



ФАЙЛОВИЙ ВВІД-ВИВІД У МОВІ ПРОГРАМУВАННЯ C#

Лекція 10
Об'єктно-орієнтоване програмування

Питання лекції

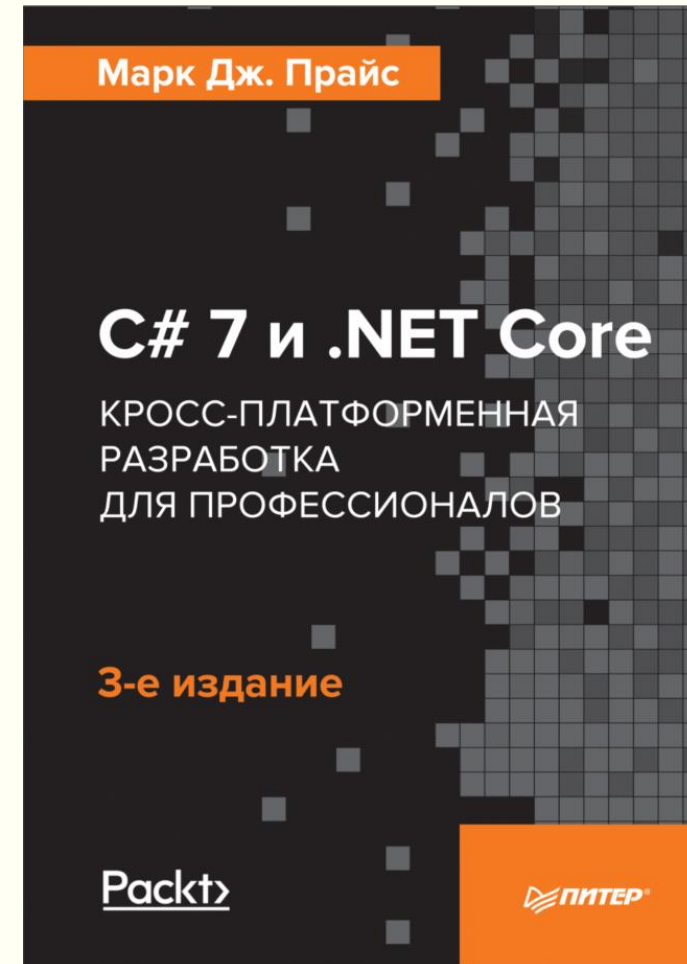
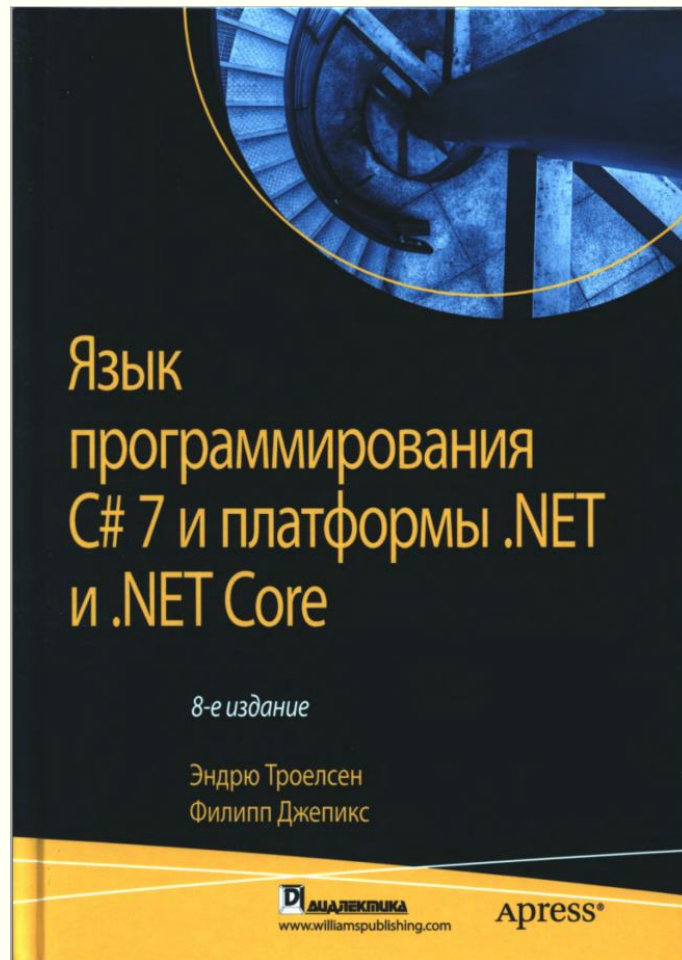
- Файловий ввід-вивід.
- Потоки вводу-виводу.
- Сериалізація об'єктів.



