать перечисления

return word.Substring(0, 1).ToUpper() + word[1..].ToLower();

}

public static bool IsAWord(string word) => word.ToCharArray().All(char.IsLetter) && word.Length > 0;

public static void WordEr(string word)

{

if (!IsAWord(word))

{

throw new System.Exception($"ошибка в {word} это вырожение не имеет смысла");

}

}

public static bool IsLogin(string login)

{

if(login.Length > 0)

{

foreach (var c in login)

{

if (!(char.IsLetterOrDigit(c) || c == '@' || c == '\_' || c == '.'))

{

return false;

}

}

return true;

}

return false;

}

public static void LoginEr(string login)

{

if (!IsLogin(login))

{

throw new System.Exception($"ошибка в {login} это не формат логина");

}

}

// несколько не понял, как это работает, пришлось просто переделать, неосознавая(

static public string ConverteToStandartPhoneNumber(string phoneNumber)

{

PhoneNumberEr(phoneNumber);

phoneNumber = new System.Text.RegularExpressions.Regex(@"\D").Replace(phoneNumber, string.Empty);

if (phoneNumber.Length == 11)

{

phoneNumber = Convert.ToInt64(phoneNumber).ToString("#(###)###-##-##");

}

else

{

throw new System.Exception($"{phoneNumber} не поддерживаемый формат");

}

return (!phoneNumber.StartsWith('8')) ? '+' + phoneNumber : phoneNumber;

}

public static bool IsPhoneNumber(string phoneNumber)

{

var count = 0;

foreach (var c in phoneNumber)

{

if (char.IsDigit(c))

{

count++;

}

else if (!(c == '+' || c == '(' || c == '-' || c == ')' || c == ' '))

{

return false;

}

}

return count == 11;

}

public static void PhoneNumberEr(string phoneNumber)

{

if (!IsPhoneNumber(phoneNumber))

{

throw new System.Exception($"{phoneNumber} не номер телефона");

}

}

public static DateTime ConverteStringToDate(string str)

{

var strArr = ToStringArray(str);

var temp = new string[3];

if (strArr.Length == 1)

{

strArr = ToStringArray(str, '.');

}

for (var i = 0; i < temp.Length; i++)

{

temp[i] = strArr[i];

}

var mounth = IsAWord(temp[1]) ? Mounth(temp[1]) : int.Parse(temp[1]);

return new DateTime(int.Parse(temp[2]), mounth, int.Parse(temp[0]));

}

private static int Mounth(string mounth)

{

var temp = \_mounth[ConverteToStandartWord(mounth)];

return (temp != 0) ? temp : throw new Exception("Такого месяца нет");

}

private readonly static SortedDictionary<string, int> \_mounth = new SortedDictionary<string, int>

{

["Января"] = 1,

["Февраля"] = 2,

["Марта"] = 3,

["Апреля"] = 4,

["Мая"] = 5,

["Июня"] = 6,

["Июля"] = 7,

["Августа"] = 8,

["Сентября"] = 9,

["Октября"] = 10,

["Ноября"] = 11,

["Декабря"] = 12

};

}

}

FulName.cs:

namespace laba\_5

{

public class FulName

{

public string FirstName { get; }

public string SecondName { get; }

public FulName(string secondName, string firstName, bool isStandartView = false)

{

FirstName = (isStandartView) ? firstName: StandartView.ConverteToStandartString(firstName);

SecondName = (isStandartView) ? secondName : StandartView.ConverteToStandartString(secondName);

}

public FulName(string fulName, bool isStandartView = false)

{

var sfm = StandartView.ToStringArray(fulName);

if (sfm.Length > 2)

{

throw new System.Exception($"{fulName} нарушение формата ввода");

}

FirstName = (isStandartView) ? sfm[1] : StandartView.ConverteToStandartString(sfm[1]);

SecondName = (isStandartView) ? sfm[0] : StandartView.ConverteToStandartString(sfm[0]);

}

public override string ToString() => $"{SecondName} {FirstName}";

}

}

Gender.cs:

using System;

using System.Collections.Generic;

namespace laba\_5

{

public class Gender

{

public enum Sex

{

female,

masculine

}

public Sex MySex { get; }

private static readonly SortedDictionary<string, int> \_sex = new SortedDictionary<string, int>

{

["f"] = 0, ["female"] = 0, ["fem"] = 0, ["ж"] = 0, ["женский"] = 0, ["жен"] = 0, ["m"] = 1, ["masculine"] = 1, ["male"] = 1, ["м"] = 1, ["мужской"] = 1, ["муж"] = 1

};

public Gender(string sex)

{

// защита от некоректного ввода

var temp = \_sex[sex.ToLower()];

MySex = (temp == 1) ? Sex.masculine : (temp == 0) ? Sex.female : throw new Exception("Пол не определён");

}

public static bool IsMyGender(string sex, bool isStandartView = false)

=> \_sex.ContainsKey(sex.ToLower());

