Práce se soubory

A) Třídy File a FileStream: (System.IO. ...)

```
byte[] data1 = { 1, 2, 3, 4, 5 };
byte[] data2 = new byte[100];

string jmeno = "file.dat";

FileStream fs = File.Open(jmeno, FileMode.OpenOrCreate);
fs.Write(data1);
fs.Close();

fs = File.Open(jmeno, FileMode.Open);
fs.Read(data2);
fs.Close();
```

Práce se soubory

B) Třídy StreamReader a StreamWriter: (System.IO. ...)

```
string jmeno = "file.txt";

StreamWriter sw = new StreamWriter(jmeno);
sw.WriteLine("Text zapisovaný do souboru");
sw.Close();

System.Text.Encoding.GetEncodings()
StreamReader sr = new StreamReader(jmeno);
Console.WriteLine(sr.ReadLine());
sr.Close();
```

File/FileStream čte a píše bajty,

StreamReader/StreamWriter čte a píše znaky.

Práce se soubory

S kodováním:

```
StreamWriter sw = new StreamWriter(jmeno, false,
                                   System.Text.Encoding.GetEncoding(1250));
```

Ale pro .NET Core je potřeba předtím ještě zavolat

```
System.Text.Encoding.RegisterProvider(
        System.Text.CodePagesEncodingProvider.Instance);
```

https://stackoverflow.com/questions/50858209/system-notsupportedexception-no-data-is-available-for-encoding-1252

Otevření souboru podle přípony

```
System.Diagnostics.ProcessStartInfo psi = new
    System.Diagnostics.ProcessStartInfo("file.txt");

psi.UseShellExecute = true;

System.Diagnostics.Process.Start(psi);
```

String.Format()

vrací string vytvořený z řetězce a objektů

```
Decimal cena = 17.36m;
String s = String.Format("Aktuální cena je {0} za kg.", cena);
Console.WriteLine(s);
```

```
Aktuální cena je 17,36 za kg.
```

Je možné uvést formát, šířku apod.:

```
s = String.Format("At {0}, the temperature is {1}°C.",
                  DateTime.Now, 20.4);
Console.WriteLine(s);
```

```
At 06.05.2026 11:58:01, the temperature is 20,4°C.
```

https://learn.microsoft.com/cs-cz/dotnet/fundamentals/runtime-libraries/system-string-format#get-started-with-the-stringformat-method

String.Format() - parametr Culture

Je možné specifikovat parametr Culture – národní zvyklosti (čárka nebo tečka, formát datumu apod.).

```
string[] cultureNames = { "en-US", "fr-FR", "de-DE", "es-ES" };
double value = 9164.32;
Console.WriteLine("Culture     Date                                  Value");
foreach (string cultureName in cultureNames)
{
    System.Globalization.CultureInfo culture
      = new System.Globalization.CultureInfo(cultureName);
    string output = String.Format(culture, "{0,-11} {1,-30:D} {2:N}",
                                  culture.Name, DateTime.Now, value);
    Console.WriteLine(output);
}
```

```
Culture     Date                                Value
en-US       Wednesday, May 6, 2026              9,164.320
fr-FR       mercredi 6 mai 2026                 9 164,320
de-DE       Mittwoch, 6. Mai 2026               9.164,320
es-ES       miércoles, 6 de mayo de 2026        9.164,320
```

https://learn.microsoft.com/cs-cz/dotnet/api/system.string.format?view=net-10.0

String.Format() vs. Strings interpolation

```
int a = 11, b = 22, c = 33;
Console.WriteLine("Prvni: {0} Druhy: {1} Treti: {2}", a, b, c);
Console.WriteLine($"Prvni: {a} Druhy: {b} Treti: {c}");
```

```
Prvni: 11 Druhy: 22 Treti: 33
Prvni: 11 Druhy: 22 Treti: 33
```

Jaký je tom rozdíl?

= KDY se vyhodnocuje:

a) interpolace HNED

b) String.Format() až při zavolání

= Pomocí String.Format můžeme vytvářet šablony (templates).

Výjimky

Podobně jako v Pythonu

- objekt
- uvnitř zajímavé informace
- různé typy
- try-blok
- catch
- finally*)
- throw

---

*) (POZOR: nemusí se zavolat pokaždé!
https://stackoverflow.com/questions/3216046/does-the-c-sharp-finally-block-always-execute

Výjimky

```
int i = 25;

try
{
    int nula = 0;
    i = 25 / nula;
}
catch (DivideByZeroException ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.StackTrace);
    i = 777;
}

Console.WriteLine($"Pokračujeme... i={i}");
```

Výjimky nebo návratové kódy

Funkce má vracet výsledek, ale možná se ho nepodaří spočítat…

Možnosti:

A) vyvolat výjimku

B) kromě výsledku vracet i zprávu o tom, jak to dopadlo

Joel Spolsky: 13, Exceptions

Ned Batchelder: Exceptions vs. status returns

Joel Spolsky: DoSomething()

Raymond Chen: Cleaner, more elegant, and wrong

Raymond Chen: Cleaner, more elegant, and harder to recognize

Atributy

informace ukládané do překladu (CIL)

vztahují se k následujícímu prvku

zapisují se do hranatých závorek těsně před prvek

může jich být víc za sebou

lze se na ně ptát za běhu programu

```
[Obsolete("Will be removed in next version.")]
public static int Add(int a, int b)
{
    return (a + b);
}
static void Main(string[] args)
{
    Add(1, 2);
}
```

https://learn.microsoft.com/cs-cz/dotnet/standard/attributes/applying-attributes
https://learn.microsoft.com/cs-cz/dotnet/standard/attributes/retrieving-information-stored-in-attributes

Automatické testy, TDD

* unit-testy
    1. Napsat testy
    2. Spustit testy a ujistit se, že selžou
    3. Napsat kód
    4. Ladit kód, dokud nesplní testy
    5. Refaktorovat kód
    6. Opakovat
* testování funcionalit
* výkonnostní testy
* regresní testy i dnes funguje, co fungovalo včera
* Průběžná integrace (CI)