РОБОТА З ФАЙЛАМИ ТА ПОТОКАМИ ДАНИХ

Питання 10.1.

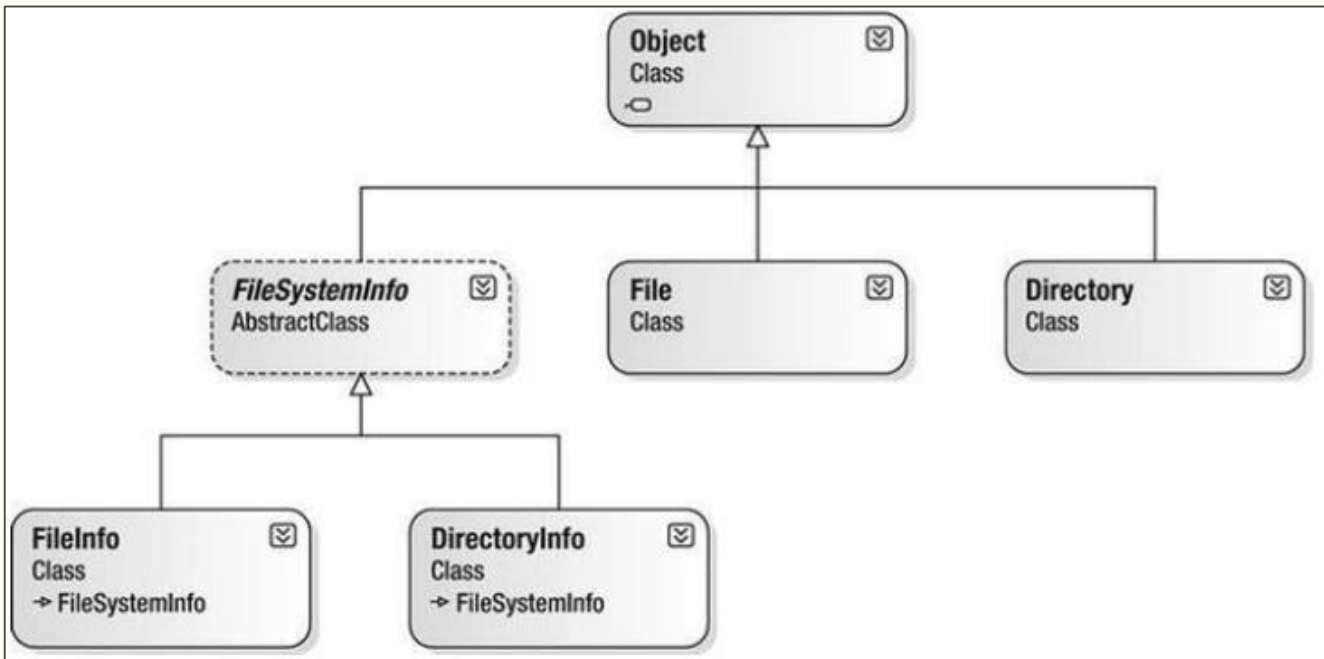
Література



Простір System.io

- Область бібліотек базових класів, присвячена службам файлового вводу-виводу та вводу-виводу з пам'яті.
 - У System.IO визначено набір класів, інтерфейсів, перелічень, структур та делегатів, більшість з яких знаходяться в mscorlib.dll.
 - На додачу, в збірці System.dll визначені додаткові члени System.IO.
 - У всіх проектах Visual Studio автоматично встановлюються посилання на обидві збірки.
- Багато типів з простору імен System.IO концентруються на програмній маніпуляції фізичними каталогами та файлами.
 - Додаткові типи надають підтримку читання/запису даних у рядкові буфери та області пам'яті.
- Члени простору System.io (.NET Framework 4.8)
- Члени простору System.io (.NET 5.0 Preview 7)

Класи Directory (DirectoryInfo) та File (FileInfo)



- Класи **Directory** та **File** пропонують операції створення, видалення, копіювання та переміщення з використанням різних статичних членів.
 - Класи **FileInfo** й **DirectoryInfo** забезпечують схожу функціональність у вигляді методів рівня екземпляра (тому повинні розміщуватись у пам'яті за допомогою ключового слова `new`).
- Зазвичай **FileInfo** й **DirectoryInfo** доречніші для отримання повних деталей щодо файлу чи каталогу (наприклад, часу створення, можливості читання/запису тощо), оскільки їх члени повертають строго типізовані об'єкти.
 - Члени класів **Directory** та **File**, як правило, повертають прості рядкові значення.
 - Проте в багатьох випадках роботу можна виконати, використовуючи **File/FileInfo** або **Directory/DirectoryInfo**.

Використання класу DirectoryInfo

