

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра ПЗ

Практична робота №2-4

з дисципліни «Розробка проектів засобами платформи .NET»

Виконав: ст. 5-ПІ-23Б

Козловський Д.В.

Перевірив: асистент

Малініч П.П.

Практична робота №2-4

Завдання 1: Імплементувати збереження даних про задачі у текстовий файл work-items.json

Завдання 2: Розширити функціонал консольного інтерфейсу

Завдання 3: Організувати проекти за допомогою solution folders

Завдання 4: Dependency Injection

Завдання 5: Ігнорувати виконані завдання при створенні плану виконання задач

Завдання 6: Модульне тестування

Код програми

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Kozlovskiy.TaskPlanner.Domain.Models;
using Kozlovskiy.TaskPlanner.Domain.Logic;
using Kozlovskiy.TaskPlanner.DataAccess;
using Kozlovskiy.TaskPlanner.DataAccess.Abstractions;

namespace Kozlovskiy.TaskPlanner
{
    internal static class Program
    {
        private static Guid ReadGuid(string prompt)
        {
            while (true)
            {
                Console.Write(prompt);
                if (Guid.TryParse(Console.ReadLine(), out Guid id))
                {
                    return id;
                }
            }
        }
    }
}
```

```
        }

        Console.WriteLine("Некоректний формат GUID. Спробуйте ще раз.");
    }

}

private static void ShowMenu()
{
    Console.WriteLine("\n--- Меню ---");
    Console.WriteLine("1 - Додати завдання");
    Console.WriteLine("2 - Показати план");
    Console.WriteLine("3 - Позначити завдання як виконане");
    Console.WriteLine("4 - Видалити завдання");
    Console.WriteLine("5 - Завершити роботу");
    Console.Write("Ваш вибір: ");

}

private static void AddWorkItem(IWorkItemsRepository repository)
{
    Console.WriteLine("\n--- Додавання нового завдання ---");

    Console.Write("Введіть називу завдання: ");
    string title = Console.ReadLine();

    DateTime dueDate;
    while (true)
    {
        Console.Write("Введіть дату виконання (рррр-мм-дд): ");
        if (DateTime.TryParse(Console.ReadLine(), out dueDate)) break;
        Console.WriteLine("Некоректний формат дати. Спробуйте ще раз.");
    }

    Priority priority;
    while (true)
    {
        Console.WriteLine("Оберіть пріоритет:");
        foreach (var pr in Enum.GetValues(typeof(Priority)))

```

```

{
    Console.WriteLine($"- {pr}");
}

Console.Write("Введіть пріоритет: ");
string priorityInput = Console.ReadLine();

if (Enum.TryParse(priorityInput, true, out priority)) break;
Console.WriteLine("Некоректний пріоритет. Спробуйте ще раз.");
}

WorkItem newItem = new WorkItem(title, dueDate, priority);

repository.Add(newItem);
repository.SaveChanges();

Console.WriteLine("\nНовий елемент додано та збережено.\n");
}

private static void BuildPlan(SimpleTaskPlanner planner, IWorkItemsRepository repository)
{
    WorkItem[] sortedItems = planner.CreatePlan();

    Console.WriteLine("\n--- План виконання завдань (відсортований) ---");
    if (sortedItems.Any())
    {
        Console.WriteLine($"{{\"ID\",-36} | {"Пріоритет",-10} | {"Дата",-12} | {"Завершено",-10} | {"Назва"} }");
        Console.WriteLine(new string('-', 90));

        foreach (var item in sortedItems)
        {
            string completedStatus = item.IsCompleted ? "Так" : "Ні";
            Console.WriteLine($"{{item.Id,-36} | {item.priority,-10} | {item.DueDate.ToShortDateString(),-12} | {completedStatus,-10} | {item.Title}}");
        }
    }
    else

```

```
{  
    Console.WriteLine("Немає завдань для відображення.");  
}  
}  
  
private static void MarkCompleted(IWorkItemsRepository repository)  
{  
    Console.WriteLine("\n--- Позначити завдання як виконане ---");  
    Guid id = ReadGuid("Введіть ID завдання, яке потрібно позначити як виконане: ");  
  
    WorkItem itemToUpdate = repository.Get(id);  
  
    if (itemToUpdate == null)  
    {  
        Console.WriteLine($"Завдання з ID {id} не знайдено.");  
        return;  
    }  
  
    if (itemToUpdate.IsCompleted)  
    {  
        Console.WriteLine("Це завдання вже позначено як виконане.");  
        return;  
    }  
  
    itemToUpdate.IsCompleted = true;  
  
    if (repository.Update(itemToUpdate))  
    {  
        repository.SaveChanges();  
        Console.WriteLine($"Завдання '{itemToUpdate.Title}' успішно позначено як виконане та збережено.");  
    }  
    else  
    {  
        Console.WriteLine("Помилка при оновленні завдання.");  
    }  
}
```

