Практичне заняття 6
Файловий ввід-вивід та серіалізація даних

№	Тема	К-ть балів
1.	*Захист принаймні одного завдання з роботи*	1
2.	Завдання 1	0,8*
3.	Завдання 2	0,8*
4.	Завдання 3	0,8*
5.	Завдання 4	1,2*
6.	*Здача звіту*	0,4
	Всього	**5**

Завдання до лабораторної роботи

1. *Введення-виведення інформації у/з текстового файлу*

Для роботи з текстовими файлами у просторі імен System.IO визначені спеціальні класи: StreamReader та StreamWriter. Клас StreamReader дозволяє зчитувати весь текст або окремі рядки з текстового файлу.

У даному завданні слід створити додаток, що дозволятиме виконувати зчитування та запис текстової інформації для деякого файлу. Якщо файл із заданим шляхом не існуватиме, слід вивести повідомлення, що такого файлу немає. Переконайтесь, що поле вводу тексту підтримує багаторядковий запис – у XAML для відповідного елементу додайте властивості

```
TextWrapping="Wrap"
AcceptsReturn="True"
```

Приблизний вигляд додатку наведено на рис. 1.

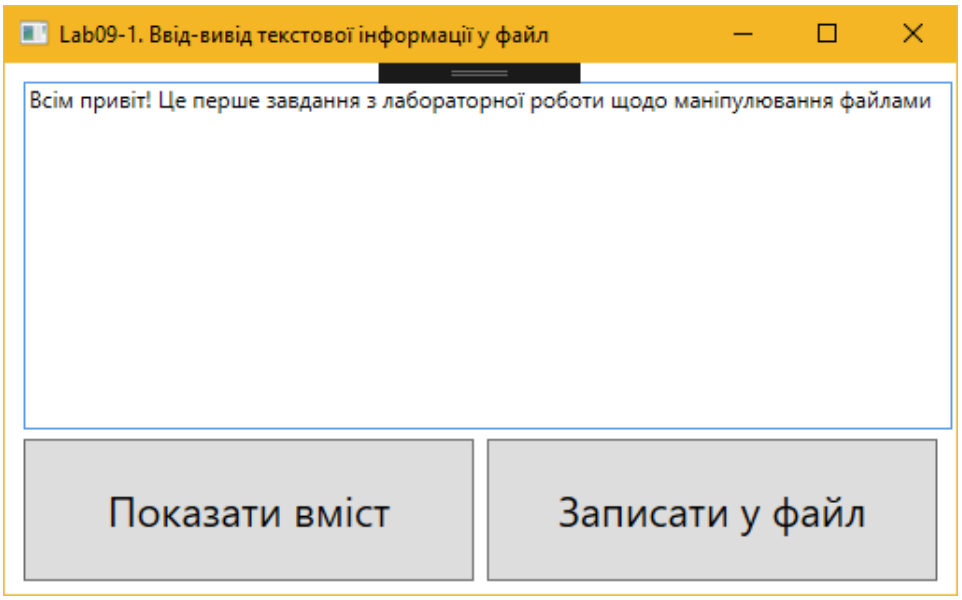


Рис. 1. Приблизний вигляд додатку до задачі 1

2. *Доповніть програму із завдання 1 так, щоб у вікні також відображались додаткові операції з файлом та інформація про нього:*

- Видалення файлу (метод File.Delete()) при натисненні на кнопку
- Кількість рядків у файлі

- Розмір файлу в байтах (з класу `FileInfo`)

Розгляньте атрибути FileAttributes та спробуйте доповнити інформацію, що виводиться. Зразок інтерфейсу для програми наведено на рис. 2

