Střední škola pedagogická, hotelnictví a služeb, Litoměřice, příspěvková organizace

Předmět:	**Vývoj aplikací**
Téma:	**Visual Studio**
Vyučující:	**Ing. Milan Káža**
Třída:	**EK3**
Hodina:	**21-22**
Číslo:	**V/5**

Programování v jazyce C a C# - Visual Studio

6. Pojmy v C#

V této části budou popsány pojmy, které patří přímo k programovacímu jazyku C# a také se s nimi pracuje ve Visual Studiu. Většina těchto pojmů a příkazů již byla popsáná v první části, zde se popíší jen specifické pro C# a Visual Studio.

Jak již bylo řečeno, jazyk C# podporuje pouze jednoduchou dědičnost, čímž jsme se vyhnuli kolizi jmen. C# ale umožňuje, aby třída implementovala několik rozhraní. V takovém to případě není možné se vyhnout kolizi jmen. Rozhraní lze rozšiřovat a kombinovat pomocí dědičnosti. Takto lze přidávat další metody, slučovat několik rozhraní do jednoho nebo předefinovat existující metodu.

Příkaz **throw** slouží k vyvolání výjimky. Uplatní se tehdy, pokud je potřebné ošetřit nějakou část kódu. V případě, kdy se vyskytnou jiné podmínky v programu, než bylo předpokládáné , je možné pomocí **throw** vyvolat jinou výjimku.

Direktiva **using**, slouží k deklaraci určitého prostoru jmen, pokud by nebyla provedena deklarace, tak by musel programátor vypisovat celý prostor jmen. Programy v C# začínají deklarací **using System;** v tomto případě je možné psát **Console.WriteLine,** není možné, ale napsat **File.Create;** příkaz **File.Create** neleží v prostoru jmen System, proto je nutné napsat **using System.IO;** a ten zajistí práci se soubory.

Příkaz **new** slouží k vytvoření nové instance, za ním se zapisuje konstruktor. Příkaz **new** vrací odkaz na nově vytvořenou instanci. Deklarace příkazu **new**.

```
Text tet;
tet=new Text("Moje C# ");
```

nyní byl vytvořen odkaz na třídu Text.

6.1.Třídy

Třída slouží k systematickému uspořádí informací do smysluplné formy, což je určité nastavení, které vykonává daná třída. To je možné najít i v běžném životě, např. auto, každý si představí jezdí, je říditelné, zrychluje, zpomaluje a brzdí atd. pokud se bude auto prezentovat jako třída potom pokud, někdo řekne auto, tak si každý představí tyto vlastnosti. I v programu je možné si vytvořit třídu, např. výpočet procent a pouze ji volat.

Třídy v C# jsou bohatší než v C ++, přesto zde není úplně všechno a něco chybí z C++. Důležité body pro třídy v C# jsou tyto:

- v C# není vícenásobná a virtuální dědičnost
- všechny třídy v C# mají společného předka což je třída **object**
- je možné vytvářet pouze dynamické instance o zrušení se stará automatická správa paměti
- třídy mohou obsahovat destruktor, ale není zaručené jeho zavolání
- třídy mohou obsahovat datové složky, metodyudálosti a přetížené operátory
- třídy mohou implementovat rozhraní
- je možné zakázat dědění tříd

To jsou pravidla třídy v C#.

Deklarace třídy

modifikátor ***class*** *identifikátor předek tělo*

modifikátor: není nutné ho uvádět, může vyjadřovat přístupová práva k třídě
class *identifikátor*: představuje jméno třídy, klíčové slovo class vyjadřuje, že se jedná o třídu
předek: specifikuje předka vytvářené třídy, zde je vztah dědičnosti mezi předky, tato část se nemusí deklarovat
tělo: může být prázdné nebo vyplněné datovými složkami, konstruktory, vlastnostmi nebo vnořenými datovými typy

Vytvoření třídy

```
 class Kruh
{
  double Area()
{
return 3.141592 * radius*radius;
}
double radius;
}
```

Po slovu **class** je uveden název třídy **Kruh**, tělo třídy obsahuje metodu **(Area)** a proměnnou **radius**.
Třídu je možné vytvořit v progamovém kódu i tímto způsobem:

```
public class Volba
{
private int odp=42;
private string moto=" Jsem neljlepší progarmátor";
}
```

V tomto případě přibyl před třídu pojem public, to ukazuje na možnost přístupu ke třídě, nebo proměnným. Jaké jsou tedy možnosti přístupu:

public– je možné přistupovat zcela volně. Z kterékoli části programu mohu zavolat třídu načíst z ní nebo do ní data a pracovat s nimi, stejně tak s proměnnými pokud budou deklarovány jako public.

private – přístup je možný pouze „zevnitř“ třídy. Zde se pracuje pouze v dané třídě do třídy je možné vkládat data poku byly mimo třídu deklarovány jako public nebo ve třídě public, pokud ne tak není možné s výsledky dané třídy pracovat v další části programu.

protected – přístup je možný pouze „zevnitř“ třídy (v metodách třídy) nebo z odvozené třídy. Vytvořením potomka je možné s proměnným ve třídě pracovat dál.

Přístupy se budou ukazovat přímo v příkladech, které si je dobré vyzkoušet. Jen jednoduchá ukázka programu.

```
public partial class Form1 : Form
  {
      double num1;
      double num2;
      double vys;

      public Form1()
      {
        InitializeComponent();
      }

      private void plus_Click(object sender, EventArgs e)
      {
          double vys1;
          num1 = Double.Parse(cislo1.Text);
          num2 = Double.Parse(cislo2.Text);
          vys1 = num1 + num2;
          vys = vys1;
          vysledek.Text = "" + vys;
      }
```

V části ***public partial class Form1 : Form*** jsou zadeklarovány proměnné ***double num1*** a další proměnné ty jsou vidět v celém programovém kódu, ale v části ***private void plus_Click(object sender, EventArgs e)***

je proměnná ***double vys1***; ta je viditelná pouze v uvedené části ***plus.***

Kontrolní otázky

1. Vysvětlete pojem Třída a příkaz new
2. Vysvětlete jakým způsobem se přistupuje ke třídám a viditelnost proměnných
3. Objasněte příkaz throw a using.

Použitá literatura:

Miroslav Virius: od C++ k C# ,1.vydání, KOOP České Budějovice, 2002
Eric Gunnerson: Začínáme programovat v C# , 1. vydání, Computer Press, 2001
John Sharp, Jon Jagger: Microsoft Visual C# .NET krok za krokem, 1. vydání, Knihy.iDnes
Miroslav Virius: C# Hotová řešení, 1. vydání, Computer Press, 2006
Amadeo Mareš: 1001 tipů a triků pro C#