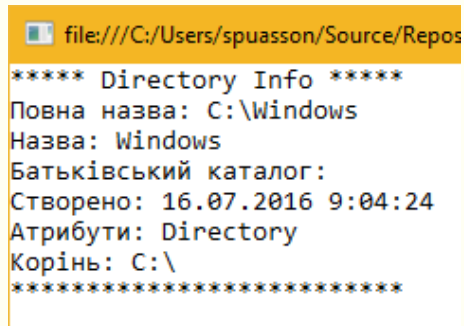
- Класи DirectoryInfo та FileInfo успадковують значну частину своєї поведінки від абстрактного базового класу FileSystemInfo.
 - Тип DirectoryInfo містить набір членів для створення, переміщення, видалення та перелічення каталогів і підкаталогів.
 - На додачу до успадкованої від FileSystemInfo функціональності, клас DirectoryInfo пропонує наступні ключові члени

Член	Описание
Create () CreateSubdirectory ()	Создает каталог (или набор подкаталогов) по заданному путевому имени
Delete ()	Удаляет каталог и все его содержимое
GetDirectories ()	Возвращает массив объектов DirectoryInfo, представляющих все подкаталоги в текущем каталоге
GetFiles ()	Извлекает массив объектов FileInfo, представляющий набор файлов в заданном каталоге
MoveTo ()	Перемещает каталог со всем содержимым по новому пути
Parent	Извлекает родительский каталог данного каталога
Root	Получает корневую часть пути

Використання класу DirectoryInfo. Статичні атрибути

```
class Program {
    static void Main(string[] args) {
        ShowWindowsDirectoryInfo();
        Console.ReadLine();
    }

    static void ShowWindowsDirectoryInfo() {
        // Dump directory information.
        DirectoryInfo dir = new DirectoryInfo(@"C:\Windows");
        Console.WriteLine("***** Directory Info *****");
        Console.WriteLine("Повна назва: {0}", dir.FullName);
        Console.WriteLine("Назва: {0}", dir.Name);
        Console.WriteLine("Батьківський каталог: {0}", dir.Parent);
        Console.WriteLine("Створено: {0}", dir.CreationTime);
        Console.WriteLine("Атрибути: {0}", dir.Attributes);
        Console.WriteLine("Корінь: {0}", dir.Root);
        Console.WriteLine("*****\n");
    }
}
```