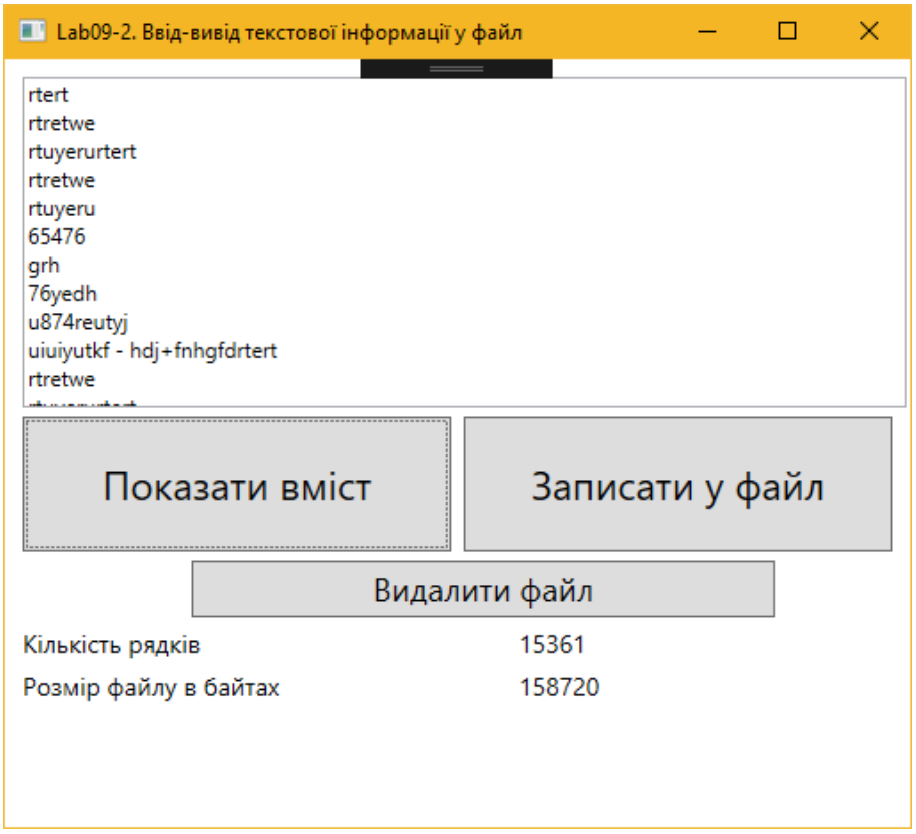


Рис. 2. Зразок інтерфейсу до задачі 2

3. <u>Візуальний вибір файлу</u>

Розширте можливості задачі 2 на вибір файлу за допомогою діалогу. Для цього можна використовувати клас `OpenFileDialog`. Не забудьте поставити фільтр на текстові файли. Діалог вибору виглядатиме аналогічно до рис. 3.

