

Лабораторна робота №2

Багатопоковість. Асинхронність. IEnumerable. LINQ.

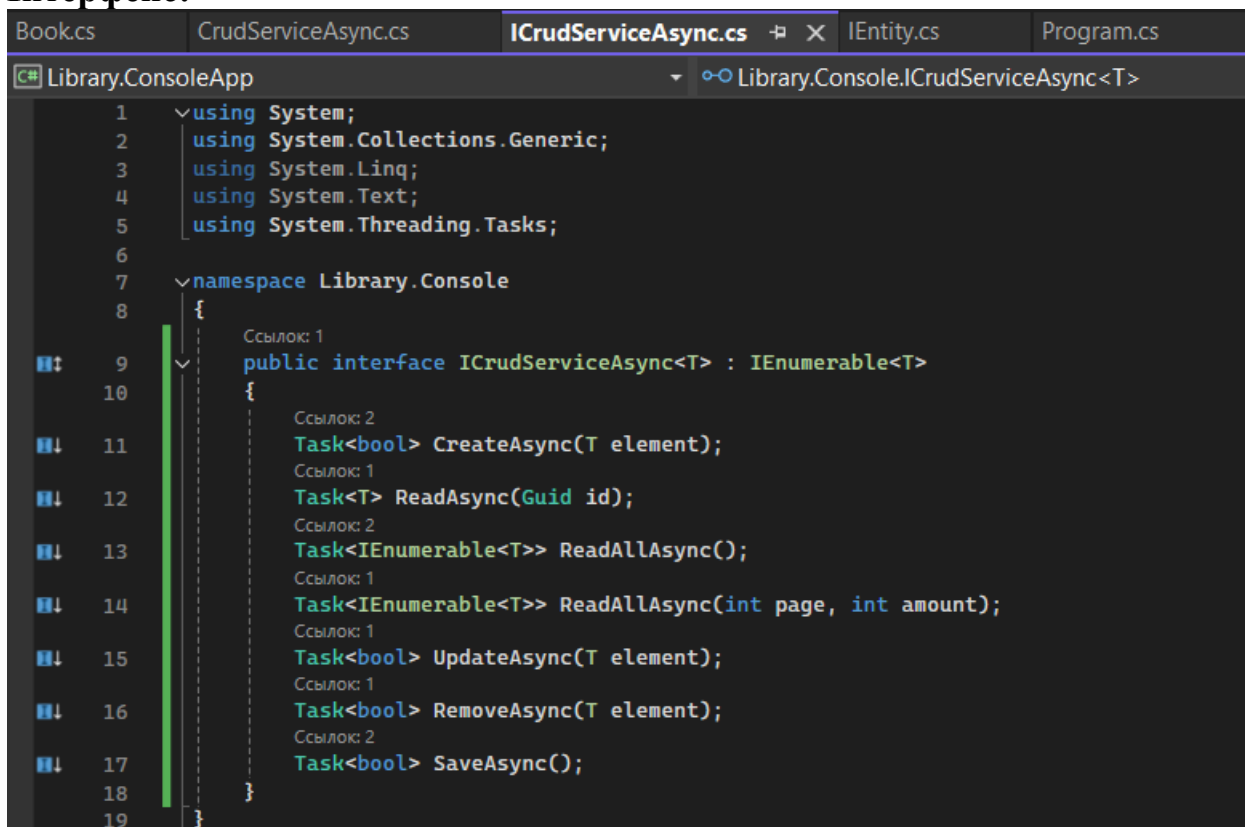
Виконала студентка групи 302-ТК
Варинська Євгенія

