C# - I/O, streamy, práce se soubory

Centrum pro virtuální a moderní metody a formy vzdělávání na Obchodní akademii T.G. Masaryka, Kostelec nad Orlicí

I/O operace a datové proudy

- V .NET jsou všechny I/O operace realizovány prostřednictvím datových proudů
- Datový proud je reprezentován třídou Stream a jednotlivé typy datové komunikace jsou reprezentovány jejími členy
- Výhoda použití třídy Stream spočívá v jednotném přístupu k objektům, na které chceme výstup (popř. vstup) aplikovat (soubor, zařízení, paměť, …)

Základní metody datových proudů

- Základními metodami použitými pro práci s datovými toky jsou metody Read, Write
- Některé datové proudy mají podporu náhodného přístupu – můžeme se v nich pohybovat, ne je jen číst nebo zapisovat od začátku do konce - podpora je zajištěna parametrem CanSeek. Pokud je náhodný přístup podporován můžeme se v datovém proudu nastavit na určitou pozici pomocí vlastnosti Position

Třída FileStream

- Slouží k přístupu k souborům

```
/// Ukazka zapisu do souboru pomoci metody Write tridy Stream
public static void WriteToFile()
{
  FileStream myFileStream = null; //deklarace datového toku
  try
  {
    //vytvoreni streamu
    myFileStream = new FileStream(@"C:\pokus.txt",FileMode.Create);
    //vytvoreni bufferu bytu, ze ktereho bude zapsano
    byte[] lBuffer = new byte[]{1,2,3,4};
    myFileStream.Write(lBuffer,0,lBuffer.Length); //pokyn k zapisu
    myFileStream.Flush(); //vyprazdneni bufferu a provedeni vsech neprovedenych
  }                                  //zapisu do zarizeni (souboru)
  finally
  {
    myFileStream.Close(); //uzavreni streamu
  }
}
```

```
/// Ukazka nacteni dat ze souboru pomoci metody Read tridy Stream
public static void ReadFromFile()
{
  FileStream myFileStream = null; //deklarace datového toku
  try
  {
    //vytvoreni streamu
    myFileStream = new FileStream(@"C:\pokus.txt",FileMode.Open);
    //vytvoreni bufferu bytu, do ktereho nactu data ze souboru
    byte[] lBuffer = new byte[4];
    myFileStream.Read(lBuffer,0,lBuffer.Length); //pokyn ke cteni
    for(int i = 0; i < lBuffer.Length; i++)  //vypsání obsahu bufferu
       Console.WriteLine(lBuffer[i]);
  }
  finally
  {
    myFileStream.Close(); //uzavreni streamu
  }
}
```

Binary Writer and Reader

- Protože zápis a čtení z datových proudů pouze přes byty by byl poněkud nepohodlný lze využít binárního zapisování a čtení datových proudů, které umí pracovat se všemi základními datovými typy.

```
BinaryWriter myBinWriter = new BinaryWriter(myFileStream);
//zapsani retezce
myBinWriter.Write("Retezec");
//zapsani cisla int
myBinWriterWriter.Write(69);
myBinWriter.Flush();
myBinWriter.Close();
BinaryReader myBinReader = new BinaryReader(lStream);
//vypsani retezce a integeru
Console.WriteLine(myBinReader.ReadString());
Console.WriteLine(myBinReader.ReadInt32());
```

TextReader a TextWriter

- Třídy jmenného prostoru Systém.IO
- Každá z těchto tříd má dva potomky
- **TextReader:**
  - StringReader: čte řetězec
  - StreamReader: čte datový tok bytů
- **TextWriter:**
  - StringWriter: zapisuje řetězec
  - StreamWriter: zapisuje datový tok bytů