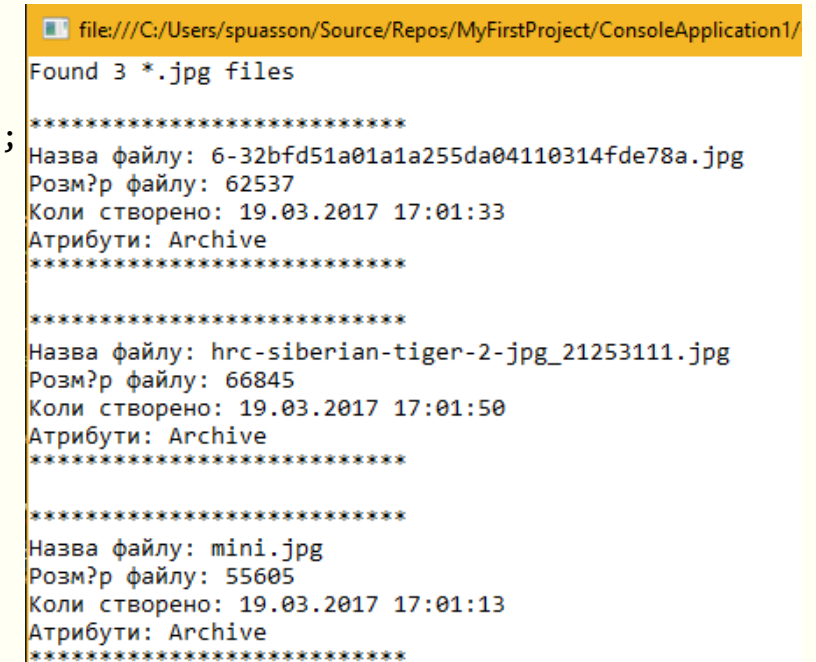


```
file:///C:/Users/spuasson/Source/Repos
***** Directory Info *****
Повна назва: C:\Windows
Назва: Windows
Батьківський каталог:
Створено: 16.07.2016 9:04:24
Атрибути: Directory
Корінь: C:\
*****
```

- Робота з типом DirectoryInfo починається із вказівки деякого шляху в якості параметру конструктора.
 - Шлях до поточного каталогу = ".".
 - DirectoryInfo dir1 = new DirectoryInfo(".");
 - DirectoryInfo dir2 = new DirectoryInfo(@"C:\Windows");
- При спробі взаємодії з неіснуючим каталогом генерується виняток System.IO.DirectoryNotFoundException.
 - Щоб вказати ще не створений каталог, доведеться викликати метод Create():
 - DirectoryInfo dir3 = new DirectoryInfo(@"C:\MyCode\Testing"); dir3.Create();

Використання класу DirectoryInfo. Перелічення файлів

```
DirectoryInfo dir = new DirectoryInfo(@"C:\Wallpaper");
// Отримати всі файли з розширенням *.jpg.
FileInfo[] imageFiles = dir.GetFiles("*.jpg", SearchOption.AllDirectories);
// Скільки було знайдено?
Console.WriteLine("Found {0} *.jpg files\n", imageFiles.Length);
// Тепер виведемо інформацію про кожний файл.
foreach (FileInfo f in imageFiles)
{
    Console.WriteLine("*****");
    Console.WriteLine("Назва файлу: {0}", f.Name);
    Console.WriteLine("Розмір файлу: {0}", f.Length);
    Console.WriteLine("Коли створено: {0}", f.CreationTime);
    Console.WriteLine("Атрибути: {0}", f.Attributes);
    Console.WriteLine("*****\n");
}
```



file:///C:/Users/spuasson/Source/Repos/MyFirstProject/ConsoleApplication1/

Found 3 *.jpg files

Назва файлу: 6-32bfd51a01a1a255da04110314fde78a.jpg
Розм?р файлу: 62537
Коли створено: 19.03.2017 17:01:33
Атрибути: Archive