// если что то это посоветовал компилятор

public static bool operator ==(Gender c1, Gender c2) => c1.MySex == c2.MySex;

public static bool operator !=(Gender c1, Gender c2) => c1.MySex != c2.MySex;

public override string ToString() => MySex.ToString();

}

}

ZodiacSign.cs:

using System;

namespace laba\_5

{

public class ZodiacSign

{

public enum ZodiacSigns

{

aries,

taurus,

twins,

cancer,

leo,

virgo,

libra,

scorpio,

sagittarius,

capricorn,

aquarius,

pisces,

}

public ZodiacSigns MyZodiacSign { get; }

public ZodiacSign(in DateTime birthday)

{

var year = birthday.Year;

if ((new DateTime(year, 12, 22) <= birthday) || (birthday <= new DateTime(year, 1, 19)))

{

MyZodiacSign = ZodiacSigns.aquarius;

}

else if (birthday <= new DateTime(year, 2, 18))

{

MyZodiacSign = ZodiacSigns.aquarius;

}

else if (birthday <= new DateTime(year, 3, 21))

{

MyZodiacSign = ZodiacSigns.pisces;

}

else if (birthday <= new DateTime(year, 4, 19))

{

MyZodiacSign = ZodiacSigns.aries;

}

else if (birthday <= new DateTime(year, 5, 20))

{

MyZodiacSign = ZodiacSigns.taurus;

}

else if (birthday <= new DateTime(year, 6, 20))

{

MyZodiacSign = ZodiacSigns.twins;

}

else if (birthday <= new DateTime(year, 7, 22))

{

MyZodiacSign = ZodiacSigns.cancer;

}

else if (birthday <= new DateTime(year, 8, 22))

{

MyZodiacSign = ZodiacSigns.leo;

}

else if (birthday <= new DateTime(year, 9, 22))

{

MyZodiacSign = ZodiacSigns.virgo;

}

else if (birthday <= new DateTime(year, 10, 22))

{

MyZodiacSign = ZodiacSigns.libra;

}

else if (birthday <= new DateTime(year, 11, 21))

{

MyZodiacSign = ZodiacSigns.scorpio;

}

else if (birthday <= new DateTime(year, 12, 21))

{

MyZodiacSign = ZodiacSigns.sagittarius;

}

}

public int IsAPaer(in ZodiacSign zodiacSignPaer)

{

var temp = Math.Abs(MyZodiacSign - zodiacSignPaer.MyZodiacSign);

temp = (temp <= 6) ? temp : 12 - temp;

if (temp == 0 || temp == 6)

{

return 3;

}

else if (temp == 2)

{

return 7;

}

else if (temp == 4)

{

return 11;

}

return 0;

}

public override string ToString() => MyZodiacSign.ToString();

}

}

СompatibilityByName.cs:

using System.Collections.Generic;

namespace laba\_5

{

public class СompatibilityByName

{

// проблема с английскими букавами

private static readonly Dictionary<int, SortedSet<char>> \_map

= new Dictionary<int, SortedSet<char>>

{

{1, new SortedSet<char> {'а', 'и','с', 'ъ', 'a', 'j', 's'} },

{2, new SortedSet<char> {'б', 'й','т', 'ы', 'b', 'k', 't'} },

{3, new SortedSet<char> {'в', 'к','у', 'ь', 'c', 'l', 'u'} },

{4, new SortedSet<char> {'г', 'л','ф', 'э', 'd', 'm', 'v'} },

{5, new SortedSet<char> {'д', 'м','х', 'ю', 'e', 'n', 'w'} },

{6, new SortedSet<char> {'е', 'н','ц', 'я', 'f', 'o', 'x'} },

{7, new SortedSet<char> {'ё', 'о','ч', 'g', 'p', 'y'} },

{8, new SortedSet<char> {'ж', 'п','ш', 'h', 'q', 'z'} },

{9, new SortedSet<char> {'з', 'р','щ', 'i', 'r'} },

};

private static readonly int[,] \_arr =

{

{5,8,3,8,3,9,7,7,9},

{8,3,7,5,4,6,6,8,5},

{3,7,3,5,7,6,2,5,6},

{8,5,5,3,8,5,7,3,8},

{3,4,7,8,9,9,8,3,8},

{9,6,6,5,9,5,6,9,5},

{7,6,2,7,8,6,4,2,9},

{8,8,5,3,3,9,2,6,5},

{9,5,6,8,8,5,9,5,6}

};

public int MyNamber { get; }

public СompatibilityByName(in string name)

{

var temp = 0;

foreach (var c in name)

{

for (var i = 1; i < 10; i++)

{

var tempBool = \_map[i].Contains(char.ToLower(c));

temp += tempBool ? i : 0;

if (tempBool)

{

break;

}

}

}

MyNamber = DigitsSum(DigitsSum(temp));

if (MyNamber < 1 || MyNamber > 9) throw new System.Exception("Не возможно обработать слова - возможно только английские и русские буквы");