```
private static void RemoveWorkItem(IWorkItemsRepository repository)
{
    Console.WriteLine("\n--- Видалити завдання ---");
    Guid id = ReadGuid("Введіть ID завдання, яке потрібно видалити: ");

    if (repository.Remove(id))
    {
        repository.SaveChanges();
        Console.WriteLine($"Завдання з ID {id} успішно видалено та збережено.");
    }
    else
    {
        Console.WriteLine($"Завдання з ID {id} не знайдено.");
    }
}

static void Main(string[] args)
{
    Console.OutputEncoding = System.Text.Encoding.UTF8;

    IWorkItemsRepository repository = new FileWorkItemsRepository();

    SimpleTaskPlanner planner = new SimpleTaskPlanner(repository);

    bool running = true;
    while (running)
    {
        ShowMenu();
        string choice = Console.ReadLine()?.Trim().ToUpper();

        switch (choice)
        {
            case "1":
                AddWorkItem(repository);
                break;
        }
    }
}
```

```

        case "2":
            BuildPlan(planner, repository);
            break;

        case "3":
            MarkCompleted(repository);
            break;

        case "4":
            RemoveWorkItem(repository);
            break;

        case "5":
            running = false;
            Console.WriteLine("Збереження змін та завершення програми...");
            repository.SaveChanges();
            break;

        default:
            Console.WriteLine("Невідома команда. Введіть 1, 2, 3, 4 або 5.");
            break;
    }

}

Console.WriteLine("Програма завершена.");
}
}
}

```

Workitem.cs

```

using Microsoft.VisualBasic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```
using System.Threading.Tasks;

namespace Kozlovskiy.TaskPlanner.Domain.Models

{
    public class WorkItem
    {
        public Guid Id { get; set; }

        public DateTime CreationDate;
        public DateTime DueDate;
        public Priority priority;
        public Complexity complexity;
        public string Title;
        public string Description;
        public bool IsCompleted;

        public override string ToString()
        {
            return Title + ": due" + DueDate + ", " + priority.ToString().ToLower();
        }

        public WorkItem(string title, DateTime date, Priority p)
        {
            Id = Guid.Empty;
            Title = title;
            DueDate = date;
            priority = p;
            IsCompleted = false;
            Description = string.Empty;
            CreationDate = DateTime.Now;
        }

        public WorkItem Clone()
        {
            return new WorkItem(Title, DueDate, priority);
        }
    }
}
```

```

    {
        Id = this.Id,
        CreationDate = this.CreationDate,
        complexity = this.complexity,
        Description = this.Description,
        IsCompleted = this.IsCompleted
    };

}

}

```

FileWorkItemsRepository.cs

```

using Kozlovskiy.TaskPlanner.DataAccess.Abstractions;
using Kozlovskiy.TaskPlanner.Domain.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace Kozlovskiy.TaskPlanner.DataAccess
{
    public class FileWorkItemsRepository : IWorkItemsRepository
    {

        private const string WorkItemsFileName = "work-items.json";

        private readonly Dictionary<Guid, WorkItem> _workItems;

        public FileWorkItemsRepository()
        {
            _workItems = new Dictionary<Guid, WorkItem>();

            if (File.Exists(WorkItemsFileName))
            {

```

```
try
{
    string jsonContent = File.ReadAllText(WorkItemsFileName);

    if (!string.IsNullOrWhiteSpace(jsonContent))
    {
        WorkItem[] itemsArray = JsonConvert.DeserializeObject<WorkItem[]>(jsonContent);

        if (itemsArray != null)
        {
            foreach (var item in itemsArray)
            {
                _workItems.Add(item.Id, item);
            }
        }
    }

    catch (Exception ex)
    {
        Console.WriteLine($"Error reading or deserializing {WorkItemsFileName}: {ex.Message}");
    }
}

}

public Guid Add(WorkItem workItem)
{
    WorkItem itemCopy = workItem.Clone();

    Guid newId = Guid.NewGuid();

    itemCopy.Id = newId;
}
```

```
        _workItems.Add(newId, itemCopy);

    return newId;
}

public WorkItem Get(Guid id)
{
    if (_workItems.TryGetValue(id, out WorkItem workItem))
    {
        return workItem.Clone();
    }
    return null;
}

public WorkItem[] GetAll()
{
    return _workItems.Values.Select(item => item.Clone()).ToArray();
}

public bool Update(WorkItem workItem)
{
    if (_workItems.ContainsKey(workItem.Id))
    {
        _workItems[workItem.Id] = workItem.Clone();
        return true;
    }
    return false;
}

public bool Remove(Guid id)
{
    return _workItems.Remove(id);
}
```

```

public void SaveChanges()
{
    WorkItem[] itemsArray = _workItems.Values.ToArray();

    string jsonContent = JsonConvert.SerializeObject(itemsArray, Formatting.Indented);