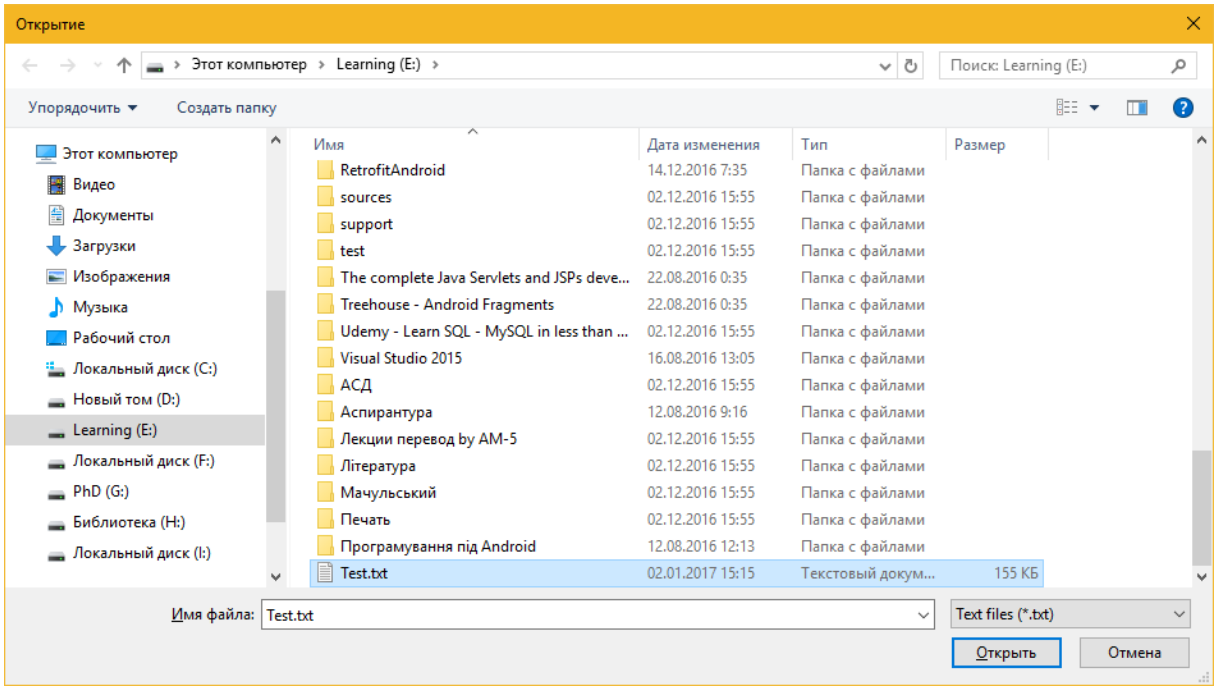


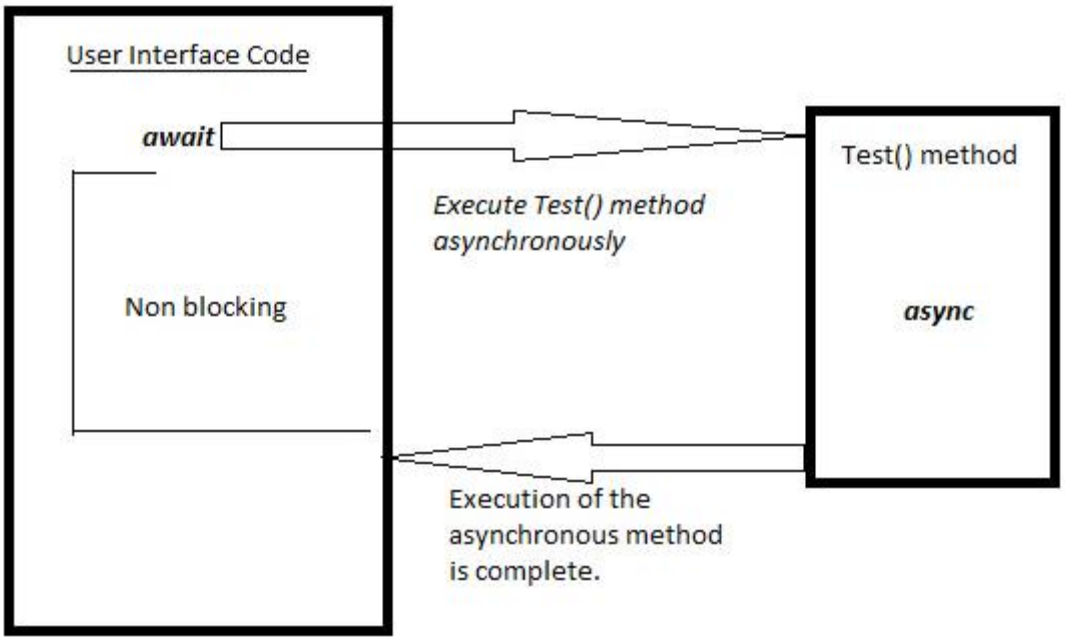
Рис. 3. Вибір файлу в задачі 3

Також слід передбачити виконання зберігання інформації в довільний текстовий файл (SaveFileDialog).

