МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет имени М. Т. Калашникова»

Институт «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_»

Дата \_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: «Система учета рабочего времени»

Выполнил

студент гр. Б04-191-3 Я. П. Семенов

Руководитель

к.т.н., доцент Р. О. Султанов

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc43938522)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc43938523)

[2. ИЕРАРХИЯ КЛАССОВ 5](#_Toc43938524)

[3. КОНТРОЛЬНЫЙ ПРИМЕР 6](#_Toc43938525)

[ЗАКЛЮЧЕНИЕ 20](#_Toc43938526)

[ПРИЛОЖЕНИЕ 21](#_Toc43938527)

# ВВЕДЕНИЕ

Согласно исследованиям достаточно много работников занимаются посторонними делами в рабочее время. Этот фактор и частые опоздания работников являются причиной потери организации от неэффективной работы персонала. Добиться повышения дисциплины в организации, и свести к минимуму влияние человеческого фактора помогает автоматизированная система учета рабочего времени сотрудников.

В данной курсовой работе разработана система учета рабочего времени. Разработанная система должна обеспечить ведение справочников, учет опозданий, отгулов, больничных и отпусков сотрудников.

# 1. ПОСТАНОВКА ЗАДАЧИ

В курсовой работе должны выполняться такие функции, как ведение справочников, учет опозданий, отгулов, больничных и отпусков сотрудников, а также составление графика работы. Должны создаваться выходные документы: список работников, составление графика работы, отчеты.

В программе реализуются следующие запросы:

1. Список всех работников

2. Список работников по должностям

3. Список работников с графиком работы

4. Список работников с заработными платами

5. Добавить болезнь в историю болезни работника

6. Посмотреть историю болезни

7. Изменить график работы заданного работника

8. Список опозданий, отгулов, больничных, отпусков

9. Изменить количество опозданий

10. Изменить количество отгулов

11. Изменить количество больничных

12. Изменить количество отпусков

13. Добавить работника

Программа выполнена на языке C#. Выходные документы сохраняются в файлах форматов xml и json.

# 2. ИЕРАРХИЯ КЛАССОВ

На рис. 1 представлена иерархия классов

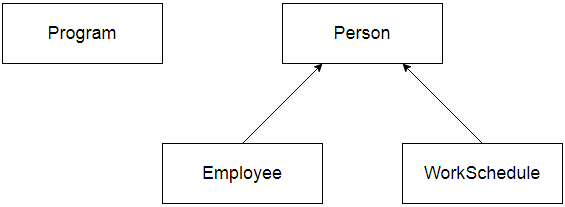


Рис. 1. Иерархия классов

# 3. КОНТРОЛЬНЫЙ ПРИМЕР

На рис. 2 представлено главное меню программы:

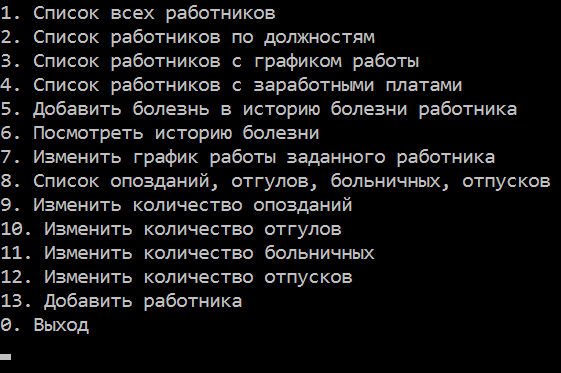


Рис. 2. Главное меню

На рис. 3 представлено выполнение первого запроса:

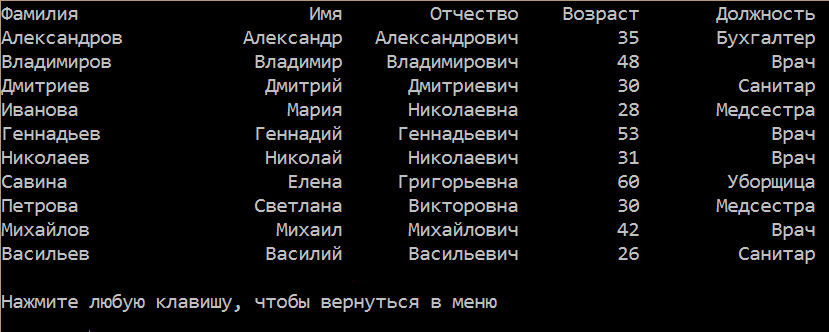


Рис. 3. Первый запрос

Выполнение второго запроса представлено на рис. 4-5:

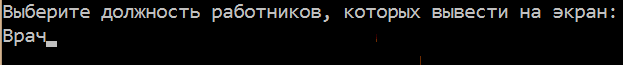


Рис. 4

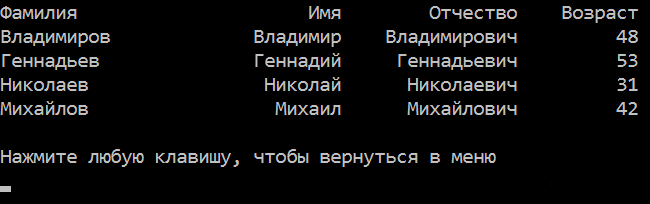


Рис. 5

На рис. 6 представлено выполнение третьего запроса:

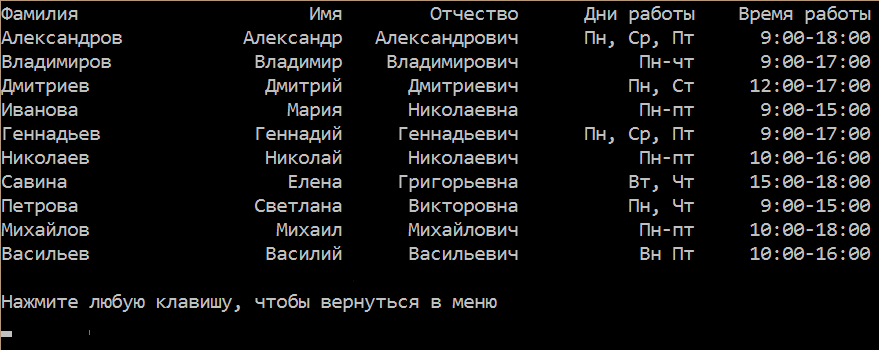


Рис. 6. Третий запрос

На рис. 7 представлено выполнение четвертого запроса:

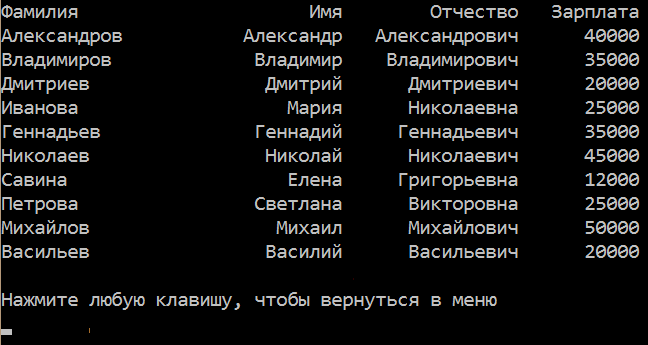


Рис. 7. Четвертый запрос

Выполнение пятого запроса представлено на рис. 8-12:



Рис. 8



Рис. 9

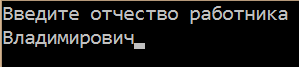


Рис. 10

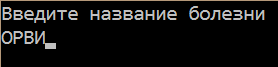


Рис. 11

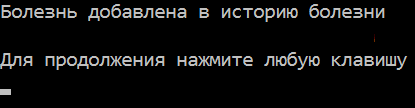


Рис. 12

Программа будет запрашивать ввести название болезни, до тех пор, пока не будет введена пустая строка. Если введена пустая строка, программа возвращается в главное меню.

Выполнение шестого запроса представлено на рис. 13-16:



Рис. 13

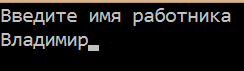


Рис. 14

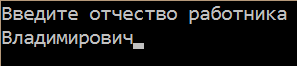


Рис. 15

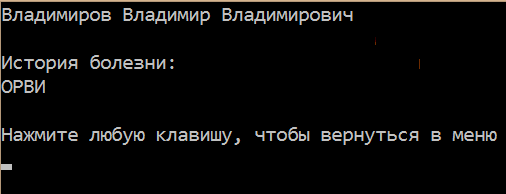


Рис. 16

Выполнение седьмого запроса представлено на рис. 17-22:



Рис. 17

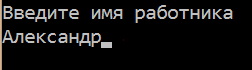


Рис. 18



Рис. 19

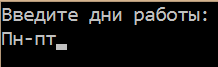


Рис. 20



Рис. 21

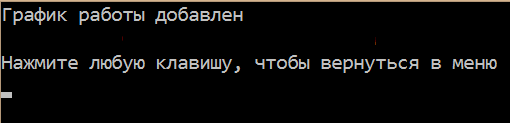


Рис. 22

Теперь график работы Александрова Александра Александровича изменился на заданный:

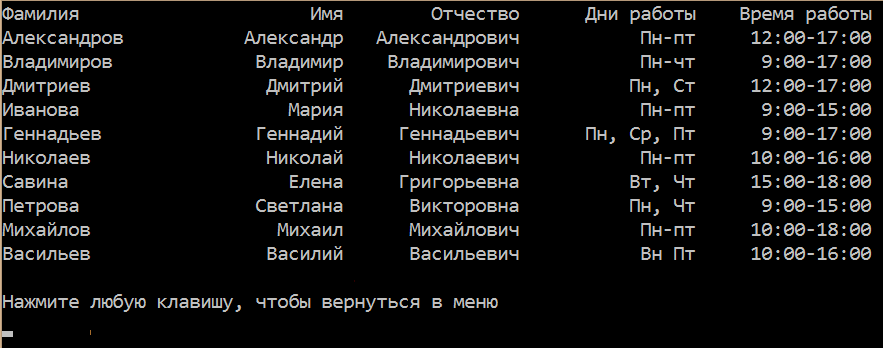


Рис. 23. Новый график работы

На рис. 24 представлено выполнение восьмого запроса:

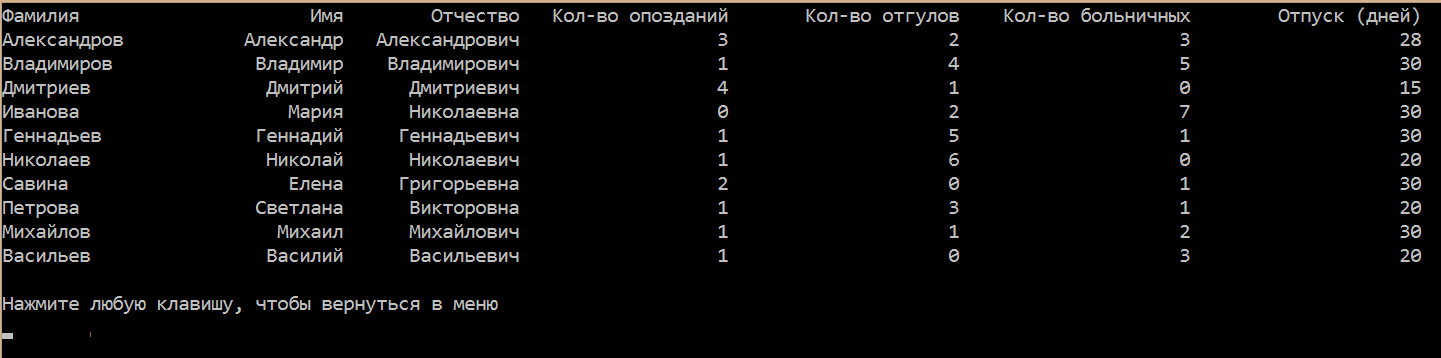


Рис. 24. Восьмой запрос

Выполнение девятого, десятого, одиннадцатого и двенадцатого запросов представлено на рис. 17-22:



Рис. 25

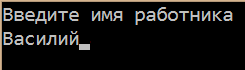


Рис. 26

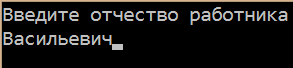


Рис. 27



Рис. 28

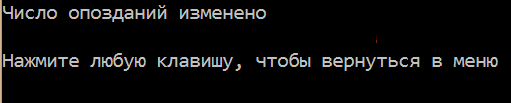


Рис. 29



Рис. 30

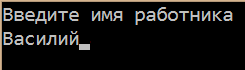


Рис. 31

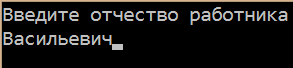


Рис. 32

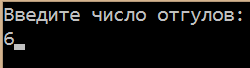


Рис. 33

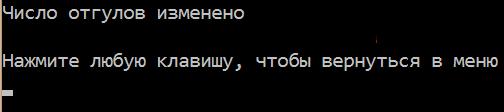


Рис. 34



Рис. 35

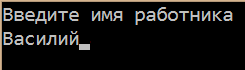


Рис. 36

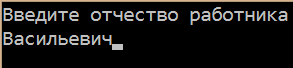


Рис. 37

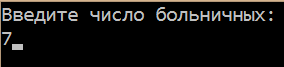


Рис. 38

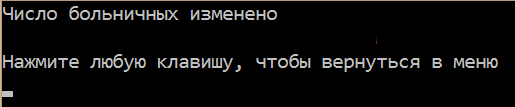


Рис. 39



Рис. 40

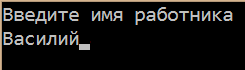


Рис. 41

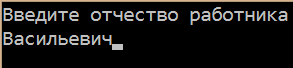


Рис. 42



Рис. 43

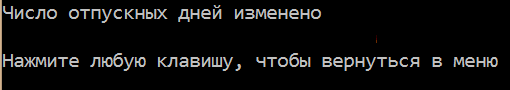


Рис. 44

Теперь у Васильева Василия Васильевича измененное количество опозданий, отгулов, больничных и дней отпуска:

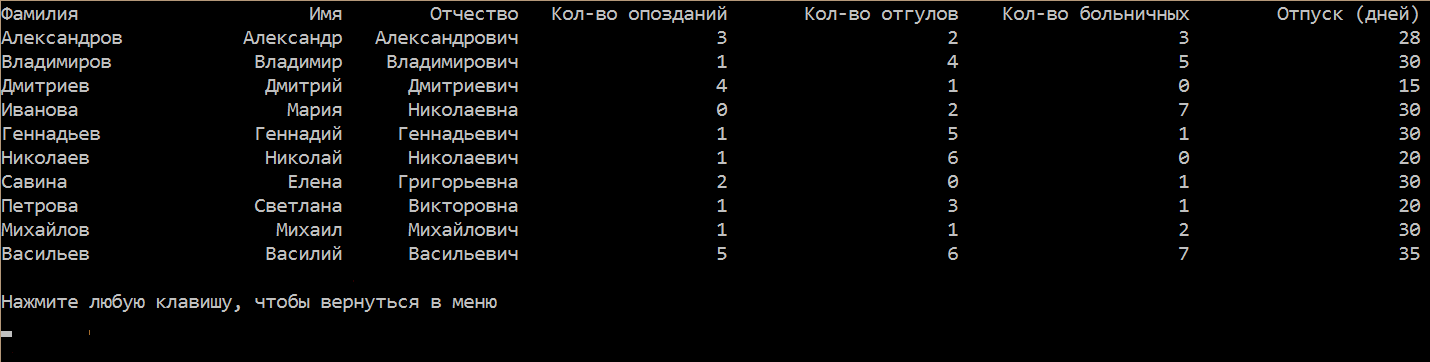


Рис. 45. Новые данные

Во всех запросах, где нужно посмотреть или изменить данные конкретного заданного работника, при неправильном вводе имени, фамилии или отчества программа выдает сообщение о том, что такой работник не найден:

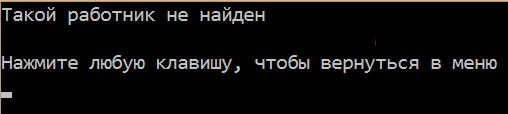


Рис. 46. Работник не найден

Выполнение тринадцатого запроса представлено на рис. 47-59:



Рис. 47



Рис. 48

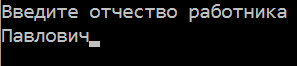


Рис. 49

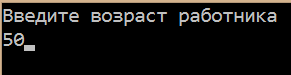


Рис. 50

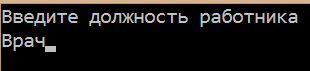


Рис. 51



Рис. 52

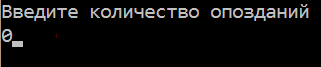


Рис. 53

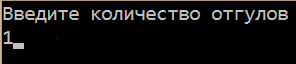


Рис. 54

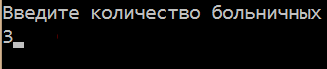


Рис. 55

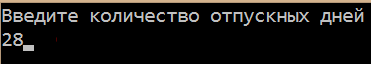


Рис. 56

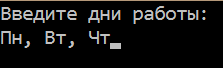


Рис. 57



Рис. 58

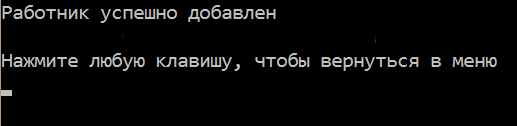


Рис. 59

Теперь добавленный работник Павлов Павел Павлович отображается во всех таблицах со всеми введенными данными:

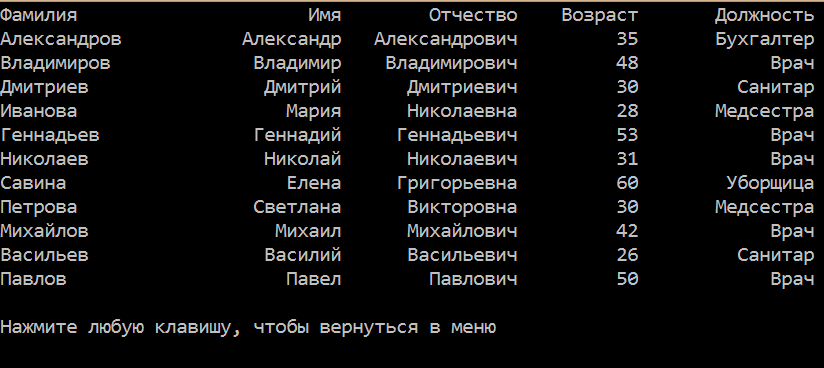


Рис. 60

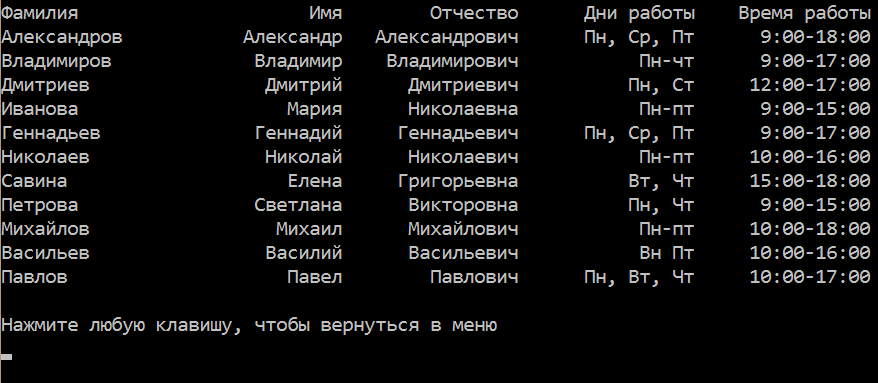


Рис. 61

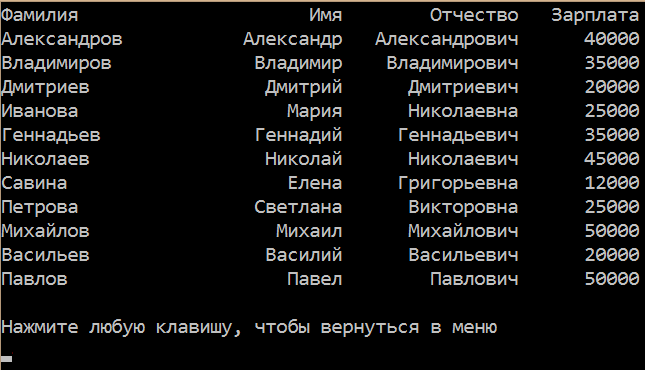


Рис. 62

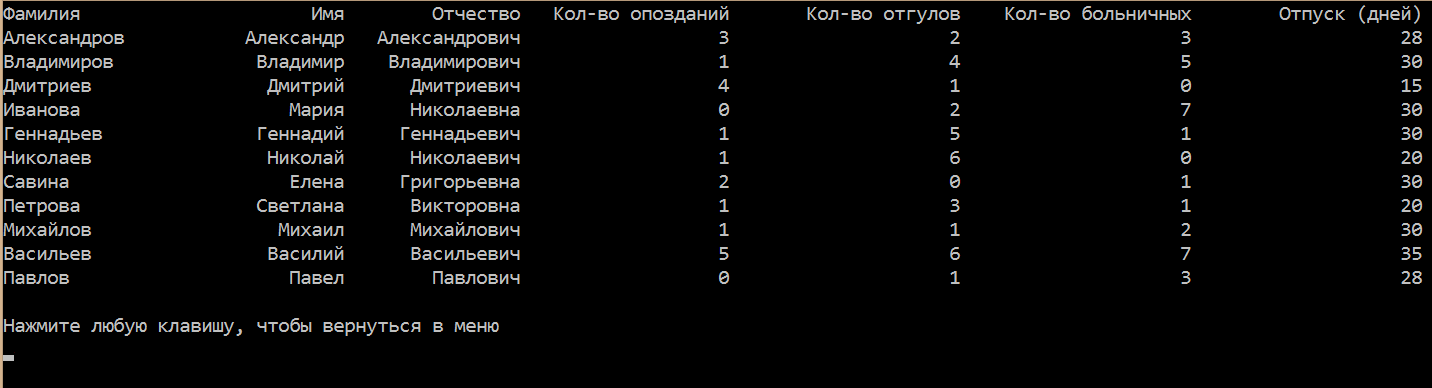


Рис. 63

При успешном завершении программы создаются выходные документы. Все работники с их зарплатой, опозданиями, отгулами, больничными, историями болезней и отпусками сохраняются в файле employees.xml. Работники с их графиками работы сохраняются в файл shedule.json.

Часть файла employees.xml представлена на рис. 64:



Рис. 64. файл employees.xml

Часть файла shedule.json представлена на рис. 65:

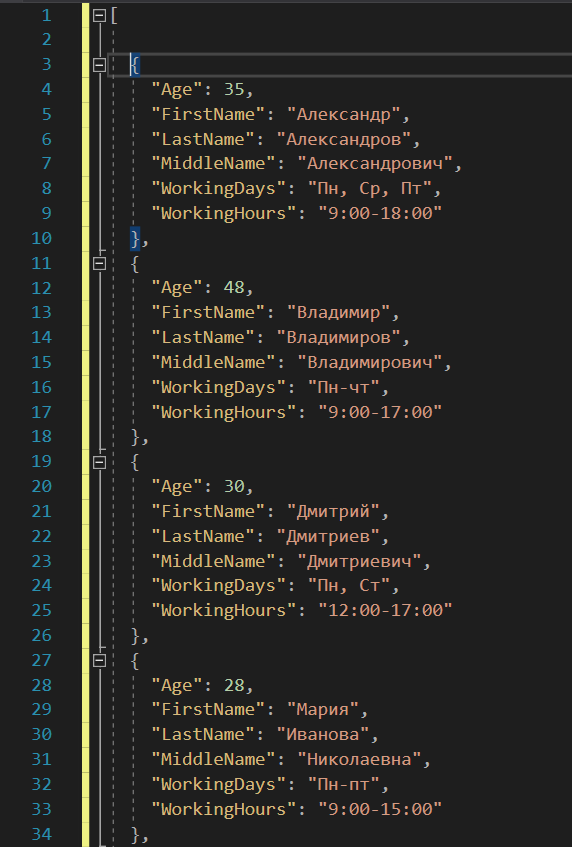


Рис. 65. файл shedule.json

# ЗАКЛЮЧЕНИЕ

Программа успешно реализована и работает в соответствии с поставленными задачами. Реализованы такие запросы, как список всех работников, список работников по должностям, список работников с графиком работы, список работников с заработными платами, посмотреть историю болезни, список опозданий, отгулов, больничных, отпусков. Также программа выполняет такие функции, как добавить болезнь в историю болезни работника, изменить график работы заданного работника, изменить количество опозданий, изменить количество отгулов, изменить количество больничных, изменить количество отпусков, добавить работника. Создан файл employees.xml, хранящий работников, их зарплату, опоздания, отгулы, больничные, истории болезней и отпуска, а также файл shedule.json, хранящий работников и их графики работы. В программе используются такие