    File.WriteAllText(WorkItemsFileName, jsonContent);
}
}
}

```

IWorkItemsRepository.cs

```

using System;
using Kozlovskiy.TaskPlanner.Domain.Models;

namespace Kozlovskiy.TaskPlanner.DataAccess.Abstractions
{
    public interface IWorkItemsRepository
    {

        Guid Add(WorkItem workItem);

        WorkItem Get(Guid id);

        WorkItem[] GetAll();

        bool Update(WorkItem workItem);

        bool Remove(Guid id);

        void SaveChanges();
    }
}

```

Результат тестування роботи програми

```
--- Меню ---
1 - Додати завдання
2 - Показати план
3 - Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 1

--- Додавання нового завдання ---
Введіть назву завдання: Купити арбуз
Введіть дату виконання (рррр-мм-дд): 2025-12-17
Оберіть пріоритет:
- None
- Low
- Medium
- High
- Urgent
Введіть пріоритет: Urgent

Новий елемент додано та збережено.

--- Меню ---
1 - Додати завдання
2 - Показати план
3 - Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 2
```

Рисунок 1 – Результат тестування роботи програми

```
--- Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 2

--- План виконання завдань (відсортований) ---
ID | Пріоритет | Дата | Завершено | Назва
---|---|---|---|---
9c106028-ca2e-430f-89d3-ec03f82341b9 | Urgent | 17.12.2025 | Hi | Купити арбуз

--- Меню ---
1 - Додати завдання
2 - Показати план
3 - Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 3

--- Позначити завдання як виконане ---
Введіть ID завдання, яке потрібно позначити як виконане: 9c106028-ca2e-430f-89d3-ec03f82341b9
Завдання 'Купити арбуз' успішно позначене як виконане та збережено.

--- Меню ---
1 - Додати завдання
2 - Показати план
3 - Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 1
```

Рисунок 2 – Результат тестування роботи програми

```

Введіть пріоритет: High
Новий елемент додано та збережено.

--- Меню ---
1 - Додати завдання
2 - Показати план
3 - Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 2

--- План виконання завдань (відсортований) ---
ID | Пріоритет | Дата | Завершено | Назва
-----
f840feaa-9d22-4a67-9b56-16a44a9082af | High | 01.09.2026 | Hi | Захопи

--- Меню ---
1 - Додати завдання
2 - Показати план
3 - Позначити завдання як виконане
4 - Видалити завдання
5 - Завершити роботу
Ваш вибір: 4

--- Видалити завдання ---
Введіть ID завдання, яке потрібно видалити: f840feaa-9d22-4a67-9b56-16a44a9082af
Завдання з ID f840feaa-9d22-4a67-9b56-16a44a9082af успішно видалено та збережено.

```

Рисунок 3 – Результат тестування програми

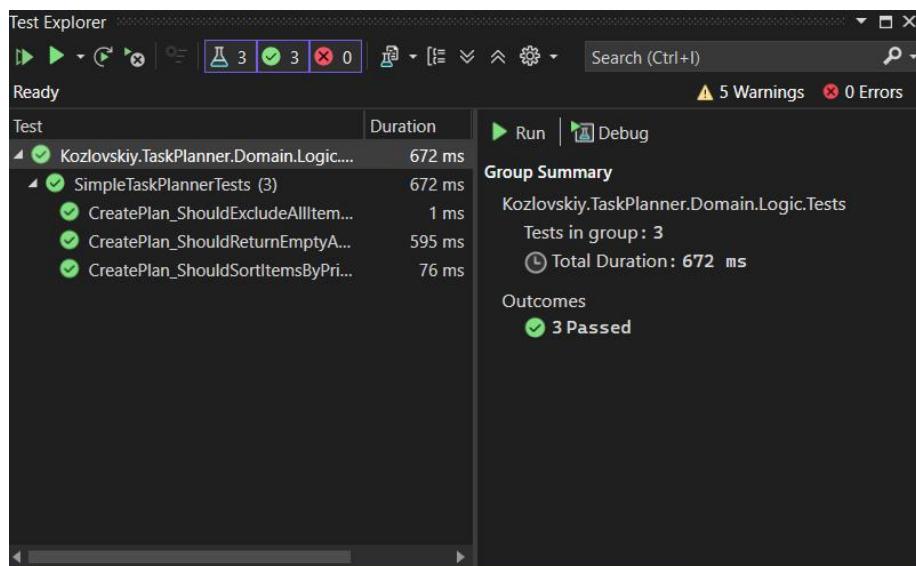


Рисунок 4 – Результат Unit-тестів

Висновок: в результаті виконання лабораторної роботи були поглиблені знання про роботу з фреймворком .Net, була покращена програма TaskPlanner.

Посилання на Git hub: <https://github.com/kozlovskiydimas/TaskPlanner>