

Лабораторна робота №2

Багатопоковість. Асинхроність. IEnumerable. LINQ.

Виконав студент 301-пТК Тараненко Олександр

Передумова

Виконайте лабораторну роботу №1. Створіть нову гілку у раніше створеному гіт репозиторії *NUPP_NET_2025_{Номер Групи}_ТК_{Прізвище}_Lab* із назвою **lab2**, яка міститиме код із першої лабораторної роботи.

Для здачі лабораторної роботи, необхідно буде створити пул реквест із гілки **lab2** у **master**, або якщо на момент здачі пул реквест із гілки lab1 у master не закритий - із гілки lab2 у lab1.

Завдання

1. Реалізувати асинхрону версію дженерік CRUD сервісу, який буде зберігати дані у одній із вбудованих колекцій .NET, буде багатопотоково-безпечною (thread safe), зберігатиме асинхронно колекцію у серіалізованому вигляді у файлі за шляхом `FilePath`, матиме вбудовану підтримку пагінації та реалізовуватиме наступний інтерфейс та інтерфейс `IEnumerable`:

```
public interface ICrudServiceAsync<T> : IEnumerable<T>
{
    public Task<bool> CreateAsync(T element);
    public Task<T> ReadAsync(Guid id);
    public Task<IEnumerable<T>> ReadAllAsync();
    public Task<IEnumerable<T>> ReadAllAsync(int page, int amount);
    public Task<bool> UpdateAsync(T element);
    public Task<bool> RemoveAsync(T element);
    public Task<bool> SaveAsync();
}
```

2. Створіть статичні методи у класах, які створюватимуть нові об'єкти цього класу із згенерованими даними:

```
public class Bus
{
    public static Bus CreateNew()
    {
        throw new NotImplementedException();
    }
}
```

3. Модифікуйте консольний застосунок, щоб він паралельно створив від тисячі об'єктів використовуючи `CRUD service`, створені у об'єктах метод `CreateNew` та

клас Parallel. Для створених об'єктів знайдіть мінімальні, максимальні та середні значення для цифрових значень, отримані результати виведіть у консоль. Згенеровану колекцію збережіть у файл.

4. Додайте приклади використання примітивів синхронізації як Lock, Semaphore, AutoResetEvent та інші.

До PR готової лабораторної роботи додайте PDF файл у якому будуть результати виконання консольного застосунку.

Додаткове завдання

Створіть модульні (Unit) тести для створеною вами реалізації ICrudServiceAsync<T>.