принципы ОПП, как инкапсуляция (созданы классы на основе сущностей), наследование (класс Person является базовым для классов Employee и WorkShedule) и полиморфизм (в классе Person определены виртуальные методы GetAllEmployees() и GetShedule(), которые переопределены в классе Employee).

# ПРИЛОЖЕНИЕ

Листинг программы

using System;

using System.Collections.Generic;

using System.IO;

using System.Xml.Serialization;

using System.Runtime.Serialization;

using System.Runtime.Serialization.Json;

namespace ConsoleApp17

{

[DataContract]

public class Person

{

[DataMember]

public string FirstName { get; set; }

[DataMember]

public string LastName { get; set; }

[DataMember]

public string MiddleName { get; set; }

[DataMember]

public int Age { get; set; }

public virtual void GetAllEmployees() { }

public virtual void GetShedule() { }

}

[DataContract]

public class WorkSchedule : Person

{

[DataMember]

public string WorkingDays { get; set; }

[DataMember]

public string WorkingHours { get; set; }

}

[DataContract]

public class Employee : Person

{

static public List<Employee> employees = new List<Employee>();

static public List<WorkSchedule> shedule = new List<WorkSchedule>();

public string Position { get; set; }

public int Salary { get; set; }

public int DelayCount { get; set; }

public int CompensatoryTimeCount { get; set; }

public int SickDayCount { get; set; }

public int VacationCount { get; set; }

public List<string> CaseHistory;

public override void GetAllEmployees()