4. *Асинхронні операції з файлами*

Клас FileStream використовується і для асинхронних операцій вводу-виводу. У більшості випадків це дає змогу переконатись, що потік, який викликав операцію, не блокується. Для цього потрібно задіяти опцію асинхронізації (useAsync: true) при створенні екземпляру класу FileStream. Приклад наведено нижче:

```
FileStream fileStream = new FileStream("C:\\IDG.txt", FileMode.Append, FileAccess.Write, FileShare.None, bufferSize: 4096, useAsync: true);
```



При цьому використовуватиметься ключове слово await при виклику методу WriteAsync() та ключове слово async в сигнатурі методу. Якщо ключове слово await не буде задіяно всередині асинхронного методу, весь метод буде виконуватись синхронно.

```
static async Task WriteToFileAsync(string filePath, string text)
    {
      if (string.IsNullOrEmpty(filePath))
         throw new ArgumentNullException("filePath");
      if (string.IsNullOrEmpty(text))
         throw new ArgumentNullException("text");
      byte[] buffer = Encoding.Unicode.GetBytes(text);
      Int32 offset = 0;
      Int32 sizeOfBuffer = 4096;
      FileStream fileStream = null;
      try
      {
```

```
                fileStream = new FileStream(filePath, FileMode.Append,
                FileAccess.Write, FileShare.None, bufferSize: sizeOfBuffer,
                useAsync: true);
                await fileStream.WriteAsync(buffer, offset, buffer.Length);
            }
            catch
            {
               //Write code here to handle exceptions.
            }
            finally
            {
               if(fileStream != null)
               fileStream.Dispose();
            }
        }
```

Зауважте, що використання async та await не створює додаткових потоків. Наступний код демонструє, як можна викликати метод WriteToFileAsync() з методу Main() з метою асинхронного запису тексту.

```
static Task WriteToFile()
        {
           string filePath = "C:\\IDG.txt";
           string text = "Hello World\r\n";
           return WriteToFileAsync(filePath, text);
        }
static void Main(string[] args)
        {
           WriteToFile().Wait();
           Console.Write("Press any key to exit... ");
           Console.ReadKey();
        }
```

Наступний метод зчитує текст з файлу асинхронно. Необхідно передавати назву файлу та розмір буферу в якості параметрів. Зверніть увагу на те, що об’єкт readBuffer встановлюється null у catch-блоці, якщо трапляється виключення. Тому викликаючий метод може ідентифікувати, чи трапилось виключення при спробі зчитування тексту з файлу.

```
static async Task<string> ReadFromFileAsync(string filePath, int bufferSize) {
        if (bufferSize < 1024)
              throw new ArgumentNullException("bufferSize");
```

```
        if (string.IsNullOrEmpty(filePath))
            throw new ArgumentNullException("filePath");
        StringBuilder readBuffer = null;
        byte[] buffer = new byte[bufferSize];
        FileStream fileStream = null;
        try
        {
            readBuffer = new StringBuilder();
            fileStream = new FileStream(filePath, FileMode.Open,
                        FileAccess.Read, FileShare.Read, bufferSize: bufferSize,
                        useAsync: true);
            Int32 bytesRead = 0;
            while ((bytesRead = await fileStream.ReadAsync(buffer, 0,
buffer.Length)) > 0)
            {
                readBuffer.Append(Encoding.Unicode.GetString(buffer, 0,
bytesRead));
            }
        }
        catch
        {
            readBuffer = null;
            //Write code here to handle exceptions;
        }
        finally
        {
            if (fileStream != null)
                fileStream.Dispose();
        }
        return readBuffer.ToString();
    }
```

Використання класу FileStream пояснюється тим, що класи StreamReader та StreamWriter не підтримують асинхронні операції зчитування/запису. Можна викликати метод ReadFromFileAsync() з наступного методу. Зауважте, що метод ReadFromFile() повертає об’єкт Task<string>.

```
    static async Task<string> ReadFromFile()
    {
        string filePath = "C:\\IDG.txt";
        return await ReadFromFileAsync(filePath, 4096);
```

```
}
```

Тепер можна викликати метод ReadFromFile() з методу Main(), щоб отримати та вивести весь текст з файлу в консоль.

```
static void Main(string[] args)
{
   string text = ReadFromFile().Result;
   Console.Write(text);
   Console.ReadKey();
}
```

Зберіть загальний проект, який дозволятиме візуально виконувати асинхронні операції зчитування/запису. Для цього можна взяти великі текстові файли, наприклад, твори письменників у текстовому форматі.