}

public int IsAPaer(in СompatibilityByName person) => \_arr[MyNamber - 1, person.MyNamber - 1];

private static int DigitsSum(int number)

{

if (number == 0)

{

return 0;

}

return number % 10 + DigitsSum(number / 10);

}

}

}

User.cs:

using System;

using System.Collections.Generic;

namespace laba\_5

{

public class Name

{

public string first { get; set; }

public string last { get; set; }

}

public class Location

{

public string city { get; set; }

}

public class Login

{

public string username { get; set; }

public string password { get; set; }

}

public class Dob

{

public DateTime date { get; set; }

}

public class Result

{

public string gender { get; set; }

public Name name { get; set; }

public Location location { get; set; }

public Login login { get; set; }

public Dob dob { get; set; }

}

public class Root

{

public List<Result> results { get; set; }

}

}

DataBase.cs:

using System;

using System.Collections.Generic;

using System.Linq;

namespace laba\_5

{

public class DataBase

{

private readonly SortedDictionary<string, Admin> \_admins = new SortedDictionary<string, Admin>

{ ["AlexShev"] = new Admin("AlexShev", "0123") };

private readonly SortedDictionary<string, Client> \_clients = new SortedDictionary<string, Client>();

private readonly SortedDictionary<string, ClientTonw> \_sortedClients = new SortedDictionary<string, ClientTonw>();

private class ClientTonw

{

private readonly List<Client> \_female = new List<Client>();

private readonly List<Client> \_masculine = new List<Client>();

public List<Client> ChoesPartTounBySex(Client client, bool oppositeSex = false)

{

var isMasculine = client.MySex.MySex == Gender.Sex.masculine;

return (oppositeSex) ? (isMasculine) ? \_female : \_masculine : (isMasculine) ? \_masculine : \_female;

}

}

private List<Client> ChoesToun(Client client, bool oppositeSex = false)

{

if (!\_sortedClients.ContainsKey(client.MyCity))

{

\_sortedClients.Add(client.MyCity, new ClientTonw());

}

return \_sortedClients[client.MyCity].ChoesPartTounBySex(client, oppositeSex);

}

public void GenerateCityInDataBase(int size, string city = null)

{

for (var i = 0; i < size; i++)

{

try

{

var generateСlient = new GeneratorUserInfoClient();

AddСlient(new Client(generateСlient.GenerateСlient(city), true));

}

catch (Exception){}

}

}

public void AddAdmin(Admin admin)

{

\_admins.Add(admin.Login, admin);

}

public void AddСlient(Client client)

{

\_clients.Add(client.Login, client);

ChoesToun(client).Add(client);

}

public bool IsFreeLoginClients(string login) => !(\_clients.ContainsKey(login));

public bool IsFreeLoginAdmins(string login) => !(\_admins.ContainsKey(login));

public bool IsMyAdmin(Admin admin) => \_admins.ContainsValue(admin);

public bool IsMyClient(string[] loginPassword) => \_clients.ContainsKey(loginPassword[0])&&\_clients[loginPassword[0]].IsMyPassword(loginPassword[1]);

public Client GetClient(string[] loginPassword)

=> (IsMyClient(loginPassword)) ? \_clients[loginPassword[0]] : throw new Exception("Такого пользователя нет в базе данных");

// бунтовал, что не может привести к этому виду, надеюсь правильно сделал

public IOrderedEnumerable<KeyValuePair<int, List<Client>>> FindPiars(string[] loginPassword, int maxAgeDif = 5)

{

var result = new SortedDictionary<int, List<Client>>();

Client client = GetClient(loginPassword);

foreach (var c in ChoesToun(client, true))

{

var tempScore = client.ScoreIsPaerWithoutSexAndLocalization(c, maxAgeDif);

if (tempScore >= 25)

{

if (!result.ContainsKey(tempScore))

{

result.Add(tempScore, new List<Client>());

}

result[tempScore].Add(c);

}

}

return result.OrderByDescending(kvp => kvp.Key);

}

public void DeleteAdmin(Admin admin)

{

if (IsMyAdmin(admin))

{

\_admins.Remove(admin.Login);

}

else

{

throw new Exception("Такого админа нет или неправильный пароль");

}

}

public void DeleteClientByClient(string[] loginPassword)

{

DeleteClient(GetClient(loginPassword));

}

public void DeleteClientByAdmin(Admin admin, string loginClient)

{

if (IsMyAdmin(admin))

{

DeleteClient(\_clients[loginClient]);

}

else

{

throw new Exception("Вы не админ");

}

}

private void DeleteClient(Client client)

{

\_clients.Remove(client.Login);

ChoesToun(client).Remove(client);

}

public void DeleteAllClient(Admin admin)

{

if (IsMyAdmin(admin))

{

\_clients.Clear();

\_sortedClients.Clear();

}

else

{

throw new Exception("Вы не админ");

}

}

}

}

ConsolIneractor.cs:

using System;

using System.Linq;

namespace laba\_5

{

// не вижу смыса делать его прям суперским, всё равно сносить

// нужен для демнстрации логики

public class ConsolIneractor

{

public static void Run()