Результати виконання консольного застосунку.

```
Консоль отладки Microsoft Visual Studio

=== Вс? ПК ?гри ===
Game: Cyberpunk 2077, Price: 59,99, Rating: 7,5
Game: The Witcher 3, Price: 39,99, Rating: 9,5

=== П?сля знижки на Cyberpunk ===
Game: The Witcher 3, Price: 39,99, Rating: 9,5
Game: Cyberpunk 2077, Price: 47,99200000000004, Rating: 7,5

=== П?сля видалення The Witcher 3 ===
Game: Cyberpunk 2077, Price: 47,99200000000004, Rating: 7,5

Дан? збережено у файл: pcgames.json

=== Дан? п?сля завантаження з файлу ===
Game: Cyberpunk 2077, Price: 47,99200000000004, Rating: 7,5

=== Демонстрац?я прим?тив?в синхрон?зац?ї ===
=== Демонстрац?я lock ===
Пот?к 10: 0
Пот?к 10: 1
Пот?к 10: 2
Пот?к 10: 3
Пот?к 10: 4
Пот?к 11: 0
Пот?к 11: 1
Пот?к 11: 2
Пот?к 11: 3
Пот?к 11: 4

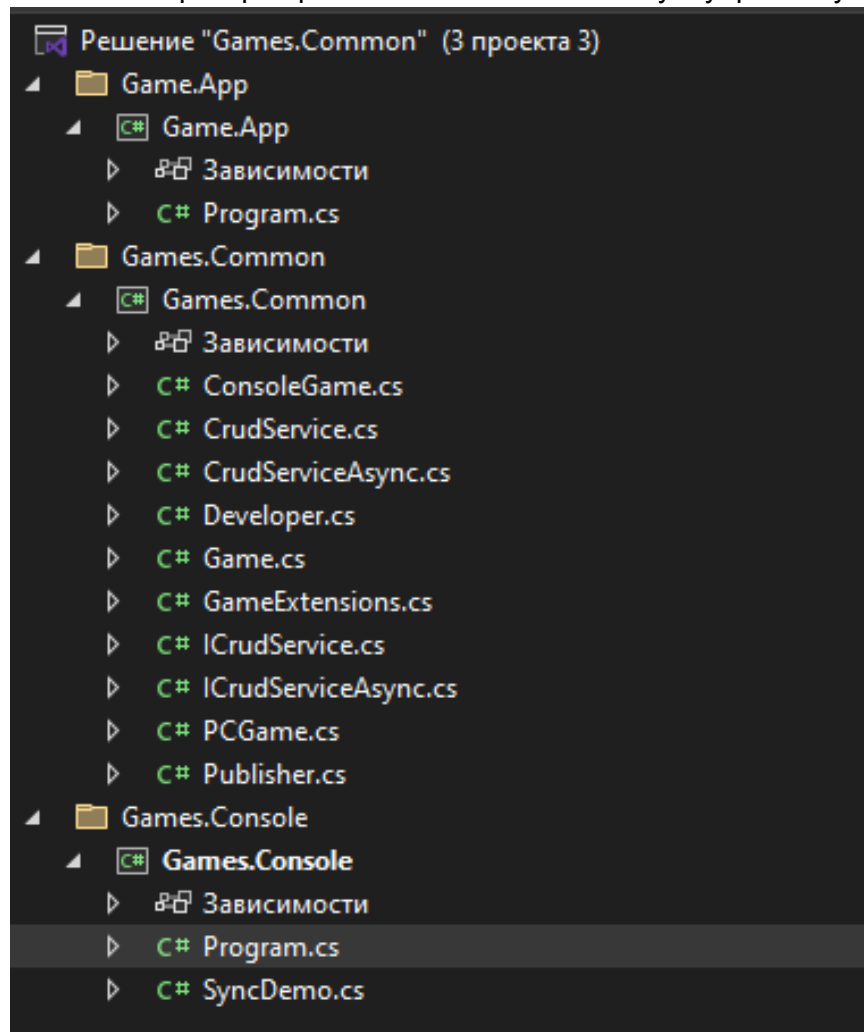
=== Демонстрац?я Semaphore ===
Пот?к 1 чекає доступ...
Пот?к 0 чекає доступ...
Пот?к 0 ув?йшов!
Пот?к 3 чекає доступ...
Пот?к 1 ув?йшов!
Пот?к 4 чекає доступ...
Пот?к 2 чекає доступ...
Пот?к 0 виходить.
Пот?к 1 виходить.
Пот?к 3 ув?йшов!
Пот?к 2 ув?йшов!
Пот?к 3 виходить.
Пот?к 2 виходить.
Пот?к 4 ув?йшов!
Пот?к 4 виходить.

=== Демонстрац?я AutoResetEvent ===
Оч?кування сигналу...
Отримано сигнал!
Надсилаю сигнал...

Натисн?ть будь-яку клавшу для виходу...

C:\Users\Monolit\Desktop\Lab2\Games.Common\Games.Console\bin\Debug\net9.0\Games.Console.exe (процесс 12896) завершил раб
```

Готова лабораторна робота повинна мати наступну файлову ієрархію:



Контрольні запитання

1. Що таке інтерфейс `ICrudServiceAsync<T>`? Яке його призначення? Чому доцільно використовувати generic тип `T` для CRUD-сервісу?