Назва файлу: hrc-siberian-tiger-2-jpg_21253111.jpg
Розм?р файлу: 66845
Коли створено: 19.03.2017 17:01:50
Атрибути: Archive

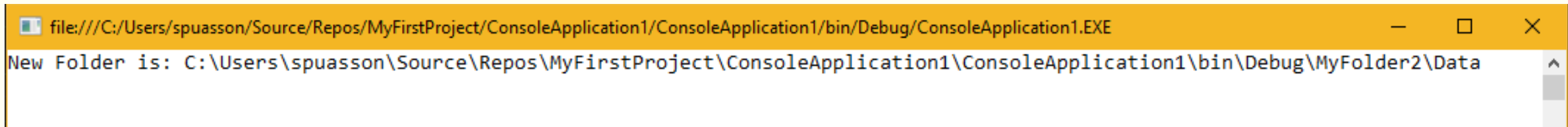
Назва файлу: mini.jpg
Розм?р файлу: 55605
Коли створено: 19.03.2017 17:01:13
Атрибути: Archive

- Метод `GetFiles()` повертає масив об'єктів типу `FileInfo`, кожний з яких відображає детальну інформацію щодо конкретного файлу.
 - Зверніть увагу на опцію пошуку `SearchOption.AllDirectories` у виклику `GetFiles()`, яка забезпечує перегляд усіх підкаталогів кореня.

Використання класу DirectoryInfo. Створення підкаталогів

```
static void ModifyAppDirectory() {  
    DirectoryInfo dir = new DirectoryInfo(".");  
    // Створення \MyFolder  
    dir.CreateSubdirectory("MyFolder");  
    // Перехоплення об'єкту DirectoryInfo, що повертається  
    DirectoryInfo myDataFolder =  
        dir.CreateSubdirectory(@"MyFolder2\Data");  
    // Друкує шлях ..\MyFolder2\Data.  
    Console.WriteLine("New Folder is: {0}", myDataFolder);  
}
```

- Використовується метод DirectoryInfo.CreateSubdirectory().
 - Хоч отримувати значення від методу CreateSubdirectory() не обов'язково, майте на увазі, що при успішному виконанні повертається об'єкт DirectoryInfo, що представляє створений елемент.



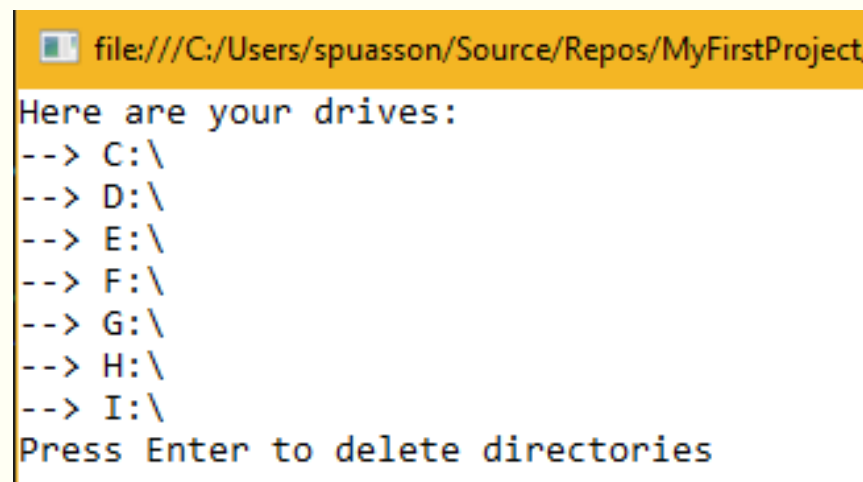
The screenshot shows a Windows command prompt window with a yellow title bar. The title bar text is "file:///C:/Users/spuasson/Source/Repos/MyFirstProject/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1.EXE". The command prompt window has a white background and a black border. The output text is "New Folder is: C:\Users\spuasson\Source\Repos\MyFirstProject\ConsoleApplication1\ConsoleApplication1\bin\Debug\MyFolder2\Data".

```
file:///C:/Users/spuasson/Source/Repos/MyFirstProject/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1.EXE  
New Folder is: C:\Users\spuasson\Source\Repos\MyFirstProject\ConsoleApplication1\ConsoleApplication1\bin\Debug\MyFolder2\Data
```

