Отчёт

Алексеев Иван

1. Описать классы "Работник" и "Предприятие" позволяющие:

* создавать работников,
* принимать,
* увольнять,
* изменять должность,
* переводить на другое предприятие,
* изменять зарплату.

1. Предусмотреть текстовое меню, обеспечивающее доступ ко всем ее функциям
2. Предусмотреть защиту от ввода некорректных данных для всех членов всех классов
3. Предусмотреть повторное предложение Пользователю на ввод данных, в случае их некорректности
4. Предусмотреть назначение Работникам и Автомобилям уникальных идентификаторов
5. Предусмотреть выбор Предприятия на прием и перевод работников

3) Доработать программу, написать класс для сохранения состояния программы в текстовом файле, на примере стоянки:

место; модель; цвет; номер; дата прибытия

1; ваз 2107; белый; а123бв; 20:00 05.10.2023

2; ...

при изменении данных перезаписывать файл,

при новом запуске программы восстанавливать данные из файла

4) написать класс для логирования событий в текстовый файл, например:

20:00 05.10.2023: прибытие автомобиля ваз 2107

21:30 05.10.2023: убытие автомобиля ваз 2107

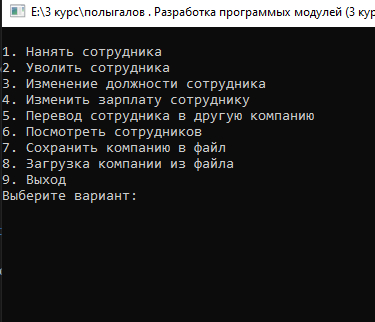


Рисунок 1 - текстовое меню

using System;

using System.Collections.Generic;

using System.IO;

class Employee

{

public int EmployeeId { get; private set; }

public string Name { get; set; }

public string Position { get; set; }

public double Salary { get; set; }

public Employee(int employeeId, string name, string position, double salary)

{

EmployeeId = employeeId;

Name = name;

Position = position;

Salary = salary;

}

}

class Company

{

private List<Employee> employees = new List<Employee>();

public void HireEmployee(int employeeId, string name, string position, double salary)

{

Employee employee = new Employee(employeeId, name, position, salary);

employees.Add(employee);

}

public void FireEmployee(int employeeId)

{

Employee employee = employees.Find(e => e.EmployeeId == employeeId);

if (employee != null)

{

employees.Remove(employee);

Console.WriteLine($"Сотрудник с идентификатором {employeeId} был уволен.");

}

else

{

Console.WriteLine($"Сотрудник с идентификатором {employeeId} не найден.");

}

}

public void ChangePosition(int employeeId, string newPosition)

{

Employee employee = employees.Find(e => e.EmployeeId == employeeId);

if (employee != null)

{

employee.Position = newPosition;

Console.WriteLine($"Должность сотрудника с идентификатором {employeeId} была изменена на {newPosition}.");

}

else

{

Console.WriteLine($"Сотрудник с идентификатором {employeeId} не найден.");

}

}

public void ChangeSalary(int employeeId, double newSalary)

{

Employee employee = employees.Find(e => e.EmployeeId == employeeId);

if (employee != null)

{

employee.Salary = newSalary;

Console.WriteLine($"Зарплата сотрудника с идентификатором {employeeId} была изменена на {newSalary}.");

}

else

{

Console.WriteLine($"Сотрудник с идентификатором {employeeId} не найден.");

}

}

public void TransferToAnotherCompany(int employeeId, Company newCompany)

{

Employee employee = employees.Find(e => e.EmployeeId == employeeId);

if (employee != null)

{

newCompany.HireEmployee(employee.EmployeeId, employee.Name, employee.Position, employee.Salary);

employees.Remove(employee);

Console.WriteLine($"Сотрудник с идентификатором {employeeId} был переведен в другую компанию.");

}

else

{

Console.WriteLine($"Сотрудник с идентификатором {employeeId} не найден.");

}

}

public void ViewEmployees()

{

foreach (var employee in employees)

{

Console.WriteLine($"ID: {employee.EmployeeId}, Имя: {employee.Name}, Должность: {employee.Position}, Зарплата: {employee.Salary}");

}

}

public void SaveToFile(string filename)

{

using (StreamWriter writer = new StreamWriter(filename))

{

foreach (var employee in employees)