Užitečné metody

- **Třída TextReader:**
  - ***ReadLine:*** načte jeden řádek a posune se na další
  - ***ReadToEnd:*** načte všechny znaky a vrátí je jako souvislý řetězec
- **Třída TextWriter:**
  - ***WriteLine:*** zapíše jeden řádek ze zdroje

```
// Ukazka pouziti tridy StreamWriter pro zapis textu do souboru
public static void WriteUsingStreamWriteru()
{
  Stream myTxtStream = null;
  try
  {
    myTxtStream = new FileStream(@"C:\pokus.txt",FileMode.Create);
    TextWriter myTxtWriter = new StreamWriter(myTxtStream);
    myTxtWriter.WriteLine("Prvni radek");
    myTxtWriter.WriteLine("Druhy radek");
    myTxtWriter.Close();
  }
  finally
  {
    myTxtStream.Close();
  }
}
```

```
/// Ukazka pouziti tridy StreamReader pro cteni textu ze souboru
public static void ReadUsingStreamReaderu()
{
  Stream myTxtStream = null;
  try
  {
    myTxtStream = new FileStream(@"C:\pokus.txt",FileMode.Open);
    TextReader myTxtReader = new StreamReader(myTxtStream);
    string myStrRadek;
    while((myStrRadek = myTxtReader.ReadLine()) != null)
      Console.WriteLine(myStrRadek);
  }
  finally
  {
    myTxtStream.Close();
  }
}
```

Memory Stream

- Analogicky jako při práci se zápisem a čtením do/ze souboru lze použít třídu MemoryStream pro práci s pamětí počítače
- Slouží většinou pro dočasné uchování dat, které nemusí být uloženy v žádném trvalém úložišti

Třída File

- Určená pro pohodlnou práci se soubory
- Obsahuje pouze statické metody
- ***Umožňuje:*** vytváření, mazání, kontrolování existence, kopírování, přesouvání, změnu atributů, editaci času vytvoření, otevírání, …

Příklad použití třídy File

```
public static void FileExample ()
{
  string myPath = @"C:\pokusny.txt";
  //pokud soubor na dane ceste neexistuje vytvorime jej
  if (!File.Exists(myPath))
  {
    Stream myStream = File.Create(myPath);
    myStream.Close();
  }
  StreamWriter myWriter = File.AppendText(myPath);
  myWriter.WriteLine(„Nejaky nas radek textu");
  myWriter.Close();
  Console.WriteLine("Cas vytvoreni : {0}",File.GetCreationTime(myPath));
  //nastaveni atributu archivace
  File.SetAttributes(myPath,FileAttributes.Archive);
  Console.WriteLine("Attributy : ",File.GetAttributes(myPath));
}
```

Úkoly 

- 1. Vytvořte program (s využitím tříd StringWriter, StringReader), který bude mít dvě metody. První funkce bude sloužit pro uložení textu do souboru. Druhá funkce bude vracet řetězec načtený ze souboru. Název souboru můžete v první verzi použít „natvrdo“. Vylepšete následně program tak aby cesta k souboru byla předávána funkcím jako parametr.
- 2. Pomocí třídy File utvořte program, který bude obsahovat metody:
  - ***CreateFileInPath:*** vytvoří soubor v cestě zadané parametrem
  - ***CopyFileTo:*** kopíruje soubory z cesty zadané prvním parametrem do cesty zadané druhým parametrem
  - ***CheckFileSize:*** vrátí velikost souboru zadaného v parametru

  Vytvořte aplikaci a uživatelské prostředí k otestování Vámi vytvořených funkcí

  Použijte nápovědu na MSDN

Úkoly

3. Vytvořte metodu, která pomocí třídy Stream a TextReader vytvoří dvojrozměrné pole čísel z textového souboru, kde jednotlivá čísla jsou oddělena čárkou (mezerou, atd.) a přechodem na další řádek.

   Příklad obsahu souboru:

   5,88,125,600

   99,56,78,200

   5,5,5,5

   7,7,7,7

   Parametry Vámi vytvořené funkce budou:

   – Cesta k souboru (string)
   – Oddělovač jednotlivých čísel (char)

- Vytvořte jednoduchou aplikaci, na které otestujete funkčnost Vámi vytvořené metody