**ЛАБОРАТОРНОЯ РАБОТА 1-4**

**«Консольные приложения. Работа с классами»**

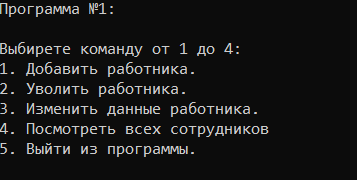
**Цель:** Разработать консольные приложения с использованием классов.

**Задание:**

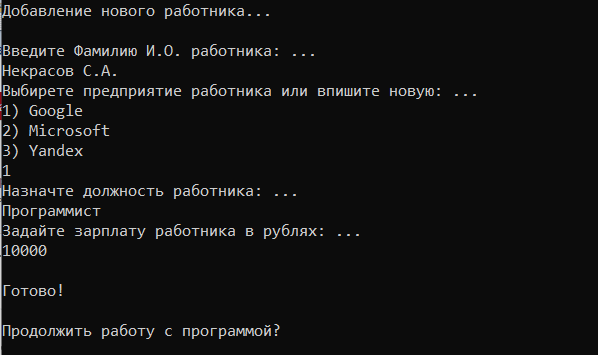
1. Описать классы "Работник" и "Предприятие".
2. Описать классы "Автомобиль" и "Стоянка".
3. Доработать программы, написать класс для сохранения состояния программы в текстовом файле.
4. Написать класс для логирования событий в текстовый файл.

**Класс «Работники»**

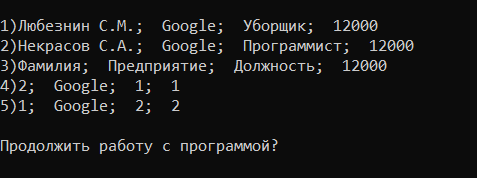
Выполнение задания:



Добавление работника:



Просмотр работников:



Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Text.RegularExpressions;

namespace Lab\_Class

{

class Program

{

static void Main(string[] args)

{

string path = AppContext.BaseDirectory + @"\Base.csv";

int command = 0;

Console.WriteLine("Программа №1:");

Console.WriteLine();

Console.WriteLine("Выбирете команду от 1 до 4: ");

Console.WriteLine("1. Добавить работника.");

Console.WriteLine("2. Уволить работника.");

Console.WriteLine("3. Изменить данные работника.");

Console.WriteLine("4. Посмотреть всех сотрудников");

Console.WriteLine("5. Выйти из программы.");

try

{

command = Convert.ToInt32(Console.ReadLine());

}

catch

{

Main(args);

Console.WriteLine("Неккоректные данные!");

}

Console.Clear();

if (command == 1)

{

//

Console.WriteLine("Добавление нового работника...");

Console.WriteLine();

Console.WriteLine("Введите Фамилию И.О. работника: ...");

string fio = Console.ReadLine();

Console.WriteLine("Выбирете предприятие работника или впишите новую: ...");

string[] factoryChose = new string[] { "Google", "Microsoft", "Yandex" };

for (int i = 0; i < factoryChose.Length; i++)

{

Console.WriteLine((i + 1).ToString() + ") " + factoryChose[i]);

}

string facrory = Console.ReadLine();

if (facrory == "1")

{

facrory = "Google";

}

else if (facrory == "2")

{

facrory = "Microsoft";

}

else if (facrory == "3")

{

facrory = "Yandex";

}

Console.WriteLine("Назначте должность работника: ...");

string post = Console.ReadLine();

Console.WriteLine("Задайте зарплату работника в рублях: ...");

float pay = 0;

try

{

pay = Convert.ToInt32(Console.ReadLine());

}

catch

{

Console.WriteLine("Неккоректные данные!");

Console.WriteLine("Задайте зарплату работника в рублях: ...");

}

Worker worker = new Worker { FIO = fio, facrory = facrory, post = post, pay = pay };

string text = worker.FIO + "; " + worker.facrory + "; " + worker.post + "; " + worker.pay;

if (File.Exists(path) == false)

{

FileStream file1 = new FileStream(path, FileMode.Create);

StreamWriter writer = new StreamWriter(file1);

writer.WriteLine(text);

writer.Close();

Console.WriteLine("");

Console.WriteLine("Готово!");

}

else

{

File.AppendAllText(path, Environment.NewLine + text);

Console.WriteLine("");

Console.WriteLine("Готово!");

}

}

else if (command == 2)

{

Console.WriteLine("Увольнение работника...");

Console.WriteLine();

Console.WriteLine("Введите Фамилию И.О. работника: ...");

string fio = Console.ReadLine();

