Programování

Objektově orientované programování

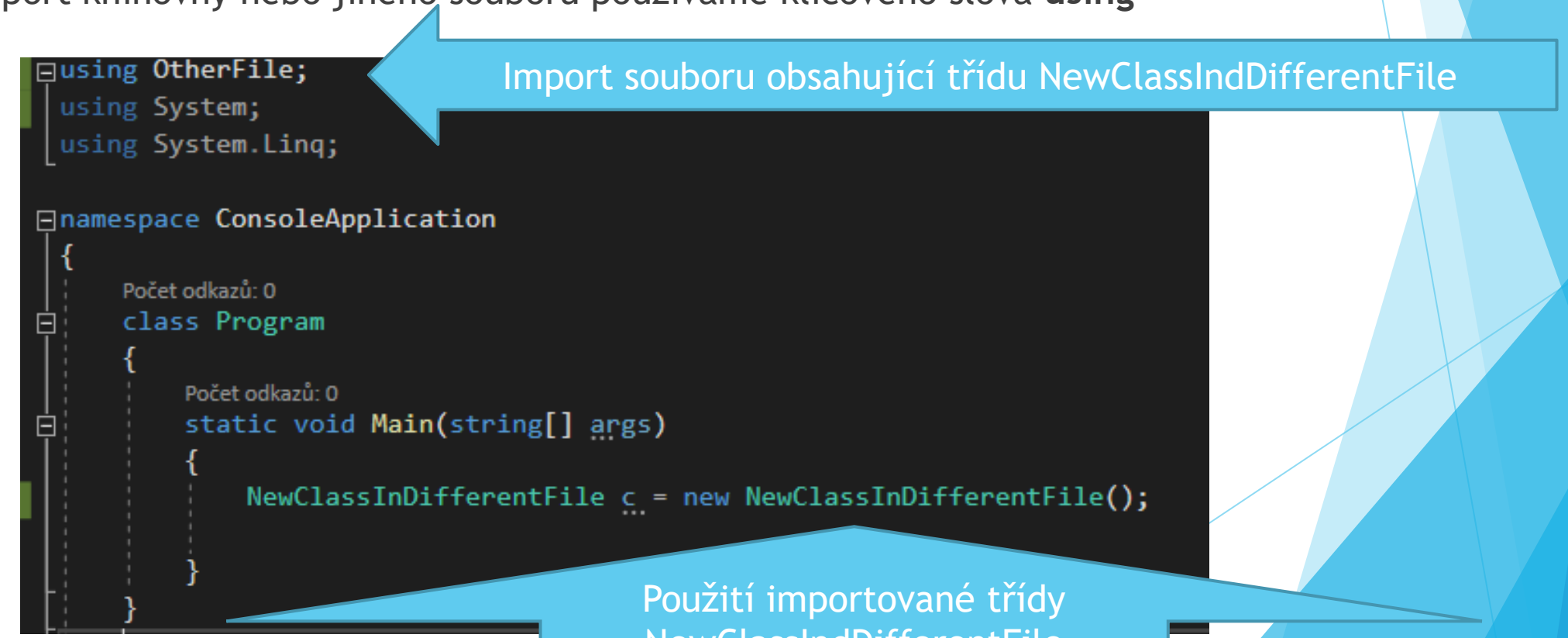
Vytvoření třídy v C#

- Každou třídu je ideální mít v samostatném souboru pro lepší strukturu celého projektu
- V menších projektech lze třídy psát do jednoho souboru
  - Stále musíme ale dodržet místo, kam ji vytvoříme
- Shromažďovat lze například třídy, které spolu souvisí případně situace, kdy malá třída tvoří komponentu třídy větší

```
namespace ConsoleApplication
{
    Počet odkazů: 0
    class Program
    {
        Počet odkazů: 0
        static void Main(string[] args)
        {

        }
    }

    Počet odkazů: 0
    class newCreatedClass
    {

    }

}
```