{

Console.Clear();

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 10} {4, 15}", "Фамилия", "Имя", "Отчество", "Возраст", "Должность");

foreach (var employee in employees)

{

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 10} {4, 15}", employee.LastName, employee.FirstName, employee.MiddleName, employee.Age, employee.Position);

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void GetEmployeesByPosition()

{

Console.Clear();

Console.WriteLine("Выберите должность работников, которых вывести на экран:");

var position = Console.ReadLine();

Console.Clear();

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 10}", "Фамилия", "Имя", "Отчество", "Возраст");

var employeesCount = 0;

foreach (var employee in employees)

{

if (employee.Position == position)

{

employeesCount++;

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 10}", employee.LastName, employee.FirstName, employee.MiddleName, employee.Age);

}

}

if (employeesCount == 0)

{

Console.Clear();

Console.WriteLine("Работников с такой должностью не найдено");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public override void GetShedule()

{

Console.Clear();

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 15} {4, 15}", "Фамилия", "Имя", "Отчество", "Дни работы", "Время работы");

foreach (var employee in shedule)

{

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 15} {4, 15}", employee.LastName, employee.FirstName, employee.MiddleName, employee.WorkingDays, employee.WorkingHours);

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void GetSalary()

{

Console.Clear();

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 10}", "Фамилия", "Имя", "Отчество", "Зарплата");

foreach (var employee in employees)

{

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 10}", employee.LastName, employee.FirstName, employee.MiddleName, employee.Salary);

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void SetDisease()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach(var employee in employees)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

string disease = null;

do

{

Console.Clear();

Console.WriteLine("Введите название болезни");

disease = Console.ReadLine();

if (disease.Length != 0)

{

employee.CaseHistory.Add(disease);

Console.Clear();

Console.WriteLine("Болезнь добавлена в историю болезни");

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

Console.ReadKey();

}

else

{

Console.Clear();

break;

}

} while (true);

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void GetDisease()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach (var employee in employees)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

Console.WriteLine(employee.LastName + " " + employee.FirstName + " " + employee.MiddleName);

Console.WriteLine("\nИстория болезни:");

for (int i = 0; i < employee.CaseHistory.Count; i++)

{

Console.WriteLine(employee.CaseHistory[i]);

}

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void ChangeShedule()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach (var employee in shedule)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

Console.WriteLine("Введите дни работы:");

var workingDays = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите часы работы:");

var workingHours = Console.ReadLine();

Console.Clear();

employee.WorkingDays = workingDays;

employee.WorkingHours = workingHours;

Console.WriteLine("График работы добавлен");

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void GetDelayCompensatorySickVacation()

{

Console.Clear();

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 18} {4, 20} {5, 20} {6, 20}", "Фамилия", "Имя", "Отчество", "Кол-во опозданий", "Кол-во отгулов", "Кол-во больничных", "Отпуск (дней)");

foreach (var employee in employees)

{

Console.WriteLine("{0, -20} {1, 10} {2, 15} {3, 18} {4, 20} {5, 20} {6, 20}", employee.LastName, employee.FirstName, employee.MiddleName, employee.DelayCount, employee.CompensatoryTimeCount, employee.SickDayCount, employee.VacationCount);

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void ChangeDelay()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach (var employee in employees)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

Console.WriteLine("Введите число опозданий:");

var delayCount = 0;

do

{

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

} while (delayCount < 0);

employee.DelayCount = delayCount;

Console.Clear();

Console.WriteLine("Число опозданий изменено");

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void ChangeCompensatory()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach (var employee in employees)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

Console.WriteLine("Введите число отгулов:");

var сompensatoryCount = 0;

do

{

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

} while (сompensatoryCount < 0);

employee.CompensatoryTimeCount = сompensatoryCount;

Console.Clear();

Console.WriteLine("Число отгулов изменено");

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void Sick()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach (var employee in employees)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