{

var dataBase = new DataBase();

// я считаю, никто не должен знать, что данные вымешленные, поэтому захардкодил

// генерит 4 минуты

//dataBase.GenerateCityInDataBase(500, "Таганрог");

//dataBase.GenerateCityInDataBase(20, "Ростов-на-Дону");

var condition = true;

while (condition)

{

Console.Clear();

Menu();

switch (ReadComend())

{

case ('1'): RunForClient(dataBase); break;

case ('2'): RunForClient(dataBase, true); break;

case ('3'): RunForAdmin(dataBase); break;

case ('0'): Console.WriteLine("До свидания"); condition = false; break;

default: Console.WriteLine("Такой команды нет"); break;

}

}

}

private static void Menu()

{

Console.WriteLine("1: Войти");

Console.WriteLine("2: Зарегестрироваться");

Console.WriteLine("3: Войти как администратор");

Console.WriteLine("0: Завершение работы");

}

private static char ReadComend()

{

Console.Write(">>");

var comand = Console.ReadKey().KeyChar;

Console.Clear();

return comand;

}

private static void Сontinue()

{

Console.WriteLine("\nДля продолжения нажмите любую клавишу");

Console.ReadKey();

Console.Clear();

}

private static void RunForClient(DataBase dataBase, bool isNewClient = false)

{

Console.WriteLine("Добро пожаловать\n");

var temp = (isNewClient) ? AddClient(dataBase) : Identification();

if (isNewClient || dataBase.IsMyClient(temp))

{

Console.Clear();

if (isNewClient)

{

if (!Confirmation())

{

return;

}

}

var condition1 = true;

while (condition1)

{

try

{

Console.Clear();

MenuForClient();

switch (ReadComend())

{

case ('1'): FindPaer(dataBase, temp); break;

case ('2'): Console.WriteLine("Спасибо, нам это очень важно\n"); Сontinue(); break;

case ('3'): DeleteClient(dataBase, temp); condition1 = false; break;

case ('0'): Console.WriteLine("Вы вышли из своего аккаунта\n"); condition1 = false; break;

default: Console.WriteLine("Такой команды нет\n"); break;

}

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

}

}

else if (dataBase.IsFreeLoginClients(temp[0]))

{

Console.WriteLine("Такого пользователя пока нет, зарегестрируйтесь\n");

Сontinue();

}

else

{

Console.WriteLine("Вы ошиблись при вводе пароля\n");

Сontinue();

}

}

private static string[] Identification() => new string[2] { ReadLogin(), ReadPassword() };

private static string ReadLogin()

{

while (true)

{

Console.WriteLine("Введите логин:");

var login = Console.ReadLine();

Console.WriteLine();

if (StandartView.IsLogin(login))

{

return login;

}

else

{

Console.WriteLine("Не возможный формат логина\n");

}

}

}

private static string ReadPassword()

{

while (true)

{

Console.WriteLine("Введите пароль:");

var password = Console.ReadLine();

Console.WriteLine();

if (password.Length > 0 && !password.Contains(' '))

{

return password;

}

else

{

Console.WriteLine("Не возможный формат пароля\n");

}

}

}

private static bool Confirmation()

{

Console.Write("\nЕсли хотите продолжить нажмите Enter, иначе другую клавишу... ");

return Console.ReadKey().Key == ConsoleKey.Enter;

}

private static void MenuForClient()

{

Console.WriteLine("1: Подобрать пару");

Console.WriteLine("2: Поощрить нашу работу");

Console.WriteLine("3: Удалить свой аккаунт");

Console.WriteLine("0: Выйти");

}

private static void FindPaer(DataBase dataBase, string[] person)

{

var counter = 0;

Console.Write("Введите максимальную разницу в возрасте: ");

var maxDiferentAge = int.Parse(Console.ReadLine());

var temp = dataBase.FindPiars(person, maxDiferentAge);

if (temp.Any())

{

Console.WriteLine("\nЧем ближе число к 100, тем больше Вам подходит человек\n");

var flag = false;

foreach (var c in temp)

{

if (flag) break;

Console.WriteLine($"\n===================================================={c.Key}====================================================\n");

foreach (var c1 in c.Value)

{

if (counter % 5 == 0 && counter != 0)

{

if (!Confirmation())

{

flag = true;

break;

}

else

{

Console.WriteLine();

}

}

Console.Write($"\n№{++counter} ");

PrintClient(c1);

}

}

}

else

{

Console.WriteLine("Мы не смогли подобрать Вам пару, подождите немного, надеемся, что скоро мы сможем это сделать");

}

Console.WriteLine("\n====================================================КОНЕЦ СПИСКА====================================================\n");

Сontinue();

}

private static void PrintClient(Client client)

{

Console.Write($"ФИО {client.MyFulName} родил{((client.MySex.MySex == Gender.Sex.masculine) ? "ся" : "ась" )} " + $"{client.MyBirthday.ToShortDateString()} возраст {client.MyAge} контакты: логин {client.Login} номер телефона {client.MyPhoneNumber} \n\n");