Робота з типом Directory

```
static void FunWithDirectoryType() {  
    // Вивести список усіх дискових пристроїв комп'ютера.  
    string[] drives = Directory.GetLogicalDrives();  
    Console.WriteLine("Here are your drives:");  
    foreach (string s in drives)  
        Console.WriteLine("--> {0} ", s);  
  
    // Видалити те, що було створено.  
    Console.WriteLine("Press Enter to delete directories");  
    Console.ReadLine();  
    try {  
        Directory.Delete(@"C:\MyFolder");  
        // Другий параметр задає,  
        // чи хочете видаляти підкаталоги.  
        Directory.Delete(@"C:\MyFolder2", true);  
    } catch (IOException e) {  
        Console.WriteLine(e.Message);  
    }  
}
```

- члени Directory зазвичай повертають рядкові дані замість строго типізованих об'єктів FileInfo/DirectoryInfo.



```
file:///C:/Users/spuasson/Source/Repos/MyFirstProject  
Here are your drives:  
--> C:\  
--> D:\  
--> E:\  
--> F:\  
--> G:\  
--> H:\  
--> I:\  
Press Enter to delete directories
```

Робота з типом DriveInfo

```
Console.WriteLine("***** Fun with DriveInfo *****\n");
// Get info regarding all drives.
DriveInfo[] myDrives = DriveInfo.GetDrives();
// Now print drive stats.
foreach (DriveInfo d in myDrives) {
    Console.WriteLine("Name: {0}", d.Name);
    Console.WriteLine("Type: {0}", d.DriveType);
    // Check to see whether the drive is mounted.
    if (d.IsReady) {
        Console.WriteLine("Free space: {0}", d.TotalFreeSpace);
        Console.WriteLine("Format: {0}", d.DriveFormat);
        Console.WriteLine("Label: {0}", d.VolumeLabel);
    }
    Console.WriteLine();
}
```

- Аналогічно до `Directory.GetLogicalDrives()`, статичний метод `DriveInfo.GetDrives()` дозволяє отримати назви пристроїв на машині.
 - На відміну від `Directory.GetLogicalDrives()`, клас `DriveInfo` надає багато додаткових деталей (тип пристрою, доступний вільний простір, мітка тому).

```
file:///C:/Users/spuasson/Source/Repos/MyFirst
***** Fun with DriveInfo *****

Name: C:\
Type: Fixed
Free space: 4381319168
Format: NTFS
Label:

Name: D:\
Type: Fixed
Free space: 4193439744
Format: NTFS
Label: Новый том

Name: E:\
Type: Fixed
Free space: 10515169280
Format: NTFS
Label: Learning

Name: F:\
Type: Fixed
Free space: 965853184
Format: NTFS
Label:

Name: G:\
Type: Fixed
Free space: 3739521024
Format: NTFS
Label: PhD

Name: H:\
Type: Fixed
Free space: 4997697536
Format: NTFS
Label: Библиотека
```

Робота з класом FileInfo

- Клас FileInfo дозволяє отримувати детальні відомості про існуючі файли на диску (час створення, розмір, атрибути та ін.) та допомагає створювати, копіювати, переміщати та видаляти файли.
 - На додачу до набору функціональності, успадкованої від FileSystemInfo, клас FileInfo має ряд унікальних членів:

Член	Опис
AppendText()	Створює об'єкт StreamWriter (розглянемо в наступному питанні) та додає текст у файл
CopyTo()	Копіює існуючий файл у новий файл
Create()	Створює новий файл та повертає об'єкт FileStream (розглянемо в наступному питанні) для взаємодії зі створеним файлом
CreateText()	Створює об'єкт StreamWriter, який записує новий текстовий файл

Метод FileInfo.Create()

- Один із способів створення дескриптора файлу передбачає застосування методу FileInfo.Create().
 - Повернений ним об'єкт FileStream надає синхронну та асинхронну операції запису/зчитування для файлу всім користувачам.
 - Зверніть увагу, що після завершення роботи з поточним об'єктом FileStream необхідно закрити його дескриптор для вивільнення внутрішніх некерованих ресурсів потоку.
 - Враховуючи, що FileStream реалізує інтерфейс IDisposable, можна застосувати контекст using та дозволити компілятору згенерувати логіку завершення

```
// Створює новий файл на диску C.  
FileInfo f = new FileInfo(@"C:\Test.dat");  
FileStream fs = f.Create();  
// Використовує FileStream...  
// Закриває file stream.  
fs.Close();
```

```
FileInfo f = new FileInfo(@"C:\Test.dat");  
using (FileStream fs = f.Create())  
{  
    // Використовуємо об'єкт FileStream...  
}
```

Метод FileInfo.Open()

- Дозволяє відкривати існуючі файли, а також створювати нові файли з набагато вищою точністю, ніж FileInfo.Create() завдяки своїм параметрам, що описують внутрішню структуру цих файлів.
 - Виклик методу повертає об'єкт типу FileStream та в цій версії потребує 3 параметри.

```
// Створення нового файлу за допомогою FileInfo.Open().
FileInfo f2 = new FileInfo(@"C:\Test2.dat");
using (FileStream fs2 = f2.Open(FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.None))
{
    // Використовуємо об'єкт типу FileStream...
}
```

- Перший параметр - загальний тип запиту вводу-виводу (створити новий файл, відкрити існуючий, дописати в файл тощо).

```
public enum FileMode
{
    CreateNew,
    Create,
    Open,
    OpenOrCreate,
    Truncate,
    Append
}
```

```
public enum FileAccess
{
    Read,
    Write,
    ReadWrite
}
```

```
public enum FileShare
{
    Delete,
    Inheritable,
    None,
    Read,
    ReadWrite,
    Write
}
```

Методи FileInfo.OpenRead() та FileInfo.OpenWrite()

- У класі FileInfo також передбачені методи OpenRead() та OpenWrite().
 - Повертають відповідним чином сконфігурований об'єкт типу FileStream без потреби у вказуванні різних значень перелічення.

```
// Отримуємо об'єкт FileStream з дозволами лише на зчитування.  
FileInfo f3 = new FileInfo(@"C:\Test3.dat");  
using (FileStream readOnlyStream = f3.OpenRead()) {  
    // Використовуємо об'єкт типу FileStream...  
}
```

```
// Отримуємо об'єкт FileStream з дозволами лише на запис.  
FileInfo f4 = new FileInfo(@"C:\Test4.dat");  
using (FileStream writeOnlyStream = f4.OpenWrite())  
{  
    // Використовуємо об'єкт типу FileStream...  
}
```


Методи FileInfo.OpenText(), FileInfo.CreateText() і FileInfo.AppendText()