Це **контракт** (набір методів), який визначає асинхронні CRUD-операції (**Create, Read, Update, Delete**) для будь-якого типу `T`.

```
public interface ICrudServiceAsync<T>
{
    Task<T> CreateAsync(T item);
    Task<T?> ReadAsync(Guid id);
    Task<IEnumerable<T>> ReadAllAsync();
    Task<T> UpdateAsync(T item);
    Task DeleteAsync(Guid id);
}
```

Призначення: відокремити логіку роботи з даними від конкретної реалізації (файл, БД, API).

Generic T: дозволяє створювати сервіс для **будь-якого класу** (наприклад, PCGame, User тощо).

2. Що таке процес та потік, яка між ними різниця? Що таке багатопотокове виконання?

Процес – програма, яка виконується (має пам'ять, ресурси, код).

Потік – "шлях виконання" всередині процесу.

Процес може мати багато потоків.

Це здатність програми запускати **кілька потоків одночасно**, щоб використовувати процесор ефективніше (паралельні обчислення, робота з файлами, мережею).

3. Що таке асинхронність, яка різниця між асинхронністю та багатопотоковістю?

Асинхронність – виконання операцій без блокування потоку (наприклад, читання з файлу: потік вільний, поки ОС читає).

Багатопотоковість – одночасна робота кількох потоків.

Асинхронність \neq багатопотоковість. Асинхронність може працювати навіть в одному потоці.


4. Для чого використовуються ключові слова `async/await`?

- `async` позначає метод як асинхронний.
- `await` "чекає" виконання асинхронної задачі, не блокуючи потік.

5. Чим відрізняється `Task` від `ValueTask`?

`Task` завжди створює об'єкт у купі (heap).

`ValueTask` легший, може повертати результат без створення нового об'єкта.

 Використовують `ValueTask`, коли **часто результат доступний одразу**, а `Task` – універсальний.

6. Що таке thread-safe колекція? Які приклади таких колекцій у .NET ви знаєте?

Це колекція, до якої можна одночасно звертатись з кількох потоків без помилок.
Приклади в .NET:

- `ConcurrentDictionary<TKey, TValue>`
- `ConcurrentQueue<T>`
- `BlockingCollection<T>`

7. Для чого використовуються примітиви синхронізації, такі як `lock`, `Semaphore`, `AutoResetEvent`?

`lock` – блокує доступ до ресурсу (тільки один потік).

`Semaphore` – дозволяє кільком потокам обмежений доступ.

`AutoResetEvent` – дає сигнал одному потоку, після чого знову стає "закритим".

8. Як забезпечити безпеку при одночасному зверненні кількох потоків до спільного ресурсу?

- Використати `lock` або `Monitor`.
- Використати thread-safe колекції.
- Використати примітиви (`Semaphore`, `Mutex`).

9. Як за допомогою LINQ можна отримати мінімальне, максимальне та середнє значення певної властивості?

```
var numbers = new[] { 3, 7, 1, 9 };  
var min = numbers.Min();  
var max = numbers.Max();  
var avg = numbers.Average();
```

10. У чому різниця між методами `Select`, `Where`, `Aggregate`, `OrderBy`?

- `Select` – проєкція (перетворення даних).

- Where – фільтрація.
- Aggregate – агрегування з кроком (наприклад, добуток).
- OrderBy – сортування.

11. Які переваги використання LINQ у порівнянні з класичними циклами?

- Код коротший та зрозуміліший.
- Оптимізація виконується під капотом.
- Є багато вбудованих методів.

12. Що станеться, якщо два потоки одночасно спробують зберегти колекцію у файл?

- Дані можуть пошкодитися (race condition).
- Вирішується lock або Semaphore.

13. Як працює Parallel.For і у яких випадках його краще використовувати?

- Розбиває цикл на кілька потоків для паралельного виконання.
- Використовується для **CPU-bound задач** (математика, обчислення).

```
Parallel.For(0, 10, i => Console.WriteLine(i));
```

14. Що таке пагінація і як вона реалізована у вашій системі?

Пагінація – розбиття результатів на сторінки.

Реалізується за допомогою Skip і Take:

```
var page = 2;  
var pageSize = 5;  
var items = collection.Skip((page - 1) * pageSize).Take(pageSize);
```