Console.WriteLine("Введите число больничных:");

var sickCount = 0;

do

{

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

} while (sickCount < 0);

employee.SickDayCount = sickCount;

Console.Clear();

Console.WriteLine("Число больничных изменено");

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void Vacation()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

var findEmployee = false;

foreach (var employee in employees)

{

if (employee.LastName == lastName && employee.FirstName == firstName && employee.MiddleName == middleName)

{

findEmployee = true;

Console.WriteLine("Введите число отпускных дней:");

var vacationCount = 0;

do

{

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

} while (vacationCount < 0);

employee.VacationCount = vacationCount;

Console.Clear();

Console.WriteLine("Число отпускных дней изменено");

}

}

if (!findEmployee)

{

Console.WriteLine("Такой работник не найден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public void AddEmployee()

{

Console.Clear();

Console.WriteLine("Введите фамилию работника");

var lastName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите имя работника");

var firstName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите отчество работника");

var middleName = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите возраст работника");

var age = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите должность работника");

var position = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите зарплату работника");

var salary = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите количество опозданий");

var delay = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите количество отгулов");

var сompensatory = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите количество больничных");

var sick = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите количество отпускных дней");

var vacation = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите дни работы:");

var workingDays = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите часы работы:");

var workingHours = Console.ReadLine();

Console.Clear();

employees.Add(new Employee { LastName = lastName, FirstName = firstName, MiddleName = middleName, Age = age, Position = position, Salary = salary, DelayCount = delay, CompensatoryTimeCount = сompensatory, SickDayCount = sick, VacationCount = vacation });

shedule.Add(new WorkSchedule { LastName = lastName, FirstName = firstName, MiddleName = middleName, Age = age, WorkingDays = workingDays, WorkingHours = workingHours });

Console.WriteLine("Работник успешно добавлен");

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

static public void Menu()

{

Console.WriteLine("1. Список всех работников");

Console.WriteLine("2. Список работников по должностям");

Console.WriteLine("3. Список работников с графиком работы");

Console.WriteLine("4. Список работников с заработными платами");

Console.WriteLine("5. Добавить болезнь в историю болезни работника");

Console.WriteLine("6. Посмотреть историю болезни");

Console.WriteLine("7. Изменить график работы заданного работника");

Console.WriteLine("8. Список опозданий, отгулов, больничных, отпусков");

Console.WriteLine("9. Изменить количество опозданий");

Console.WriteLine("10. Изменить количество отгулов");

Console.WriteLine("11. Изменить количество больничных");

Console.WriteLine("12. Изменить количество отпусков");

Console.WriteLine("13. Добавить работника");

Console.WriteLine("0. Выход");

var employee = new Employee();

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

switch (command)

{

case 1:

{

employee.GetAllEmployees();

break;

}

case 2:

{

employee.GetEmployeesByPosition();

break;

}

case 3:

{

employee.GetShedule();

break;

}

case 4:

{

employee.GetSalary();

break;

}

case 5:

{

employee.SetDisease();

break;

}

case 6:

{

employee.GetDisease();

break;

}

case 7:

{

employee.ChangeShedule();

break;

}

case 8:

{

employee.GetDelayCompensatorySickVacation();

break;

}

case 9:

{

employee.ChangeDelay();

break;

}

case 10:

{

employee.ChangeCompensatory();

break;

}

case 11:

{

employee.Sick();

break;

}

case 12:

{

employee.Vacation();

break;

}

case 13:

{

employee.AddEmployee();

break;

}

}

}

}

class Program

{

static XmlSerializer xmlFormatter = new XmlSerializer(typeof(List<Employee>));

static DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(List<WorkSchedule>));

static public void Serialization()

{

using (var file = new FileStream("employees.xml", FileMode.OpenOrCreate))

{

xmlFormatter.Serialize(file, Employee.employees);

}

using (var shedule = new FileStream("shedule.json", FileMode.OpenOrCreate))

{

jsonFormatter.WriteObject(shedule, Employee.shedule);

}

}

static public void Deserialization()

{

using (var file = new FileStream("employees.xml", FileMode.OpenOrCreate))

{

Employee.employees = xmlFormatter.Deserialize(file) as List<Employee>;

}

using (var shedule = new FileStream("shedule.json", FileMode.OpenOrCreate))

{

Employee.shedule = jsonFormatter.ReadObject(shedule) as List<WorkSchedule>;

}

}

static void Main(string[] args)

{

Deserialization();

Employee.Menu();

Serialization();

}

}

}