Масленников

Отчёт по лабораторной работе  
**Тема: "Работник" и "Предприятие"**

**Цель работы**: разработка программы для учета работников, компаний, и их взаимодействия

**Задачи**:

Описать классы "Работник" и "Предприятие" позволяющие:

создавать работников,

принимать,

увольнять,

изменять должность,

переводить на другое предприятие,

изменять зарплату.

Предусмотреть защиту от ввода некорректных данных для всех членов всех классов

Предусмотреть повторное предложение Пользователю на ввод данных, в случае их некорректности

Предусмотреть назначение Работникам уникальных идентификаторов

Предусмотреть выбор Предприятия на приема и переода работников

Написать класс для сохранения состояния программы в текстовом файле, на примере стоянки:

при изменении данных перезаписывать файл,

при новом запуске программы восстанавливать данные из файла

написать класс для логирования событий в текстовый файл

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

class Worker

{

public string ID { get; }

public string Name { get; set; }

public string Position { get; set; }

public double Salary { get; set; }

public Worker(string id, string name, string position, double salary)

{

ID = id;

Name = name;

Position = position;

Salary = salary;

}

}

class Enterprise

{

private List<Worker> workers = new List<Worker>();

private FileHandler fileHandler = new FileHandler();

private Logger logger = new Logger();

public Enterprise()

{

workers = fileHandler.ReadFromFile();

}

public void HireWorker(Worker worker)

{

workers.Add(worker);

fileHandler.SaveToFile(workers);

logger.LogEntry($"Прием работника {worker.Name} на предприятие");

}

public void FireWorker(string workerID)

{

Worker worker = workers.FirstOrDefault(w => w.ID == workerID);

if (worker != null)

{

workers.Remove(worker);

fileHandler.SaveToFile(workers);

logger.LogEntry($"Увольнение работника {worker.Name} с предприятия");

}

else

{

Console.WriteLine("Работник с таким ID не найден.");

}

}

public void ChangePosition(string workerID, string newPosition)

{

Worker worker = workers.FirstOrDefault(w => w.ID == workerID);

if (worker != null)

{

worker.Position = newPosition;

fileHandler.SaveToFile(workers);

logger.LogEntry($"Изменение должности работнику {worker.Name}: {newPosition}");

}

else

{

Console.WriteLine("Работник с таким ID не найден.");

}

}

public void ChangeSalary(string workerID, double newSalary)

{

Worker worker = workers.FirstOrDefault(w => w.ID == workerID);

if (worker != null)

{

worker.Salary = newSalary;

fileHandler.SaveToFile(workers);

logger.LogEntry($"Изменение зарплаты работнику {worker.Name}: {newSalary}");

}

else

{

Console.WriteLine("Работник с таким ID не найден.");

}

}

public void TransferWorker(string workerID, Enterprise newEnterprise)

{

Worker worker = workers.FirstOrDefault(w => w.ID == workerID);

if (worker != null)

{

workers.Remove(worker);

newEnterprise.HireWorker(worker);

logger.LogEntry($"Перевод работника {worker.Name} на другое предприятие");

}

else

{

Console.WriteLine("Работник с таким ID не найден.");

}

}

public void ViewWorkers()

{

foreach (var worker in workers)

{

Console.WriteLine($"\nID: {worker.ID}\nИмя: {worker.Name}\nДолжность: {worker.Position}\nЗарплата: {worker.Salary}\n");

}

}

}

class FileHandler

{

private string filePath = "workers\_data.txt";

public void SaveToFile(List<Worker> workers)

{

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (var worker in workers)

{

writer.WriteLine($"{worker.ID};{worker.Name};{worker.Position};{worker.Salary}");

}

}

}

public List<Worker> ReadFromFile()

{

List<Worker> workers = new List<Worker>();

if (File.Exists(filePath))

{

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

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

if (data.Length == 4)

{

Worker worker = new Worker(data[0], data[1], data[2], double.Parse(data[3]));

workers.Add(worker);

}

}

}

}

return workers;

}

}

class Logger

{

private string logFilePath = "workers\_log.txt";

public void LogEntry(string logEntry)

{

using (StreamWriter writer = new StreamWriter(logFilePath, true))

{

writer.WriteLine($"{DateTime.Now}: {logEntry}");

}

}

}

class Program

{

static void Main(string[] args)

{

Enterprise enterprise = new Enterprise();

Enterprise anotherEnterprise = new Enterprise(); // Для перевода работника на другое предприятие

while (true)

{

Console.WriteLine("Выберите действие:");

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

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

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

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

Console.WriteLine("5. Перевести работника на другое предприятие");

Console.WriteLine("6. Посмотреть информацию о работниках");

Console.WriteLine("7. Выход");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.WriteLine("Введите ID, имя, должность и зарплату работника (через пробел):");

string[] hireInput = Console.ReadLine().Split(' ');

if (hireInput.Length == 4)

{

string workerID = hireInput[0];

string name = hireInput[1];

string position = hireInput[2];

double salary;

if (double.TryParse(hireInput[3], out salary))

{

Worker newWorker = new Worker(workerID, name, position, salary);

enterprise.HireWorker(newWorker);

Console.WriteLine("Работник успешно нанят.");

}

else

{

Console.WriteLine("Некорректная зарплата.");

}

}

else

{

Console.WriteLine("Некорректные данные.");

}

break;

case "2":

Console.WriteLine("Введите ID работника для увольнения:");

string workerToFire = Console.ReadLine();

enterprise.FireWorker(workerToFire);

break;

case "3":

Console.WriteLine("Введите ID работника и новую должность (через пробел):");

string[] positionInput = Console.ReadLine().Split(' ');

if (positionInput.Length == 2)

{

enterprise.ChangePosition(positionInput[0], positionInput[1]);

}

else

{

Console.WriteLine("Некорректные данные.");

}

break;

case "4":

Console.WriteLine("Введите ID работника и новую зарплату (через пробел):");

string[] salaryInput = Console.ReadLine().Split(' ');

if (salaryInput.Length == 2)

{

double newSalary;

if (double.TryParse(salaryInput[1], out newSalary))

{

enterprise.ChangeSalary(salaryInput[0], newSalary);

}

else

{

Console.WriteLine("Некорректная зарплата.");

}

}

else

{

Console.WriteLine("Некорректные данные.");

}

break;

case "5":

Console.WriteLine("Введите ID работника для перевода и ID другого предприятия (через пробел):");

string[] transferInput = Console.ReadLine().Split(' ');

if (transferInput.Length == 2)

{

Enterprise targetEnterprise = transferInput[1] == "1" ? enterprise : anotherEnterprise;

targetEnterprise.TransferWorker(transferInput[0], targetEnterprise);

}

else

{

Console.WriteLine("Некорректные данные.");

}

break;

case "6":

Console.WriteLine("Информация о работниках:");

enterprise.ViewWorkers();

break;

case "7":

Environment.Exit(0);

break;

default:

Console.WriteLine("Неверный выбор. Пожалуйста, выберите снова.");

break;

}

}

}

}