Import knihoven a dalších komponent

- Pokud chceme v některé třídě použít třídu z jiného souboru je nutné tyto definice importovat podobným voláním, jak voláme standardní knihovny
- Import knihovny nebo jiného souboru používáme klíčového slova **using**



Atributy nové třídy

- Atributy jsou pojmenovanými vlastnostmi vytvořené třídy
- Dodržujeme zapouzdření a jednotlivé atributy nejsou volně přístupné z vnějšího kódu
- V C# atributy v souvislosti s možností čtení a zápisu ještě rozlišujeme na
  - Fields – jednoduché atributy, které jsou skryté bez možnosti přímého přístupu
  - Properties – atributy, které nám umožňují manipulovat s fields (čtení/zápis)

```
Počet odkazů: 0
class newCreatedClass
{
    private char pismeno;

    Počet odkazů: 0
    private char Letter {
        get { return pismeno; }
        set { pismeno = value; }
    }

    Počet odkazů: 0
    private char UltimateLetter { get; set; }

}
```

Atributy nové třídy - validace

- Atributy jako fields není třeba uvádět a lze tak využít zkráceného zápisu

```
Počet odkazů: 2
private char UltimateLetter
{
    get { return UltimateLetter; }
    set { UltimateLetter = value; }
}

Počet odkazů: 0
private char UltimateLetterShort { get; set; }
```

Stejná funkcionalita, ale s použitím zkráceného zápisu

- Využitím zkráceného zápisu ovšem ztrácíme možnost validovat si metody operující s atributem – zejména tedy set

```
Počet odkazů: 2
private char UltimateLetter
{
    get { return UltimateLetter; }
    set { if (value == 'a') UltimateLetter = 'A'; }
}
```

Metody nové třídy

- Metody jsou funkcemi objektů, které umí vykonat
- Stejně jako v jiných případech pomocí modifikátorů viditelnosti můžeme určit, které funkce budou viditelné pro použití a které ne
- Metody třídy mohou pracovat s atributy třídy
- Využitím polymorfismu můžeme některé třídy, které vychází z obecné třídy Object přepsat (**override**)

```
class newCreatedClass
{
    Počet odkazů: 4
    private char Letter
    {
        get; set;
    }

    Počet odkazů: 0
    public void Triple()
    {
        Console.WriteLine(Letter + Letter + Letter);
    }

    Počet odkazů: 0
    public override string ToString()
    {
        return "Pismeno: " + Letter;
    }
}
```

Konstruktor třídy

- Konstruktor struktury obsahuje jako vstupní parametry všechny atributy
- Konstruktor třídy může obsahovat nula až maximální počet vstupních parametrů – třídy tak mohou mít více než jeden konstruktor

```
Počet odkazů: 6
class Obdelnik
{
    Počet odkazů: 4
    public int A { get; set; }
    Počet odkazů: 4
    public int B { get; set; }
    Počet odkazů: 1
    public Obdelnik() { A = 5; B = 8; }
    Počet odkazů: 1
    public Obdelnik(int x) { A = x; B = x; }
    Počet odkazů: 1
    public Obdelnik(int x, int y) { A = x; B = y; }
    Počet odkazů: 1
    public override string ToString()
    {
        return "[" + A + "," + B + "]";
    }
}
```

Vytvoření nového objektu třídy

- Nový objekt deklarujeme podobně jako jsme deklarovali struktury pomocí klíčového slova **new**
- Oproti strukturám máme výhodu, že máme více možností, jak bude náš objekt při deklaraci vypadat díky většímu počtu konstruktorů třídy

```
Počet odkazů: 0
static void Main(string[] args)
{
    Obdelnik obdelnik1 = new Obdelnik();
    Obdelnik obdelnik2 = new Obdelnik(11);
    Obdelnik obdelnik3 = new Obdelnik(16, 20);
    Console.WriteLine(obdelnik1);
    Console.WriteLine(obdelnik2);
    Console.WriteLine(obdelnik3);
}
```