- На відміну від попередніх методів, OpenText() повертає екземпляр типу StreamReader, а не FileStream.
 - Оскільки файл boot.ini вже є на диску, отримуємо доступ до його вмісту так:
- FileInfo.CreateText() і FileInfo.AppendText() повертають об'єкт StreamWriter:

```
// Отримуємо об'єкт типу StreamReader.  
FileInfo f5 = new FileInfo(@"C:\boot.ini");  
using (StreamReader sreader = f5.OpenText())  
{  
    // Використовуємо StreamReader-об'єкт...  
}
```

```
FileInfo f6 = new FileInfo(@"C:\Test6.txt");  
using (StreamWriter swriter = f6.CreateText())  
{  
    // Використовуємо StreamWriter-об'єкт...  
}  
FileInfo f7 = new FileInfo(@"C:\FinalTest.txt");  
using (StreamWriter swriterAppend = f7.AppendText())  
{  
    // Використовуємо StreamWriter-об'єкт...  
}
```

Робота з типом File

```
// Отримати об'єкт FileStream через File.Create().  
using(FileStream fs = File.Create(@"C:\Test.dat")) {...}
```

```
// Отримати об'єкт FileStream через File.Open().  
using(FileStream fs2 = File.Open(@"C:\Test2.dat", FileMode.OpenOrCreate,  
FileAccess.ReadWrite, FileShare.None)) {...}
```

```
// Отримати об'єкт FileStream з правами тільки для зчитування.  
using(FileStream readOnlyStream = File.OpenRead(@"Test3.dat11)) {...}
```

```
// Отримати об'єкт FileStream з правами тільки для запису.  
using(FileStream writeOnlyStream = File.OpenWrite(@"Test4.dat")) {...}
```

```
// Отримати об'єкт StreamReader.  
using(StreamReader sreader = File.OpenText(@"C:\boot.ini11)) {...}
```

```
// Отримати кілька об'єктів StreamWriter.  
using(StreamWriter swriter = File.CreateText(@"C:\Test6.txt")) {...}  
using(StreamWriter swriterAppend = File.AppendText(@"C:\FinalTest.txt")) {...}
```

- Тип File надає функціональність, майже ідентичну до FileInfo, за допомогою кількох статичних методів.

- Подібно до FileInfo, тип File підтримує методи AppendText(), Create(), CreateText(), Open(), OpenRead(), OpenWrite() та OpenText().
- У багатьох випадках типи File і FileInfo взаємозамінні.

Додаткові члени File.

- Консольна програма зберігає рядкові дані в новому файлі на диску C: (і читає їх в пам'ять) з мінімальними зусиллями.
 - Коли необхідно швидко отримати файловий дескриптор, тип File дозволить зекономити на об'ємі коду.
 - Проте перевага попереднього створення об'єкта FileInfo пов'язана з можливістю дослідження файлу за допомогою членів абстрактного базового класу FileSystemInfo.

Метод	Описание
ReadAllBytes ()	Открывает указанный файл, возвращает двоичные данные в виде массива байт и затем закрывает файл
ReadAllLines ()	Открывает указанный файл, возвращает символьные данные в виде массива строк, затем закрывает файл
ReadAllText ()	Открывает указанный файл, возвращает символьные данные в виде System.String (), затем закрывает файл
WriteAllBytes ()	Открывает указанный файл, записывает в него массив байтов и закрывает файл
WriteAllLines ()	Открывает указанный файл, записывает в него массив строк и закрывает файл
WriteAllText ()	Открывает указанный файл, записывает в него данные из указанной строки и закрывает файл

```
Console.WriteLine("***** Simple I/O with the File Type *****\n");
string[] myTasks = {"Fix bathroom sink", "Call Dave",
                    "Call Mom and Dad", "Play Xbox One"};
// Записати всі дані в файл на диску C: .
File.WriteAllLines(@"C:\tasks.txt", myTasks);
// Прочитати всі дані та вивести на консоль.
foreach (string task in File.ReadAllLines(@"C:\tasks.txt"))
{
    Console.WriteLine("TODO: {0}", task);
}
Console.ReadLine();
```



ДЯКУЮ ЗА УВАГУ!

Наступне питання: потоки вводу-виводу