Programování

Výjimky v běhu programu

Ovládání programu

- Doposud ovládání tzv. NICE PATH (pěkná cesta)
  - Iterativní cykly
  - Cykly s neznámým počtem opakování
  - Switch – case
  - If else
  - Vstupní argumenty při spouštění aplikace

- Uživatel vkládá pouze validní vstupy a nesnaží se nám aplikaci rozbít
- Soubory, které bychom chtěli načíst existují
- Odkazy, na které ukazujeme mají správný formát
- Apod.

Výjimky

- = problém, který vznikne za běhu programu
- Ve většině případů způsobuje pád a ukončení programu a ztrátu dat
- V C# je výjimka odezva na výjimečný stav, který vyvstane za průběhu programu
  - Např.: dělení nulou
- **Exceptions** – třída pro řešení výjimek
- Výjimky ovládáme pomocí klíčových slov:
  - try
  - catch
  - finally
  - throw

Konstrukce try-catch-finally

- Konstrukce se využívá pro řešení krizové části kódu
- Krizová část kódu = část kódu, kde se může vyskytnou chybové chování a může tak způsobit výjimku v běhu programu

```
try
{
    // krizová část kódu
}
catch (Exception e) // typ výjimky, která se má zachytit
{
    // řešení výjimky - co se má stát, pokud se výjimka objeví
}
finally
{
    /*
     * volitelná část programu, která se provede
     * ať již se výjimka objeví či nikoliv
    */
}
```

Konstrukce try-catch-finally

- V rámci krizové části můžeme odchytávat více typu výjimek, která může mít specifické chování v závislosti na typu chyby

```
int errors = 0;
try
{
    // krizová část kódu

}
catch (InvalidTimeZoneException e1)
{
    Console.WriteLine("Zjištěna chyba: {0}", e1.Message);
    errors++;
}
catch (FormatException e2)
{
    Console.WriteLine("Zjištěna chyba: {0}", e2.Message);
    errors++;
}
finally
{
    Console.WriteLine("Vyskytlo se {0} vyjímek v krizové části", errors);
}
```

Třída Exceptions

- Veškeré výjimky jsou v C# vyjádřené jako třídy
- Všechny více či méně vychází z obecné třídy System.Exception
- Programátor má možnost si vytvořit i vlastní typ výjimky
  - Probereme později, jakmile se seznámíme s OOP
- Mimo obecné chyby, Exception, jsou v C# popsané i specifičtější chyby např.:
  - IndexOutOfRangesException – obvykle u polí
  - FormatExceptions – jakákoliv formátovací chyba
  - DivideByZeroException – dělení nulou
  - …

Klíčové slovo throw

- Pomocí klíčového slova **throw** můžeme vynutit vznik výjimky
- Obvykle vyhazujeme výjimku v části catch v try-catch konstrukci
- Tímto způsobem můžeme vyhazovat i námi vytvořené vyjímky
- Lze vyhazovat pouze objekty, které jsou nějakým způsobem odvozené od obecné třídy Exception – *více po probrání OOP*

```
throw new Exception("Chybová hláška");

Exception e = new Exception("Chyba v běhu programu");
throw e;
```

```
static void Main(string[] args)
{
    int numerator, denominator;
    Console.Write("Please enter the numerator: ");
    numerator = Convert.ToInt32(Console.ReadLine());
    Console.Write("Please enter the denominator: ");
    denominator = Convert.ToInt32(Console.ReadLine());
    try
    {
        Console.WriteLine("The result is {0}.", numerator / denominator);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
    finally
    {
        Console.WriteLine("----End of Error Handling Example----");
    }
}
```

Ukázka obsluhy výjimek v programu

```
Please enter the numerator: 17
Please enter the denominator: 3
The result is 5.
----End of Error Handling Example----
Press any key to continue . . .
```

NICE PATH

```
Please enter the numerator: 17
Please enter the denominator: 0
Došlo k pokusu o dělení nulou.
----End of Error Handling Example----
Press any key to continue . . .
```

Výskyt výjimky