ТЕМА 8-9: ФАЙЛОВЕ ВВЕДЕННЯ-ВИВЕДЕННЯ.

План

8.1. Робота з типом DirectoryInfo.

Цей клас містить набір членів, які використовуються для створення, переміщення, видалення та перерахування каталогів і підкаталогів.

Робота з типом `DirectoryInfo` починається з зазначення певного шляху в якості параметра конструктора. Якщо потрібно отримати доступ до поточного робочого каталогу (тобто каталогу, де виконується додаток), застосовується позначення у вигляді крапки (`.`). Наприклад:

```
// Прив’язатися до поточного робочого каталогу.
DirectoryInfo dir1 = new DirectoryInfo(".");
// Прив’язатися до C:\Tmp, використовуючи дослівний рядок.
DirectoryInfo dir2 = new DirectoryInfo(@"C:\Tmp");
// Створити каталог
DirectoryInfo dir3 = new DirectoryInfo(@"C:\MyCode\Testing");
dir3.Create ();
```

8.2. Перерахування файлів за допомогою типу DirectoryInfo.

Метод GetFiles() повертає масив об'єктів типу FileInfo, кожен з яких відображає детальну інформацію про конкретний файл.

```
DirectoryInfo dir = new DirectoryInfo(@"C:\Tmp");
// Отримати всі файли з розширенням *.jpg.
FileInfo[] imageFiles =
              dir.GetFiles("*.jpg",SearchOption.AllDirectories);
// Скільки файлів знайдено?
Console.WriteLine("Found {0} *.jpg files\n", imageFiles.Length);
// Вивести інформацію про кожен фай.
foreach (FileInfo f in imageFiles)
```

```
{
Console.WriteLine("File name: {0}", f.Name);         //ім’я файлу
Console.WriteLine("File size: {0}", f.Length);       //розмір
Console.WriteLine("Creation: {0}", f.CreationTime); //час створення
Console.WriteLine("Attributes: {0}", f.Attributes); //атрибути
}
```

8.3. Створення підкаталогів за допомогою типу DirectoryInfo.

Для цієї мети служить метод `DirectoryInfo.CreateSubdirectory()`.

```
DirectoryInfo dir = new DirectoryInfo(@"C:\");
// Створити \MyFolder в C:\.
dir.CreateSubdirectory("MyFolder");
// Створити \MyFolder2\Data в C:\.
dir.CreateSubdirectory(@"MyFolder2\Data");
```

У разі його успішного виконання повертається об'єкт DirectoryInfo, який являє собою новостворений елемент.

8.4. Работа з типом Directory.

```
// Повернути список всіх дискових пристроїв.
string[] drives = Directory.GetLogicalDrives();
// Видалити те, що було раніше створено.
try
{
     Directory.Delete(@"C:\MyFolder");
     // Другий параметр вказує, чи потрібно видаляти підкаталоги.
     Directory.Delete(@"C:\MyFolder2", true);
}
catch (IOException e)
{
     Console.WriteLine(e.Message);
}
```

8.5. Работа с классом FileInfo.

Клас `FileInfo` дозволяє отримувати докладні відомості про існуючі файли на жорсткому диску (наприклад, час створення, розмір і атрибути) і допомагає створювати, копіювати, переміщати і видаляти файли.

Один із способів створення дескриптора файлу передбачає застосування методу `FileInfo.Create()`:

```
// Створити новий файл на диску C: .
FileInfo f = new FileInfo(@"C:\Test.dat");
FileStream fs = f.Create();
// Використати об’єкт FileStream...
// Закрити файловий потік.
fs.Close();
```

8.6. Робота з типом File.

Тип File надає функціональність, майже ідентичну типу FileInfo, з допомогою декількох статичних методів.

```
using(FileStream fs = File.Create(@"C:\Test.dat"))
{}
using(FileStream fs2 = File.Open(@"C:\Test2.dat",
FileMode.OpenOrCreate,
FileAccess.ReadWrite, FileShare.None))
{}
using(FileStream readOnlyStream = File.OpenRead(@"Test3.dat11))
{}
using(FileStream writeOnlyStream = File.OpenWrite(@"Test4.dat"))
{}
using(StreamReader sreader = File.OpenText(@"C:\boot.ini11))
{}
using(StreamWriter swriter = File.CreateText(@"C:\Test6.txt"))
{}
using(StreamWriter swriterAppend =
File.AppendText(@"C:\FinalTest.txt"))
{}
```

Використовуючи ці методи типу File, можна здійснювати читання і запис пакетів даних за допомогою всього декількох рядків коду. Ці методи автоматично закривають файловий дескриптор, що лежить в основі. Наприклад, наступна консольна програма зберігає строкові дані в новому файлі на диску C: (і читає їх в пам'ять) з мінімальними зусиллями:

```
string[] myTasks = {
"Fix bathroom sink", "Call Dave",
"Call Mom and Dad", "Play Xbox 360"};

File.WriteAllLines(@"C:\tasks.txt", myTasks);

foreach (string task in File.ReadAllLines(@"C:\tasks.txt"))
{
    Console.WriteLine("TODO: {0}", task);
}
```

8.7. Поняття потоку.

У світі маніпуляцій введенням-виведенням потік (stream) являє собою порцію даних, що протікає від джерела до цілі. Потоки надають загальний спосіб взаємодії з послідовністю байтів, незалежно від того, якого роду пристрій (файл, мережа, з'єднання, принтер і т.п.) зберігає або відображає ці байти.

8.8. Робота з класами StreamWriter і StreamReader.

Класи StreamWriter і StreamReader зручні у всіх випадках, коли потрібно читати або записувати символьні дані (наприклад, рядки).

8.9. Записування у текстовий файл.

```
using(StreamWriter writer = File.CreateText("reminders.txt"))
{
     writer.WriteLine("Don't forget Mother's Day this year...");
     writer.WriteLine("Don't forget Father's Day this year...");
     writer.WriteLine("Don't forget these numbers:");
     for(int i = 0; i < 10; i++)
          writer.Write(i + " ");

     writer.Write(writer.NewLine) ;
}
```

8.10. Читання з текстового файлу.

```
using(StreamReader sr = File.OpenText("reminders.txt"))
{
     string input = null;
     while ((input = sr.ReadLine()) != null)
     {
          Console.WriteLine (input);
     }
}
```

8.11. Робота з класами BinaryWriter і BinaryReader.

Ці типи дозволяють читати і записувати дискретні типи даних у потоки у компактному двійковому форматі. У класі BinaryWriter визначено багаторазово перевантажений метод Write () для приміщення типів даних у потік, що лежить в основі.

У наступному прикладі об'єкти даних різних типів записуються у файл *. dat:

```
FileInfo f = new FileInfo("BinFile.dat");
using(BinaryWriter bw = new BinaryWriter(f.OpenWrite()))
{

Console.WriteLine("Base stream is: {0}", bw.BaseStream);

double aDouble = 1234.67;
int anInt = 34567;
string aString = "A, B, C";
```

```
bw.Write(aDouble);
bw.Write(anInt) ;
bw.Write(aString) ;
}
```

Завдання

1. Створити програму, яка виводить на екран усю ієрархічну структуру певної директорії (файли та папки).
2. Створити програму, що виконує пошук заданого файлу та виводить повну інформацію про місце розташування знайдених екземплярів.
3. Високий рівень. Створити програму з графічним інтерфейсом, яка б відтворювала основні можливості файлового менеджера.