}

private static void DeleteClient(DataBase dataBase, string[] loginPassword)

{

Console.WriteLine("Для подтверждения нажмите 1");

if (Console.ReadKey().Key == ConsoleKey.D1)

{

dataBase.DeleteClientByClient(loginPassword);

Console.WriteLine("\nСпасибо за использование нашей программы, надеемся, что мы Вам помогли подобрать пару\n");

}

else

{

Console.WriteLine("\nОтмена команды\n");

}

}

private static string[] AddClient(DataBase dataBase)

{

Console.WriteLine("Введите необходимые данные");

var temp = ReadClientInformation(dataBase);

dataBase.AddСlient(new Client(temp, true));

return temp[0..2];

}

private static string[] ReadClientInformation(DataBase dataBase)

{

var result = new string[7];

var temp = ReadLogAndPas(dataBase);

result[0] = temp[0]; result[1] = temp[1];

Console.WriteLine("\nВведите ФИО:");

result[2] = ReadFullName();

while (true)

{

Console.WriteLine("\nПол:");

result[3] = StandartView.ConverteToStandartWord(Console.ReadLine());

if (Gender.IsMyGender(result[3], true))

{

break;

}

else

{

Console.WriteLine("\nВы вели что-то не так, попробуйте ещё раз");

}

}

while (true)

{

try

{

Console.WriteLine("\nДата рождения:");

result[4] = Console.ReadLine();

StandartView.ConverteStringToDate(result[4]);

break;

}

catch (Exception)

{

Console.WriteLine("\nВы вели что-то не так, попробуйте ещё раз");

}

}

while (true)

{

try

{

Console.WriteLine("\nГород проживания:");

result[5] = SetString(Console.ReadLine());

break;

}

catch (Exception)

{

Console.WriteLine("\nВы вели что-то не так, попробуйте ещё раз");

}

}

while (true)

{

Console.WriteLine("\nНомер телефона:");

result[6] = Console.ReadLine();

if (StandartView.IsPhoneNumber(result[6]))

{

break;

}

else

{

Console.WriteLine("\nВы вели что-то не так, попробуйте ещё раз");

}

}

Console.WriteLine();

return result;

}

private static string[] ReadLogAndPas(DataBase dataBase)

{

var temp = new string[2];

string control;

while (true)

{

temp[0] = ReadLogin();

if (!StandartView.IsLogin(temp[0]))

{

Console.WriteLine("Извините, это не может быть логином");

}

else if (dataBase.IsFreeLoginAdmins(temp[0]))

{

break;

}

else

{

Console.WriteLine("Извините, пользоавтель с таким логином уже есть, придумайте новый");

}

}

while (true)

{

temp[1] = ReadPassword();

Console.WriteLine("Введите пароль повторно:");

control = Console.ReadLine();

if (temp[1] == control)

{

break;

}

else

{

Console.WriteLine("Вы ошиблись при вводе, они должны быть одинаковы");

}

}

return temp;

}

private static string SetString(string str)

{

try

{

var res = StandartView.ConverteToStandartString(str);

return res;

}

catch (Exception ex)

{

Console.WriteLine("\nОшибка при вводе " + ex.Message);

throw;

}

}

private static string ReadFullName()

{

var res = string.Empty;

while (true)

{

try

{

string[] strArrey;

while (true)

{

strArrey = StandartView.ToStringArray(Console.ReadLine());

if (strArrey.Length == 2)

{

break;

}

else

{

Console.WriteLine("\nВы что-то так ввели, должно быть три параметра - фамилия, имя\n");

Console.WriteLine("Если у Вас составное имя/фамилия - пишите через тире\n");

Console.WriteLine("Введите данные повторно\n");

}

}

res += SetString(strArrey[0]) + ' ' + SetString(strArrey[1]);

break;

}

catch (Exception)

{

Console.WriteLine("\nВведите данные повторно\n");

res = null;

}

}

return res;

}

private static void MenuForAdmin()

{

Console.WriteLine("1: Добавить клиента в базу");

Console.WriteLine("2: Удалить клиента из базу");

Console.WriteLine("3: Добавить админа");

Console.WriteLine("4: Удалить свой аккаунт");

Console.WriteLine("5: Удалить всю базу данных");

Console.WriteLine("0: Выйти");

}

private static void RunForAdmin(DataBase dataBase)

{

var temp = Identification();

var admin = new Admin(temp[0], temp[1]);

if (dataBase.IsMyAdmin(admin))

{

var condition = true;

while (condition)

{

try

{

Console.Clear();

MenuForAdmin();

switch (ReadComend())