FileStream file1 = new FileStream(path, FileMode.Open);

StreamReader reader = new StreamReader(file1);

string[] txtB = new string[99];

for (int i = 0; !reader.EndOfStream; i++)

{

txtB[i] = reader.ReadLine();

}

string final = "";

reader.Close();

for (int i = 0; i < txtB.Length; i++)

{

string[] line = txtB[i].Split(';');

if (line[0] == fio)

{

txtB[i] = null;

break;

}

}

int r = 0;

for (int i = 0; i < txtB.Length; i++)

{

if (i != txtB.Length - 1)

final += txtB[i] + Environment.NewLine;

else

final += txtB[i];

}

File.WriteAllText(path, final);

Console.WriteLine("");

Console.WriteLine("Готово!");

}

else if (command == 3)

{

Console.WriteLine("Изменение данных работника...");

Console.WriteLine();

Console.WriteLine("Введите Фамилию И.О. работника: ...");

string fio = Console.ReadLine();

FileStream file1 = new FileStream(path, FileMode.Open);

StreamReader reader = new StreamReader(file1);

string[] txtB = new string[99];

for (int i = 0; !reader.EndOfStream; i++)

{

txtB[i] = reader.ReadLine();

}

for (int i = 0; i < txtB.Length; i++)

{

string[] line = txtB[i].Split(';');

if (line[0] == fio)

{

break;

}

if(line[0] != fio)

{

Console.Clear();

Console.WriteLine("Такого работника нет!");

Console.WriteLine();

reader.Close();

Main(args);

}

}

string final = "";

reader.Close();

Console.WriteLine("Что вы хотели бы изменить?");

Console.WriteLine("1. ФИО");

Console.WriteLine("2. Предприятие");

Console.WriteLine("3. Должность");

Console.WriteLine("4. Зарплату");

try

{

command = Convert.ToInt32(Console.ReadLine());

}

catch

{

Console.WriteLine("Неккоректные данные!");

Console.WriteLine("Что вы хотели бы изменить?");

Console.WriteLine("1. ФИО");

Console.WriteLine("2. Предприятие");

Console.WriteLine("3. Должность");

Console.WriteLine("4. Зарплату");

command = Convert.ToInt32(Console.ReadLine());

}

if (command == 1)

{

Console.WriteLine("Задайте новую Фамилию И.О.");

string newFIO = Console.ReadLine();

for (int i = 0; i < txtB.Length; i++)

{

string[] line = txtB[i].Split(';');

if (line[0] == fio)

{

txtB[i] = newFIO + "; " + line[1] + "; " + line[2] + "; " + line[3];

break;

}

}

}

else if (command == 2)

{

Console.WriteLine("Задайте новое предприятие работника");

string newFactory = Console.ReadLine();

Console.WriteLine("Задайте его новую должность");

string newPost = Console.ReadLine();

Console.WriteLine("Задайте его заработную плату");

string newPay = Console.ReadLine();

for (int i = 0; i < txtB.Length; i++)

{

string[] line = txtB[i].Split(';');

if (line[0] == fio)

{

txtB[i] = line[0] + "; " + newFactory + "; " + newPost + "; " + newPay;

break;

}

}

}

else if (command == 3)

{

Console.WriteLine("Задайте новую должность");

string newPost = Console.ReadLine();

Console.WriteLine("Задайте его заработную плату");

string newPay = Console.ReadLine();

for (int i = 0; i < txtB.Length; i++)

{

string[] line = txtB[i].Split(';');

if (line[0] == fio)

{

txtB[i] = line[0] + "; " + line[1] + "; " + newPost + "; " + newPay;

break;

}

}

}

else if (command == 4)

{

Console.WriteLine("Задайте новую заработную плату");

string newPay = Console.ReadLine();

for (int i = 0; i < txtB.Length; i++)

{

string[] line = txtB[i].Split(';');

if (line[0] == fio)

{

txtB[i] = line[0] + "; " + line[1] + "; " + line[2] + "; " + newPay;

break;

}

}

}

int r = 0;

for (int i = 0; i < txtB.Length; i++)

{

if (i != txtB.Length - 1)

final += txtB[i] + Environment.NewLine;

else

final += txtB[i];

}

File.WriteAllText(path, final);

Console.WriteLine("");

Console.WriteLine("Готово!");

}

else if (command == 4)

{

string[] text = File.ReadAllLines(path);

int num = 1;

for (int i = 0; i < text.Length; i++)

{

text[i] = num + ")" + text[i];

num++;

