Objektově orientované programování v C# .NET

Učební pomůcka od **ITnetwork.cz** - Přehled OOP syntaxe

Třída

```
class Uzivatel
{
    public string Jmeno;
    private int vek = 42;

    public void Pozdrav()
    {
        Console.WriteLine("Ahoj");
    }
}
```

```
// Vytváříme novou instanci
Uzivatel jan = new Uzivatel();

// Používáme instanci
jan.Jmeno = "Jan Nový";
jan.Pozdrav();
```

Do třídy píšeme proměnné jako její **atributy** a funkce jako její **metody**. Ty se pak ale vztahují jen k této konkrétní třídě.

Konstruktor

```
public Uzivatel(string jmeno)
{
        this.Jmeno = jmeno;
}
```

```
// Použití
// Vytvoření instance volá konstruktor
Uzivatel jan = new Uzivatel("Jan Nový");
```

Metoda bez návratového typu, která se jmenuje vždy stejně jako třída.

Gettery a Settery = Vlastnosti

```
public string Jmeno { get; private set; } // Má jen getter
public int Vek { get; set; }              // Má getter i setter

jan.Jmeno;          jan.Vek = 39;         // Použití
```

Jakmile potřebujeme vystavit **atribut veřejně**, uděláme ho **vždy jako vlastnost** (get/set). Atributy jen pro **vnitřní účely** děláme jako **privátní**.

Dědičnost

```
class Programator: Uzivatel
{
        public string Jazyk;
}
```

Dědičností přejímáme atributy a metody předka dle modifikátorů přístupu.

Modifikátory přístupu

- **public**
  - Neomezený přístup
- **protected**
  - Přístup omezen na třídu obsahující daný prvek a její potomky
- **private** (výchozí)
  - Přístup omezen na třídu obsahující daný prvek
- **internal** (výchozí u tříd)
  - Přístup omezen pouze na třídu obsahující daný prvek a stejné aktuální sestavení (solution)
- **protected internal**
- **private protected**

Převod na text

```
public override string ToString()
{
        return Jmeno;
}
```

```
// Použití
Console.WriteLine(jan);
```

Statika (static)

```
private static int delkaHesla;
public static double StupneNaRadiany(double stupne)
{
        return 3.14 * stupne / 180;
}
Matematika.StupneNaRadiany(360); // Použití
```

Patří třídě jako takové, jsou **společné** pro všechny její instance.

Konstanty (const)

```
public const plnoletost = 18;          Uzivatel.plnoletost; // Použití
```

Rozhraní (interface)

```
interface IVykreslitelny
{
        void Vykresli();
}
```

```
class Obdelnik: IVykreslitelny // Implementace
{
        public void Vykresli() { /* … */ }
}
```

Předpisuje nám metody, která by měla třída implementovat.
Třída může implementovat i více interface, ale dědit smí jen z jedné třídy.

Abstraktní třída (abstract)

```
abstract class OvladaciPrvek
{
        public abstract void Kliknuti();
        // Zbytek můžeme implementovat...
}
```

Kombinace klasické třídy a rozhraní.

Výčtové typy (enum)

```
enum BarvaOci { Cerna, Hneda, Zelena, Modra }
BarvaOci barva = BarvaOci.Zelena; // Použití
```

Jmenné prostory (namespace)

```
namespace Clanky.Administrace;
{
        public class Editor { /* … */ }
}
```

```
// Použití
using Clanky.Administrace.Editor;
```