Politechnika Wrocławska
Wydział Mechaniczny
Katedra Konstrukcji i Badań Maszyn i Pojazdów

Programowanie obiektowe Lab 14

Dr inż. Paweł Maślak

pawel.maslak@pwr.edu.pl

Laboratorium 14: Zapis do pliku

Plik jest zbiorem danych przechowywanych na dysku z określoną nazwą i ścieżką katalogu. Gdy plik jest otwarty do odczyty lub zapisu staje się strumieniem.

Strumień to w zasadzie sekwencja bajtów. Wyróżniamy dwa główne strumienie: strumień wejściowy oraz strumień wyjściowy. Strumień wejściowy służy do odczytu danych z pliku, strumień wyjściowy jest używany do zapisywania do pliku.

Klasa I/O

Przestrzeń nazw **System.IO** posiada wiele klas, które są wykorzystywane do operacji na plikach takich jak tworzenie czy usuwanie plików, odczyt i zapis danych czy zamykanie pliku po skończonej operacji.

Poniższa tabela zawiera kilka powszechnie używanych nie abstakcyjnych klas w przestrzeni nazw **System.IO**:

Klasa I/O	Opis
BinaryReader	odczyt danych ze strumienia binarnego
BinaryWriter	zapis danych w formacie binarnym

BufferedStream	pamieć tymczasowa dla strumienia bajtów
Directory	pozwala na zmianę struktury katalogów
DirectoryInfo	służy do wykonywania operacji na katalogach
DriveInfo	zawiera informację o dyskach
File	pomaga w operacjach na plikach
FileInfo	służy do wykonywania operacji na plikach
FileStream	służy do odczytu i zapisu do dowolnego miejsca w pliku
MemoryStream	służy do losowego dostępu do danych strumieniowych przechowywanych w pamięci
Path	wykonuje operacje na ścieżce dostępu
StreamReader	używamy do odczytywania znaków ze strumienia bajtów
StreamWriter	używamy do zapisywania znaków do strumienia bajtów
StringReader	służy do odczytu znaków z bufora ciągu znaków
StringWriter	służy do zapisu znaków do bufora ciągu znaków

Klasa FileStream

Klasa **FileSystem**, która jest zdefiniowana w przestrzeni nazw **System.IO** pomaga w odczycie oraz zapisie danych. Klasa ta dziedziczy z klasy abstrakcyjnej **Stream**.

Musisz utworzyć obiekt **FileStream** aby utworzyć nowy plik lub otworzyć istniejący. Składa pozwalająca na utworzenie obiektu **FileStream** jest następująca:

```
FileStream nazwa_obiektu = new FileStream(nazwa_pliku, typ_wyliczeniowy FileMode, typ_wylic
zeniowy FileAccess, typ_wyliczeniowy FileShare);
```

Dla przykładu utwórzmy obiekt o nazwie **fs** pozwalający na odczytanie danych z pliku **przyklad.txt**:

```
FileStream fs = new FileStream("przyklad.txt", FileMode.Open, FileAccess.Read, FileShare.Re
ad);
```

Dokonajmy analizy każdego z powyższych typów wyliczeniowych:

FileMode zawiera następujące metody:

- **Append** - otwiera istniejący plik i ustawia kursor na końcu dokumentu lub tworzy nowy plik jeżeli ten nie istnieje;
- **Create** - tworzy nowy plik;
- **CreateNew** - informuje system operacyjnym o tym, że powinien utworzyć nowy plik;
- **Open** - otwiera istniejący plik;
- **OpenOrCreate** - informuje system operacyjny o tym, że powinien otworzyć plik jeżeli już istnieje, w przeciwnym wypadku powinien stworzyć nowy plik;
- **Truncate** - otwiera istniejący plik oraz go czyści (plik zajmuje 0 bajtów).

FileAccess zawiera następujące metody: **Read**, **ReadWrite** oraz **Write**.

FileShare zawiera następujące metody:

- **Inheritable** - pozwala przekazać dziedziczenie do procesów potomnych;
- **None** - nie pozwala na dzielenie się obecnym plikiem;
- **Read** - pozwala na otwarcie pliku w celu odczytania jego zawartości;
- **ReadWrite** - pozwala na otwarcie pliku w celu jego odczytania oraz zapisania;
- **Write** - pozwala na otwarcie pliku w celu jego zapisania.

Przykład użycia klasy FileStream:

```
using System;
using System.IO;
namespace IOFIleStream
{
    class Program
    {
        static void Main(string[] args)
        {
            FileStream fs = new FileStream("text.txt", FileMode.OpenOrCreate, FileAccess.Re
adWrite);
            for (int i = 0; i < 20; i++)
            {
                fs.WriteByte((byte)i);
            }
            // Cofamy się na początek naszego pliku
            fs.Position = 0;
            for (int i = 0; i < 20; i++)
            {
                Console.Write(fs.ReadByte() + " ");
            }
            // Zamykamy nasz plik
            fs.Close();
            Console.ReadKey();
            // Wynik działania programu
            //0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
        }
    }
}
```

Poprzedni przykład pokazuje jedynie proste operacje dokonywane na plikach. Jednakże, żeby w pełni wykorzystać klasę **System.IO** należy znacznie bardziej zagłębić się w ten temat.