Завдання

1. Реалізувати асинхронну версію дженерік CRUD сервісу, який буде зберігати дані у одній із вбудованих колекцій .NET, буде багатопотоково-безпечною (thread safe), зберігатиме асинхроно 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();
}
```

Інтерфейс:



The screenshot shows the Visual Studio IDE with the 'Library.ConsoleApp' project selected. The file 'ICrudServiceAsync.cs' is open, displaying the implementation of the 'ICrudServiceAsync<T>' interface. The code includes using statements for System, System.Collections.Generic, System.Linq, System.Text, and System.Threading.Tasks. The interface is defined within the 'Library.Console' namespace. The methods implemented are CreateAsync, ReadAsync, ReadAllAsync, ReadAllAsync (with pagination), UpdateAsync, RemoveAsync, and SaveAsync. The code is annotated with 'Ссылка: 1' and 'Ссылка: 2' for specific lines.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Console
8  {
9      Ссылка: 1
10     public interface ICrudServiceAsync<T> : IEnumerable<T>
11     {
12         Ссылка: 2
13         Task<bool> CreateAsync(T element);
14         Ссылка: 1
15         Task<T> ReadAsync(Guid id);
16         Ссылка: 2
17         Task<IEnumerable<T>> ReadAllAsync();
18         Ссылка: 1
19         Task<IEnumerable<T>> ReadAllAsync(int page, int amount);
20         Ссылка: 1
21         Task<bool> UpdateAsync(T element);
22         Ссылка: 1
23         Task<bool> RemoveAsync(T element);
24         Ссылка: 2
25         Task<bool> SaveAsync();
26     }
27 }
```

Реалізація:

Book.csCrudServiceAsync.csICrudServiceAsync.csIEntity.csProgram.cs

C#Library.ConsoleAppLibrary.Console.CrudServiceAsync<T>

```
1 using Newtonsoft.Json;
2 using System;
3 using System.Collections;
4 using System.Collections.Concurrent;
5 using System.Collections.Generic;
6 using System.IO;
7 using System.Linq;
8 using System.Text;
9 using System.Threading;
10 using System.Threading.Tasks;
11
12 namespace Library.Console
13 {
14     Ссылка: 2
15     public class CrudServiceAsync<T> : ICrudServiceAsync<T> where T : class, IEntity
16     {
17         private readonly ConcurrentDictionary<Guid, T> _storage = new ConcurrentDictionary<Guid, T>();
18         private readonly SemaphoreSlim _lock = new SemaphoreSlim(1, 1);
19         private readonly string _filePath;
20
21         Ссылка: 1
22         public CrudServiceAsync(string filePath)
23         {
24             _filePath = filePath;
25             LoadFromFile();
26         }
27
28         Ссылка: 2
29         public Task<bool> CreateAsync(T element)
30         {
31             return Task.FromResult(_storage.TryAdd(element.Id, element));
32         }
33
34         Ссылка: 1
35         public Task<T> ReadAsync(Guid id)
36         {
37             T value;
38             _storage.TryGetValue(id, out value);
39             return Task.FromResult(value);
40         }
41
42         Ссылка: 2
43         public Task<IEnumerable<T>> ReadAllAsync()
44         {
45             return Task.FromResult<IEnumerable<T>>(_storage.Values.ToList());
46         }
47
48         Ссылка: 1
49         public Task<IEnumerable<T>> ReadAllAsync(int page, int amount)
50         {
51             var items = _storage.Values
```

```
44     {
45         var items = _storage.Values
46             .Skip((page - 1) * amount)
47             .Take(amount)
48             .ToList();
49
50         return Task.FromResult<IEnumerable<T>>(items);
51     }
52
53     Ссылка: 1
54     public Task<bool> UpdateAsync(T element)
55     {
56         if (!_storage.ContainsKey(element.Id))
57             return Task.FromResult(false);
58
59         _storage[element.Id] = element;
60         return Task.FromResult(true);
61     }
62
63     Ссылка: 1
64     public Task<bool> RemoveAsync(T element)
65     {
66         T removed;
67         return Task.FromResult(_storage.TryRemove(element.Id, out removed));
68     }
69
70     Ссылка: 2
71     public async Task<bool> SaveAsync()
72     {
73         await _lock.WaitAsync();
74         try
75         {
76             var data = _storage.ToList();
77             var json = JsonConvert.SerializeObject(data, Formatting.Indented);
78             File.WriteAllText(_filePath, json, Encoding.UTF8);
79             return true;
80         }
81         catch
82         {
83             return false;
84         }
85         finally
86         {
87             _lock.Release();
88         }
89     }
90
91     Ссылка: 1
92     private void LoadFromFile()
93     {
94         if (!File.Exists(_filePath))
```

```
Book.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs
C# Library.ConsoleApp Library.Console.CrudServiceAsync<T>
88 private void LoadFromFile()
89 {
90     if (!File.Exists(_filePath))
91         return;
92
93     try
94     {
95         var json = File.ReadAllText(_filePath, Encoding.UTF8);
96         var data = JsonConvert.DeserializeObject<List<KeyValuePair<Guid, T>>>(json);
97         if (data != null)
98         {
99             foreach (var pair in data)
100                 _storage[pair.Key] = pair.Value;
101         }
102     }
103     catch
104     {
105         // ignore load errors
106     }
107 }
108
109 Ссылка: 1
110 public IEnumerator<T> GetEnumerator()
111 {
112     return _storage.Values.GetEnumerator();
113 }
114
115 Ссылка: 0
116 IEnumerator IEnumerable.GetEnumerator()
117 {
118     return GetEnumerator();
119 }
```