{

case ('1'): AddHumanByAdmin(dataBase); break;

case ('2'): DeleteHumanByAdmin(dataBase, admin); break;

case ('3'): AddHumanByAdmin(dataBase, true); break;

case ('4'): DeleteHumanByAdmin(dataBase, admin, true); condition = false; break;

case ('5'): DeleteDataBase(ref dataBase, admin); break;

case ('0'): Console.WriteLine("Вы вышли из аккаунта администратора\n"); condition = false; break;

default: Console.WriteLine("Такой команды нет\n"); break;

}

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

}

}

else if (dataBase.IsFreeLoginAdmins(temp[0]))

{

Console.WriteLine("Ну поиграли и хватит, Вы не администратор\n");

}

else

{

Console.WriteLine("Вы ошиблись при вводе пароля\n");

}

Сontinue();

}

private static void AddHumanByAdmin(DataBase dataBase, bool addAdmin = false)

{

if (addAdmin)

{

var temp = ReadLogAndPas(dataBase);

dataBase.AddAdmin(new Admin(temp[0], temp[1]));

}

else

{

AddClient(dataBase);

}

}

private static void DeleteHumanByAdmin(DataBase dataBase, Admin admin, bool deliteaAdmin = false)

{

Console.WriteLine("Подтвердите решения введя пароль");

if (admin.IsMyPassword(ReadPassword()))

{

if (deliteaAdmin)

{

dataBase.DeleteAdmin(admin);

}

else

{

var loginClient = ReadLogin();

dataBase.DeleteClientByAdmin(admin, loginClient);

}

}

else

{

Console.WriteLine("что-то пошло не так");

}

}

private static void DeleteDataBase(ref DataBase dataBase, Admin admin)

{

Console.WriteLine("Подтвердите решения введя пароль");

if (admin.IsMyPassword(ReadPassword()))

{

dataBase.DeleteAllClient(admin);

}

else

{

Console.WriteLine("Отмена команды\n");

}

}

}

}

Program.cs

using System;

namespace laba\_5

{

public class Program

{

public static void Main()

{

try

{

ConsolIneractor.Run();

}

catch (Exception e)

{

Console.WriteLine($"Не впойманая ошибка\n{e.Message}");

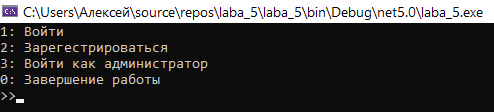
}

}

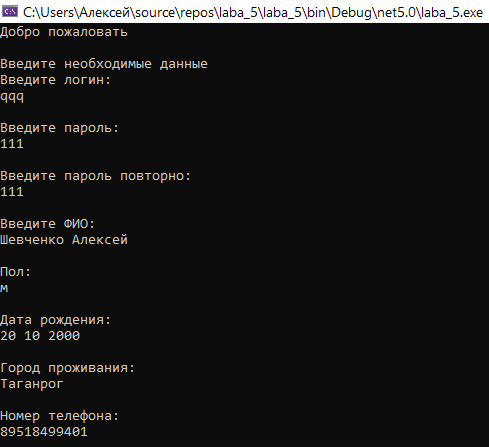
}

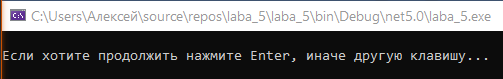
}

1. Результаты программы:

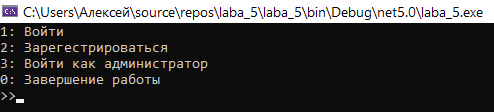


2

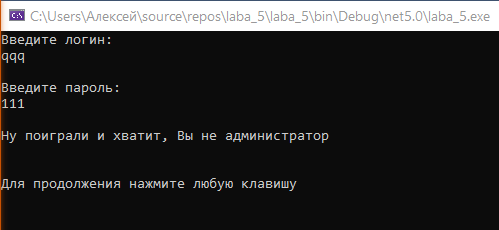


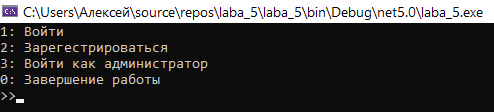


пробел

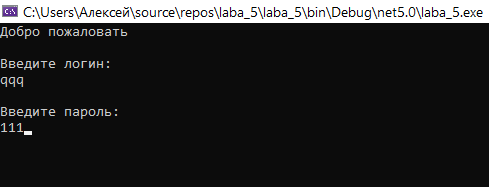


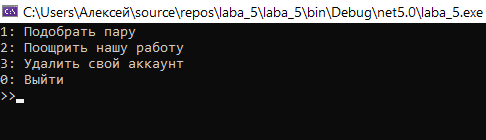
3



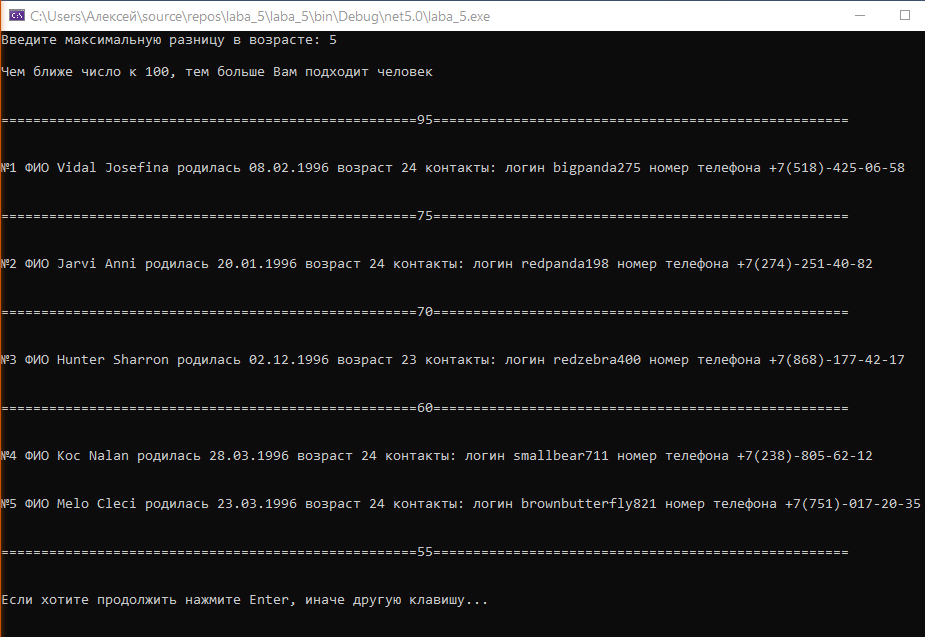


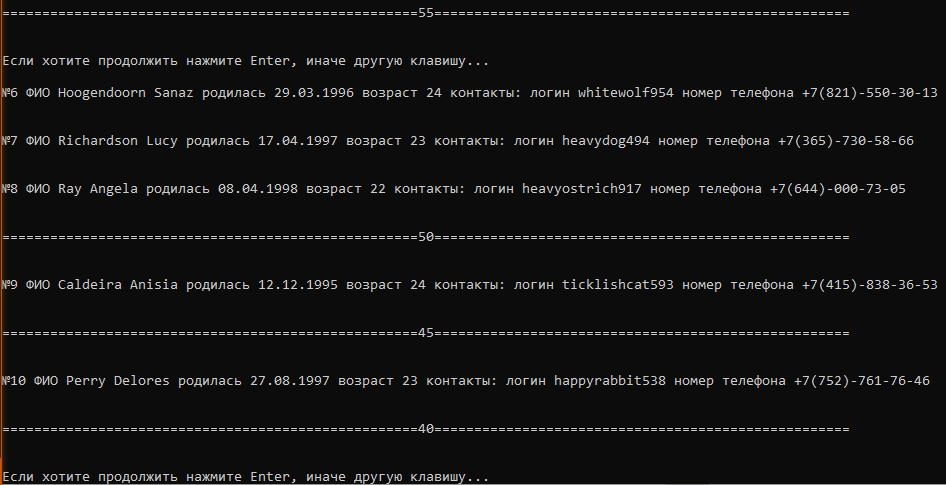
1

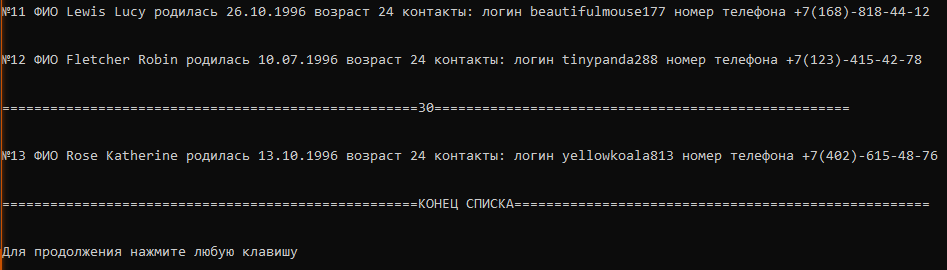


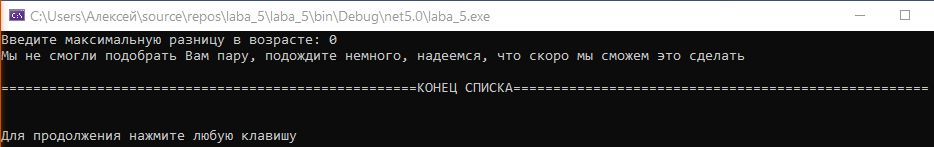


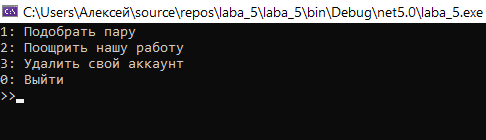
1



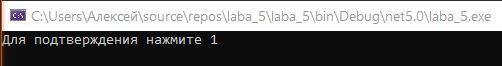




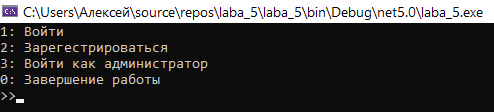




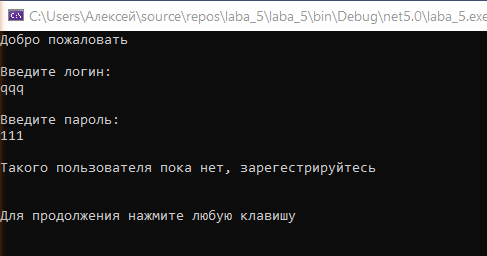
3

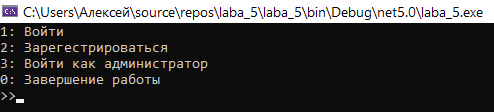


1

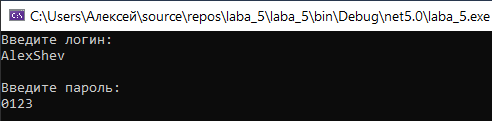