}

Console.WriteLine();

for (int i = 0; i < text.Length; i++)

{

if (((i + 1).ToString() + ")") != text[i])

Console.WriteLine(text[i]);

}

}

else if (command == 5)

{

Environment.Exit(0);

}

else if (command >= 6 || command <= 0)

{

Console.Clear();

Console.WriteLine("Ошибка, попробуйте снова...");

Console.WriteLine();

Console.Clear();

Main(args);

}

{

string[] text = File.ReadAllLines(path);

string textF = "";

for (int i = 0; i < text.Length; i++)

{

if (text[i] != string.Empty)

{

if (i == 0)

{

textF += text[i];

}

else

{

textF += Environment.NewLine + text[i];

}

}

}

File.WriteAllText(path, String.Empty);

File.Create(path).Close();

File.AppendAllText(path, textF);

textF = "";

}

Console.WriteLine();

Console.WriteLine("Продолжить работу с программой?");

string commandEnd = Console.ReadLine();

if (commandEnd.ToString() != "")

{

Console.Clear();

Main(args);

}

}

}

class Worker

{

// ФИО

public string FIO;

// Предприятие

public string facrory;

// Должность

public string post;

// Зарплата

public float pay;

public String Info(string workerInfo)

{

workerInfo = "ФИО: " + FIO + Environment.NewLine + "Предприятие: " + facrory + Environment.NewLine + "Должность: " + post + Environment.NewLine + "Зарплата: " + pay.ToString() + " руб.";

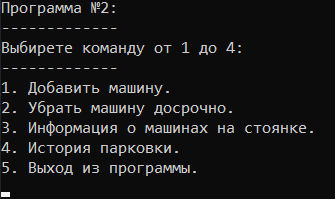
return workerInfo;

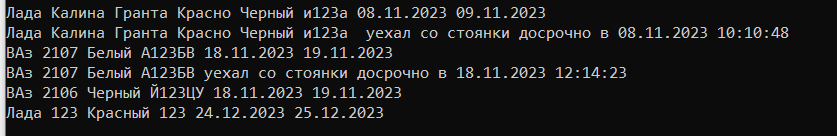
}

}

}

**Класс «Стоянка»**





Код:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

namespace Lab\_Class\_2

{

class Program

{

static void Main(string[] args)

{

string path = AppContext.BaseDirectory + @"\Base.txt";

string pathHistory = AppContext.BaseDirectory + @"\History.txt";

int command = 0;

var parking = new CarPark

{

ParkedCars = new List<Car>

{

}

};

if (File.Exists(path) == false)

{

File.Create(AppContext.BaseDirectory + @"\Base.txt");

}

else

{

parking.LoadList();

}

Console.WriteLine("Программа №2:");

Console.WriteLine("-------------");

Console.WriteLine("Выбирете команду от 1 до 4: ");

Console.WriteLine("-------------");

Console.WriteLine("1. Добавить машину.");

Console.WriteLine("2. Убрать машину досрочно.");

Console.WriteLine("3. Информация о машинах на стоянке.");

Console.WriteLine("4. История парковки.");

Console.WriteLine("5. Выход из программы.");

try

{

command = Convert.ToInt32(Console.ReadLine());

}

catch

{

Main(args);

Console.WriteLine("Неккоректные данные!");

}

if (command == 1)

{

Console.Clear();

Console.WriteLine("Добавление машины на парковку:");

Console.WriteLine("Производитель:");

var producer = Console.ReadLine();

Console.Clear();

Console.WriteLine("Модель машины:");

var model = Console.ReadLine();

Console.Clear();

Console.WriteLine("Цвет машины:");

var color = Console.ReadLine();

Console.Clear();

Console.WriteLine("Номерной знак:");

var stateNumber = Console.ReadLine();

Console.Clear();

bool once = false;

parking.ParkedCars.ForEach(p =>

{

if (stateNumber == p.PlateLicense)

{

once = true;

}

});

if(once == true)

{

Console.WriteLine("Машина уже на парковке!");

once = false;

}

else

{

parking.ParkCar(new Car

{

ArrivingTime = DateTime.Now,

DepartureTime = new DateTime(DateTime.Now.Year, DateTime.Now.Month, DateTime.Now.Day + 1, DateTime.Now.Hour, DateTime.Now.Minute, DateTime.Now.Second),

Color = color,

Model = model,

Producer = producer,

PlateLicense = stateNumber

});

Console.WriteLine("Машина находится на парковке.");

}

}

else if (command == 2)

{

Console.Clear();