{

writer.WriteLine($"{employee.EmployeeId};{employee.Name};{employee.Position};{employee.Salary}");

}

}

}

public void LoadFromFile(string filename)

{

if (File.Exists(filename))

{

string[] lines = File.ReadAllLines(filename);

foreach (string line in lines)

{

string[] data = line.Split(';');

if (data.Length == 4)

{

int id = int.Parse(data[0]);

string name = data[1].Trim();

string position = data[2].Trim();

double salary = double.Parse(data[3]);

Employee employee = new Employee(id, name, position, salary);

employees.Add(employee);

}

}

}

}

}

class Program

{

static void Main()

{

Company company = new Company();

while (true)

{

Console.WriteLine("\n1. Нанять сотрудника");

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

Console.WriteLine("3. Изменение должности сотрудника");

Console.WriteLine("4. Изменить зарплату сотруднику");

Console.WriteLine("5. Перевод сотрудника в другую компанию");

Console.WriteLine("6. Посмотреть сотрудников");

Console.WriteLine("7. Сохранить компанию в файл");

Console.WriteLine("8. Загрузка компании из файла");

Console.WriteLine("9. Выход");

Console.Write("Выберите вариант: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

try

{

Console.Write("Введите идентификатор сотрудника: ");

int id;

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

Console.Write("Введите идентификатор сотрудника: ");

}

Console.Write("Введите имя сотрудника: ");

string name = Console.ReadLine();

Console.Write("Введите должность сотрудника: ");

string position = Console.ReadLine();

Console.Write("Введите зарплату сотрудника: ");

double salary;

while (!double.TryParse(Console.ReadLine(), out salary))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите правильное двойное значение для Зарплата сотрудника.");

Console.Write("Введите зарплату сотрудника: ");

}

company.HireEmployee(id, name, position, salary);

Console.WriteLine($"Сотрудник {id} был принят на работу.");

}

catch (Exception e)

{

Console.WriteLine($"Ошибка при приеме сотрудника на работу: {e.Message}");

}

break;

case "2":

try

{

Console.Write("Введите идентификатор сотрудника для увольнения: ");

int id;

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

Console.Write("Введите идентификатор сотрудника для увольнения: ");

}

company.FireEmployee(id);

}

catch (FormatException)

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

}

break;

case "3":

try

{

Console.Write("Введите идентификатор сотрудника, чтобы изменить должность: ");

int id;

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

Console.Write("Введите идентификатор сотрудника, чтобы изменить должность: ");

}

Console.Write("Введите новую позицию: ");

string newPosition = Console.ReadLine();

company.ChangePosition(id, newPosition);

}

catch (FormatException)

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

}

break;

case "4":

try

{

Console.Write("Введите идентификатор сотрудника, чтобы изменить зарплату: ");

int id;

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

Console.Write("Введите идентификатор сотрудника, чтобы изменить зарплату: ");

}

Console.Write("Введите новую зарплату: ");

double newSalary;

while (!double.TryParse(Console.ReadLine(), out newSalary))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите правильное двойное значение для новой зарплаты.");

Console.Write("Введите новую зарплату: ");

}

company.ChangeSalary(id, newSalary);

}

catch (FormatException)

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

}

break;

case "5":

try

{

Console.Write("Введите идентификатор сотрудника для перевода: ");

int id;

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

Console.Write("Введите идентификатор сотрудника для перевода: ");

}

company.TransferToAnotherCompany(id, new Company());

}

catch (FormatException)

{

Console.WriteLine("Неверный ввод. Пожалуйста, введите действительное целое число для идентификатора сотрудника.");

}

break;

case "6":

company.ViewEmployees();

break;

case "7":

Console.Write("Введите имя файла для сохранения данных компании: ");

string saveFileName = Console.ReadLine();

company.SaveToFile(saveFileName);

Console.WriteLine($"Данные компании сохранены в {saveFileName}.");

break;

case "8":

Console.Write("Введите имя файла для загрузки данных о компании: ");

string loadFileName = Console.ReadLine();

company.LoadFromFile(loadFileName);

Console.WriteLine($"Данные о компании загружены из {loadFileName}.");

break;

case "9":

Environment.Exit(0);

break;

default:

Console.WriteLine("Неверный вариант. Пожалуйста, выберите правильный вариант.");

break;

}

}

}

}