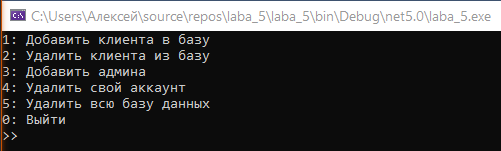
1



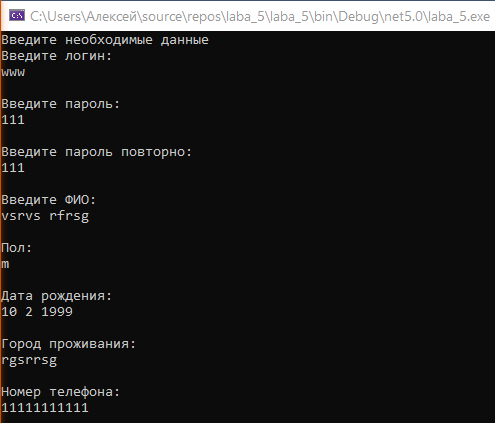


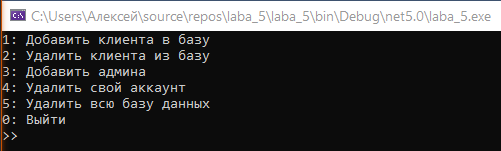
3



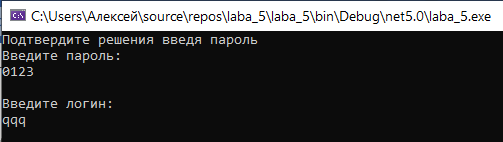


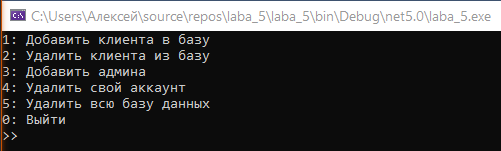
1



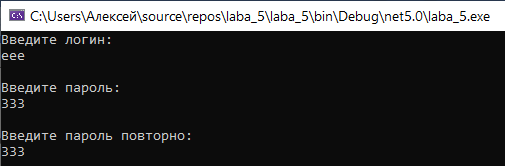


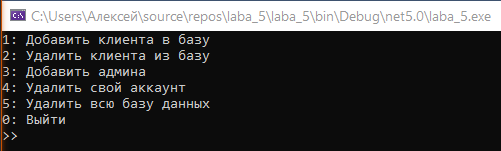
2



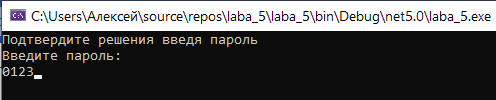


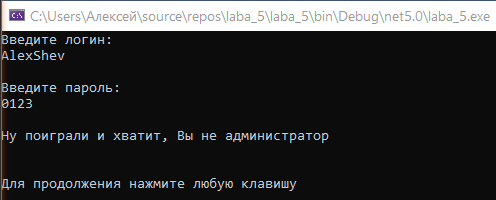
3

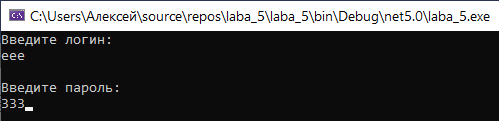


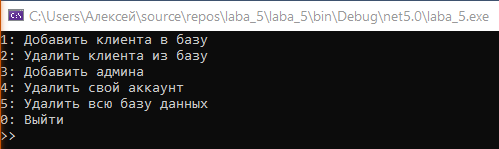


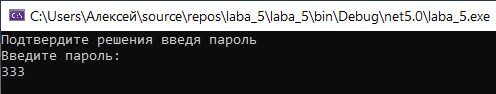
4

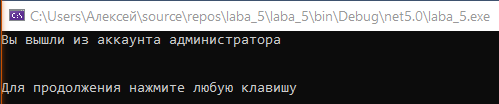


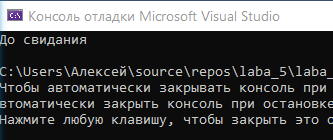












**Вывод:** был ознакомлен с основами объектно-ориентированного программирования на С#.