Модель даних:

```
Book.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs
Library.ConsoleApp  Library.Console.Book

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Console
8  {
9      Ссылка: 3
10     public class Book : IEntity
11     {
12         Ссылка: 6
13         public Guid Id { get; set; }
14         Ссылка: 3
15         public string Title { get; set; }
16         Ссылка: 3
17         public string Author { get; set; }
18
19         Ссылка: 1
20         public Book()
21         {
22             Id = Guid.NewGuid();
23             Title = "";
24             Author = "";
25         }
26
27         Ссылка: 0
28         public override string ToString()
29         {
30             return $"{Title} by {Author}";
31         }
32     }
33 }
```

Консольна програма:

```

1  using Library.Console;
2  using System;
3  using System.Threading.Tasks;
4
5  class Program
6  {
7      static async Task Main(string[] args)
8      {
9          var service = new CrudServiceAsync<Book>("books.json");
10
11          var book = new Book
12          {
13              Title = "CLR via C#",
14              Author = "Jeffrey Richter"
15          };
16
17          await service.CreateAsync(book);
18          await service.SaveAsync();
19
20          var allBooks = await service.ReadAllAsync();
21          Console.WriteLine("Books in storage:");
22          foreach (var b in allBooks)
23          {
24              Console.WriteLine(b);
25          }
26      }
27  }

```

Інтерфейс базової сутності:

```

4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Console
8  {
9      public interface IEntity
10     {
11         Guid Id { get; set; }
12     }
13 }

```

Результат програми:

```

C:\WINDOWS\system32\cmd.exe
Books in storage:
CLR via C# by Jeffrey Richter
CLR via C# by Jeffrey Richter
C# Advanced by John Doe
1984 by George Orwell
C# Advanced by John Doe
Press any key to continue . . .

```

- 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();
}
}

```

```

Book.cs*  X  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs
Library.Console.App  Library.Console.Book  Title
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Console
8  {
9      Ссылка: 5
10     public class Book : IEntity
11     {
12         Ссылка: 6
13         public Guid Id { get; set; }
14         Ссылка: 2
15         public string Title { get; set; }
16         Ссылка: 2
17         public string Author { get; set; }
18
19         // Конструктор
20         Ссылка: 2
21         public Book(string title, string author)
22         {
23             Id = Guid.NewGuid();
24             Title = title;
25             Author = author;
26         }
27
28         // Статичний метод для генерації випадкової книги
29         Ссылка: 0
30         public static Book CreateRandom()
31         {
32             var titles = new[] { "1984", "Brave New World", "CLR via C#", "The Pragmatic Programmer", "Clean Code" };
33             var authors = new[] { "George Orwell", "Aldous Huxley", "Jeffrey Richter", "Andy Hunt", "Robert C. Martin" };
34
35             var random = new Random();
36             string title = titles[random.Next(titles.Length)];
37             string author = authors[random.Next(authors.Length)];
38
39             return new Book(title, author);
40         }
41     }
42 }

```

Використання:

```

var book = Book.CreateRandom();
Console.WriteLine($"{book.Title} by {book.Author}");

```

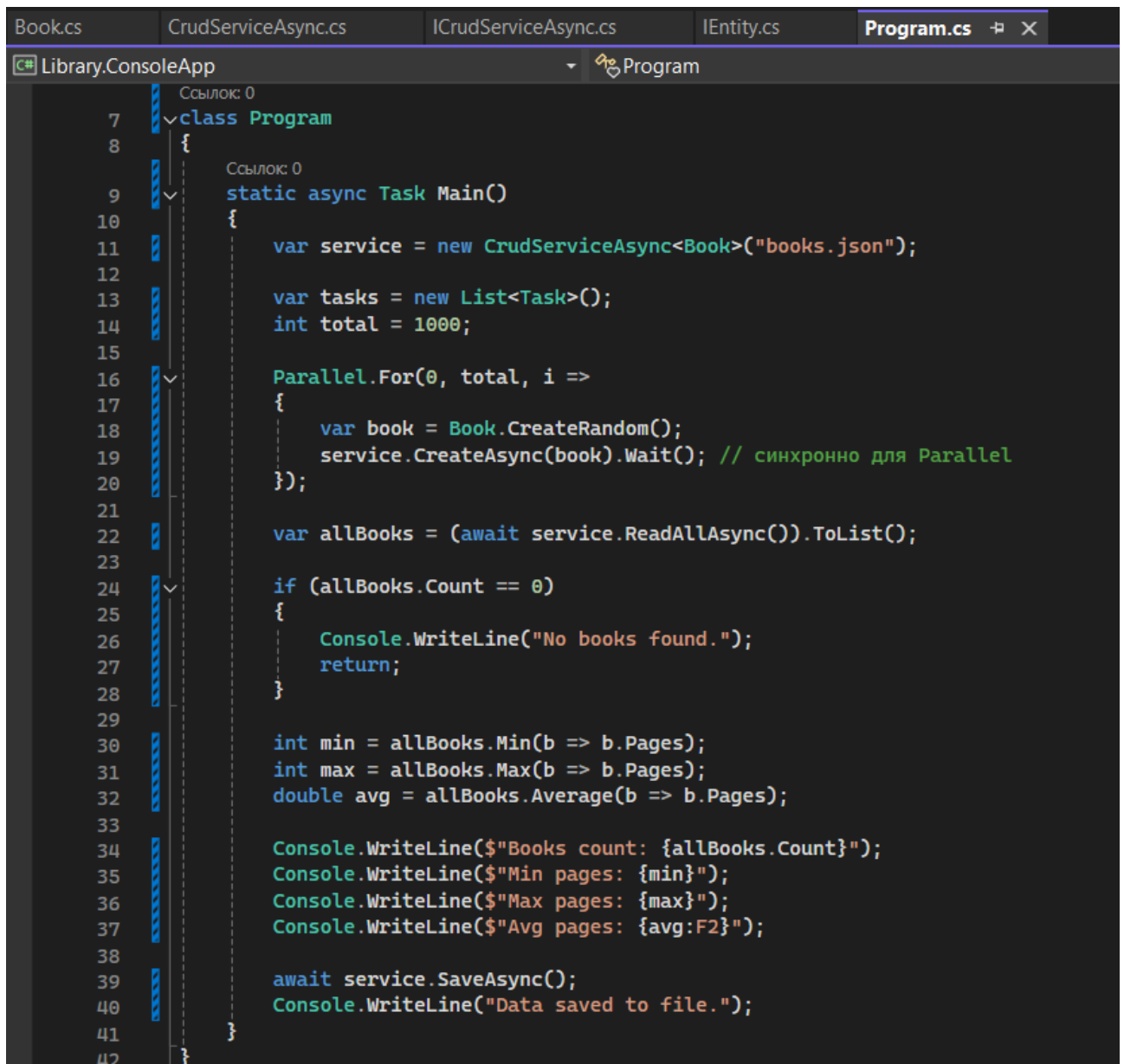
3. Модифікуйте консольний застосунок, щоб він паралельно створив від тисячі об'єктів використовуючи CRUD service, створені у об'єктах метод CreateNew та клас Parallel. Для створених об'єктів знайдіть мінімальні, максимальні та середні значення для цифрових значень, отримані результати виведіть у консоль. Згенеровану колекцію збережіть у файл.

Клас:

```
Book.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs
Library.ConsoleApp  Library.Console.Book  Title