Console.WriteLine("Введите номер машины, которую нужно убрать:");

var stateNumber = Console.ReadLine();

Car car = null;

parking.ParkedCars.ForEach(p =>

{

if(stateNumber == p.PlateLicense)

{

car = p;

}

});

if (car != null)

{

Console.Clear();

Console.WriteLine("Машина была убрана со стоянки.");

parking.CarRemove(car);

}

else

{

Console.Clear();

Console.WriteLine("На стоянке нет такой машины!");

}

}

else if (command == 3)

{

Console.Clear();

Console.WriteLine("Машины на парковке:");

Console.WriteLine();

parking.PrintParkedCars();

}

else if (command == 4)

{

Console.Clear();

Console.WriteLine("История парковки:");

Console.WriteLine();

parking.PrintHistory();

}

else if (command == 5)

{

Environment.Exit(0);

}

else if (command >= 6)

{

Console.Clear();

Console.WriteLine("Ошибка, попробуйте снова...");

Console.WriteLine();

Main(args);

}

parking.SaveList();

Console.WriteLine();

Console.WriteLine("Продолжить работу с программой?");

string commandEnd = Console.ReadLine();

if (commandEnd.ToString() != "exit")

{

Console.Clear();

Main(args);

}

}

}

class CarPark

{

public List<Car> ParkedCars { get; set; }

public List<string> FileList { get; set; }

string path = AppContext.BaseDirectory + @"\Base.txt";

string pathHistory = AppContext.BaseDirectory + @"\History.txt";

public void PrintParkedCars()

{

ParkedCars.ForEach(p =>

{

Console.WriteLine($"{p.Producer}-{p.Model}-{p.Color}-{p.PlateLicense} прибыла на стоянку в: {p.ArrivingTime.ToShortDateString()}");

Console.WriteLine($"Уедет в: {p.DepartureTime.ToShortDateString()}");

Console.WriteLine();

});

}

public void LoadList()

{

FileList = File.ReadAllLines(path).ToList();

FileList.ForEach(p =>

{

string[] s = p.Split(';');

DateTime parseMyFormat(string stringToParse)

{

return DateTime.ParseExact(stringToParse, "dd.MM.yyyy", System.Globalization.CultureInfo.InvariantCulture);

}

if(DateTime.Now < parseMyFormat(s[5]))

{

ParkCarLoad(new Car

{

Producer = s[0],

Model = s[1],

Color = s[2],

PlateLicense = s[3],

ArrivingTime = parseMyFormat(s[4]),

DepartureTime = parseMyFormat(s[5]),

});

}

else

{

string str = File.ReadAllText(pathHistory);

str += $"{s[0]} {s[1]} {s[2]} {s[3]} уехал со стоянки в {s[5]}" + Environment.NewLine;

File.WriteAllText(pathHistory, str);

str = null;

}

});

}

public void SaveList()

{

StringBuilder sb = new StringBuilder();

ParkedCars.ForEach(parkedCar =>

{

sb.Append(parkedCar.ToString());

sb.Append(Environment.NewLine);

});

File.WriteAllText(path, sb.ToString());

}

public void PrintHistory()

{

FileList = File.ReadAllLines(pathHistory).ToList();

FileList.ForEach(Console.WriteLine);

Console.WriteLine();

}

public void SaveHistory(Car car)

{

string str = File.ReadAllText(pathHistory);

str += car.ToString() + Environment.NewLine;

File.WriteAllText(pathHistory, str);

str = null;

}

public void SaveHistoryCarRemove(Car car)

{

string str = File.ReadAllText(pathHistory);

str += car.ToStringLeaveParking();

str += Environment.NewLine;

File.WriteAllText(pathHistory, str);

str = null;

}

public void ParkCarLoad(Car car)

{

ParkedCars.Add(car);

}

public void ParkCar(Car car)

{

ParkedCars.Add(car);

SaveHistory(car);

}

public void CarRemove(Car car)

{

ParkedCars.Remove(car);

SaveHistoryCarRemove(car);

}

}

class Car

{

public string Model { get; set; }

public string Producer { get; set; }

public string Color { get; set; }

public string PlateLicense { get; set; }

public DateTime ArrivingTime { get; set; }

public DateTime DepartureTime { get; set; }

public override string ToString()

{

return $"{Producer} {Model} {Color} {PlateLicense} {ArrivingTime.ToShortDateString()} {DepartureTime.ToShortDateString()}";

}

public string ToStringLeaveParking()

{

return $"{Producer} {Model} {Color} {PlateLicense} уехал со стоянки досрочно в {DateTime.Now}";

}

}

}