4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Library.Console
8  {
9      Ссылка: 5
10     public class Book : IEntity
11     {
12         Ссылка: 6
13         public Guid Id { get; set; }
14         Ссылка: 1
15         public string Title { get; set; }
16         Ссылка: 1
17         public string Author { get; set; }
18         Ссылка: 4
19         public int Pages { get; set; }
20
21         Ссылка: 1
22         public Book(string title, string author, int pages)
23         {
24             Id = Guid.NewGuid();
25             Title = title;
26             Author = author;
27             Pages = pages;
28         }
29
30         Ссылка: 1
31         public static Book CreateRandom()
32         {
33             var titles = new[] { "1984", "Brave New World", "CLR via C#", "The Pragmatic Programmer", "Clean Code" };
34             var authors = new[] { "George Orwell", "Aldous Huxley", "Jeffrey Richter", "Andy Hunt", "Robert C. Martin" };
35             var rand = new Random(Guid.NewGuid().GetHashCode());
36
37             string title = titles[rand.Next(titles.Length)];
38             string author = authors[rand.Next(authors.Length)];
39             int pages = rand.Next(100, 1000); // Випадкова кількість сторінок
40
41             return new Book(title, author, pages);
42         }
43     }
44 }
```

Консольна програма:



```
Book.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs
Library.ConsoleApp
class Program
{
    static async Task Main()
    {
        var service = new CrudServiceAsync<Book>("books.json");

        var tasks = new List<Task>();
        int total = 1000;

        Parallel.For(0, total, i =>
        {
            var book = Book.CreateRandom();
            service.CreateAsync(book).Wait(); // синхронно для Parallel
        });

        var allBooks = (await service.ReadAllAsync()).ToList();

        if (allBooks.Count == 0)
        {
            Console.WriteLine("No books found.");
            return;
        }

        int min = allBooks.Min(b => b.Pages);
        int max = allBooks.Max(b => b.Pages);
        double avg = allBooks.Average(b => b.Pages);

        Console.WriteLine($"Books count: {allBooks.Count}");
        Console.WriteLine($"Min pages: {min}");
        Console.WriteLine($"Max pages: {max}");
        Console.WriteLine($"Avg pages: {avg:F2}");

        await service.SaveAsync();
        Console.WriteLine("Data saved to file.");
    }
}
```

Результат:



```
C:\WINDOWS\system32\cmd.exe
Books count: 1005
Min pages: 0
Max pages: 999
Avg pages: 561.08
Data saved to file.
Press any key to continue . . .
```

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

Клас:

```
SynchronizationExamples.cs  Book.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs
Library.ConsoleApp  Library.Console.SynchronizationExamples  RunManualResetEventExample()

7
8 namespace Library.Console
9 {
10     Ссылка: 4
11     public class SynchronizationExamples
12     {
13         private static readonly Semaphore _semaphore = new Semaphore(2, 2);
14         private static readonly AutoResetEvent _autoResetEvent = new AutoResetEvent(false);
15         private static readonly ManualResetEvent _manualResetEvent = new ManualResetEvent(false);
16         private static readonly object _lockObject = new object();
17
18         Ссылка: 1
19         public static void RunLockExample()
20         {
21             lock (_lockObject)
22             {
23                 System.Console.WriteLine("Locked section executed by " + Thread.CurrentThread.ManagedThreadId);
24             }
25
26         Ссылка: 1
27         public static async Task RunSemaphoreExampleAsync()
28         {
29             _semaphore.WaitOne();
30             try
31             {
32                 System.Console.WriteLine($"Semaphore acquired by {Thread.CurrentThread.ManagedThreadId}");
33                 await Task.Delay(1000);
34             }
35             finally
36             {
37                 _semaphore.Release();
38             }
39
40         Ссылка: 1
41         public static void RunAutoResetEventExample()
42         {
43             new Thread(() =>
44             {
45                 System.Console.WriteLine("Thread waiting for signal...");
46                 _autoResetEvent.WaitOne();
47                 System.Console.WriteLine("Thread received signal!");
48             }).Start();
49
50             Thread.Sleep(1000);
51             _autoResetEvent.Set();
52         }
53     }
54 }
```

```
SynchronizationExamples.cs Book.cs CrudServiceAsync.cs ICrudServiceAsync.cs IEntity.cs Program.cs
Library.ConsoleApp Library.Console.SynchronizationExamples RunSemaphoreExam

23 }
24
25 Ссылка: 1
26 public static async Task RunSemaphoreExampleAsync()
27 {
28     _semaphore.WaitOne();
29     try
30     {
31         System.Console.WriteLine($"Semaphore acquired by {Thread.CurrentThread.ManagedThreadId}");
32         await Task.Delay(1000);
33     }
34     finally
35     {
36         _semaphore.Release();
37     }
38 }
39
40 Ссылка: 1
41 public static void RunAutoResetEventExample()
42 {
43     new Thread(() =>
44     {
45         System.Console.WriteLine("Thread waiting for signal...");
46         _autoResetEvent.WaitOne();
47         System.Console.WriteLine("Thread received signal!");
48     }).Start();
49
50     Thread.Sleep(1000);
51     _autoResetEvent.Set();
52 }
53
54 Ссылка: 1
55 public static void RunManualResetEventExample()
56 {
57     new Thread(() =>
58     {
59         System.Console.WriteLine("Thread waiting for signal...");
60         _manualResetEvent.WaitOne();
61         System.Console.WriteLine("Thread received signal!");
62     }).Start();
63
64     Thread.Sleep(1000);
65     _manualResetEvent.Set(); // всі потоки, що чекають, розблокуються
66 }
```

Консольна програма:

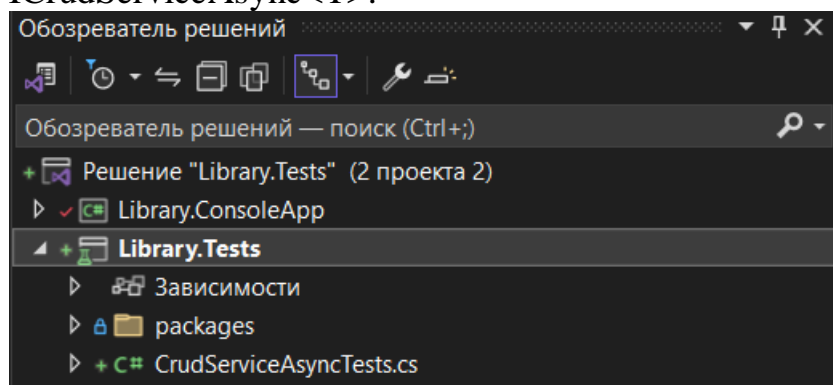
```
SynchronizationExamples.cs  Book.cs  CrudServiceAsync.cs  ICrudServiceAsync.cs  IEntity.cs  Program.cs  Main()
Library.ConsoleApp  Program
Ссылка: 0
7 class Program
8 {
9     Ссылка: 0
10    static async Task Main()
11    {
12        var service = new CrudServiceAsync<Book>("books.json");
13
14        // Створення книг з обмеженням потоків за допомогою Semaphore
15        Console.WriteLine("Creating books with semaphore...");
16        var tasks = new List<Task>();
17        for (int i = 0; i < 10; i++)
18        {
19            tasks.Add(SynchronizationExamples.RunSemaphoreExampleAsync());
20        }
21        await Task.WhenAll(tasks);
22
23        // Додавання книг у сховище
24        Console.WriteLine("Creating books in service...");
25        for (int i = 0; i < 5; i++)
26        {
27            var book = Book.CreateRandom();
28            await service.CreateAsync(book);
29        }
30
31        // Демонстрація lock
32        Console.WriteLine("Running lock example...");
33        SynchronizationExamples.RunLockExample();
34
35        // AutoResetEvent
36        Console.WriteLine("Running AutoResetEvent example...");
37        SynchronizationExamples.RunAutoResetEventExample();
38        await Task.Delay(1500); // Чекаємо, поки потік відпрацює
39
40        // ManualResetEvent
41        Console.WriteLine("Running ManualResetEvent example...");
42        SynchronizationExamples.RunManualResetEventExample();
43        await Task.Delay(1500); // Чекаємо, поки потік відпрацює
44
45        // Статистика книг
46        var allBooks = (await service.ReadAllAsync()).ToList();
47        if (allBooks.Count > 0)
48        {
49            int min = allBooks.Min(b => b.Pages);
50            int max = allBooks.Max(b => b.Pages);
51            double avg = allBooks.Average(b => b.Pages);
52
53            Console.WriteLine($"Books count: {allBooks.Count}");
54            Console.WriteLine($"Min pages: {min}");
55            Console.WriteLine($"Max pages: {max}");
56            Console.WriteLine($"Avg pages: {avg:F2}");
57        }
58        else
59        {
60            Console.WriteLine("No books found.");
61        }
62
63        // Збереження у файл
64        await service.SaveAsync();
65        Console.WriteLine("Data saved to file.");
66    }
67 }
```

Результат:

```
C:\WINDOWS\system32\cmd.exe
Creating books with semaphore...
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Semaphore acquired by 1
Creating books in service...
Running lock example...
Locked section executed by 4
Running AutoResetEvent example...
Thread waiting for signal...
Thread received signal!
Running ManualResetEvent example...
Thread waiting for signal...
Thread received signal!
Books count: 1015
Min pages: 0
Max pages: 999
Avg pages: 561,28
Data saved to file.
Press any key to continue . . .
```

Додаткове Завдання

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



Клас:

```
1  using Library.Console;
2
3  namespace Library.Tests
4  {
5      Ссылка: 0
6      public class CrudServiceAsyncTests
7      {
8          Ссылка: 5
9          private string GetTempFilePath()
10         {
11             return Path.Combine(Path.GetTempPath(), $"test_books_{Guid.NewGuid()}.json");
12         }
13
14         [Fact]
15         Ссылка: 0
16         public async Task CreateAsync_ShouldAddElement()
17         {
18             var service = new CrudServiceAsync<Book>(GetTempFilePath());
19             var book = new Book("Test Book", "Author", 123);
20
21             bool result = await service.CreateAsync(book);
22             var fetched = await service.ReadAsync(book.Id);
23
24             Assert.True(result);
25             Assert.NotNull(fetched);
26             Assert.Equal("Test Book", fetched.Title);
27         }
28
29         [Fact]
30         Ссылка: 0
31         public async Task ReadAllAsync_ShouldReturnAll()
32         {
33             var service = new CrudServiceAsync<Book>(GetTempFilePath());
34             var book1 = new Book("A", "B", 100);
35             var book2 = new Book("C", "D", 200);
36
37             await service.CreateAsync(book1);
38             await service.CreateAsync(book2);
39
40             var all = (await service.ReadAllAsync()).ToList();
41
42             Assert.Equal(2, all.Count);
43         }
44
45         [Fact]
46         Ссылка: 0
47         public async Task UpdateAsync_ShouldModifyElement()
```

```
CrudServiceAsyncTests.cs
Library.Tests
Library.Tests.CrudServiceAsyncTests

42 public async Task UpdateAsync_ShouldModifyElement()
43 {
44     var service = new CrudServiceAsync<Book>(GetTempFilePath());
45     var book = new Book("Original", "Author", 150);
46
47     await service.CreateAsync(book);
48
49     book.Title = "Updated";
50     bool result = await service.UpdateAsync(book);
51     var updated = await service.ReadAsync(book.Id);
52
53     Assert.True(result);
54     Assert.Equal("Updated", updated.Title);
55 }
56
57 [Fact]
58 Ссылка: 0
59 public async Task RemoveAsync_ShouldDeleteElement()
60 {
61     var service = new CrudServiceAsync<Book>(GetTempFilePath());
62     var book = new Book("ToDelete", "Author", 99);
63
64     await service.CreateAsync(book);
65     bool removed = await service.RemoveAsync(book);
66     var result = await service.ReadAsync(book.Id);
67
68     Assert.True(removed);
69     Assert.Null(result);
70 }
71
72 [Fact]
73 Ссылка: 0
74 public async Task SaveAsync_ShouldCreateFileWithData()
75 {
76     string path = GetTempFilePath();
77     var service = new CrudServiceAsync<Book>(path);
78     var book = new Book("SaveTest", "Author", 321);
79
80     await service.CreateAsync(book);
81     bool saved = await service.SaveAsync();
82
83     Assert.True(saved);
84     Assert.True(File.Exists(path));
85
86     string content = File.ReadAllText(path);
87     Assert.Contains("SaveTest", content);
88 }
```

Результат:

```
Консоль отладки Microsoft Visual Studio

C:\Program Files\dotnet\dotnet.exe (процесс 2784) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно: █
```

Репозиторий GitHub:

https://github.com/varinskaevg/NUPP_NET_2025_302_TK_Varinska_Lab/tree/Lab2