Číslo a název šablony	III/2 Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo didaktického materiálu	EU-OPVK-VT-III/2-ŠR-319
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech III
Téma didaktického materiálu	Objektová hierarchie C#
Vyučovací předmět	Seminář z informatiky
Cílová skupina (ročník)	Žáci ve věku 17–18 let
Úroveň žáků	Středně pokročilí
Časový rozsah	1–2 vyučovací hodiny
Klíčová slova	Objektová hierarchie C#, listBox, metoda ToString, Controls
Anotace	Studenti se aktivně seznamují s objektovou hierarchií C# a zpracovávají data pomocí vlastních tříd
Použité zdroje	ELLER, Frank. *C# - začínáme programovat: podrobný průvodce začínajícího uživatele*. 1. vyd. Praha: Grada, 2002, 240 s. ISBN 80-247-0324-6. VYSTAVĚL, Radek. *Moderní programování: sbírka úloh k učebnici pro středně pokročilé*. 1. vyd. Ondřejov: moderníProgramování, 2008-2009, 2 sv. ISBN 978-80-903951-3-8. VYSTAVĚL, Radek. *Moderní programování: učebnice pro pokročilé*. Ondřejov: moderníProgramování, 2011, 149 s. ISBN 978-80-903951-7-6. VYSTAVĚL, Radek. *Moderní programování: učebnice pro středně pokročilé*. Ondřejov: moderníProgramování s.r.o, 2008. ISBN 978-80-903951-2-1. TIŠER, Robert a Zdeněk NOVOTNÝ. *MS POWERPOINT a ACCESS v příkladech* [CD ROM]. Firma Pachner [cit. 2013-06-24]
Typy k metodickému postupu učitele, doporučené výukové metody, způsob hodnocení, typy k individualizované výuce apod.	Text je možno využít ke společné práci, samostatné přípravě studentů, domácímu studiu apod. Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním) Prezentace obsahuje stručné shrnutí poznatků potřebných pro řešení příkladů, po jejím převedení by bylo vhodné podívat se na

	dokumentaci C#. V pracovním listu je zadání cvičení – většinou se jedná o úlohy, které by měli studenti naprogramovat samostatně. Není nutné, aby všichni zpracovali všechno, vhodné je diferencovat podle jejich zájmu a schopností. Obtížnější úlohy jsou označeny hvězdičkou. Součástí materiálu je zdrojový kód těchto příkladů. Návrh způsobu hodnocení: ohodnocení samostatné práce během hodiny např. podle volby a počtu úloh a elaborace řešení (efektivnost, komentáře…).

Metodický list k didaktickému materiálu

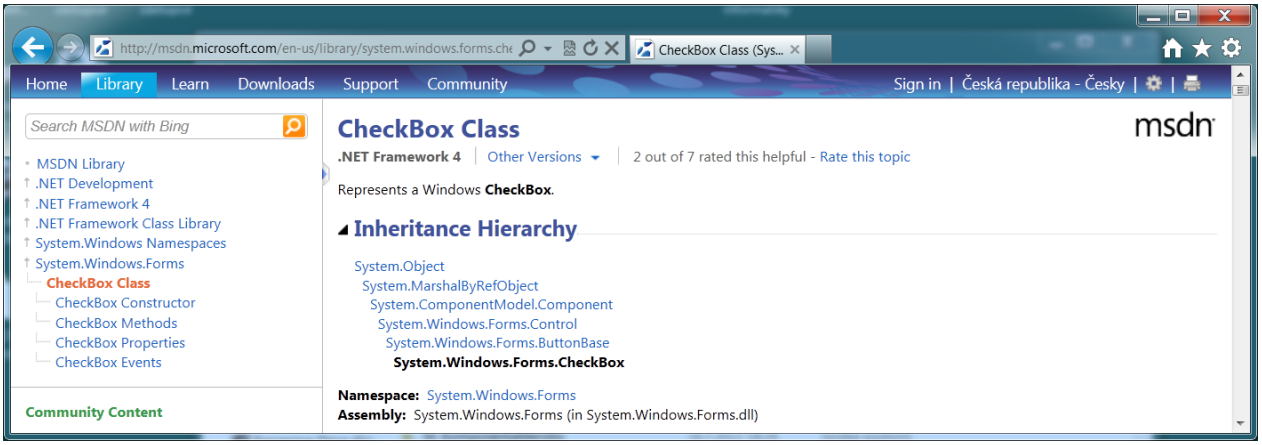
Prohlášení autora

Tento materiál je originálním autorským dílem. K vytvoření tohoto didaktického materiálu nebyly použity žádné externí zdroje s výjimkou zdrojů citovaných v metodickém listu.

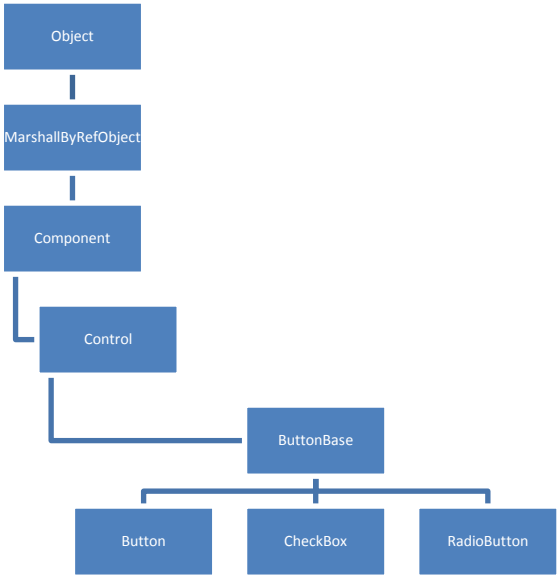
Obrázky (schémata a snímky obrazovek) pocházejí od autora.

319. Objektová hierarchie C#, metoda ToString a ListBox

Zobrazme si dokumentaci ke knihovnám .net na stránce msdn.microdoft.com a podívejme se například na CheckBox class – ze Systém.Windows.Forms. Hned na začátku vidíme, z čeho je tato třída odvozena:



Na začátku je Object, následuje MarshallByRefObject, Component, nám již známá třída Control, za ní ButtonBase. Kdybychom si zobrazili podobněButton a RadioButton, zjistili bychom, že se jedná o tři „rovnocenné“ potomky třídy ButtonBase. Graficky situaci vyjadřuje následující obrázek.



Třída Control

Obsahuje funkce společné ovládacím prvkům. Nalezneme zde vlastnosti jako BackColor, Text, metodu Refresh, událost MouseDown, atd., které má tedy každý ovládací prvek.

Tato třída má také vlastnost **Controls –** seznam ovládacích prvků na formuláři. (Ale také na jiném kotejneru např. panelu)

Příklad 1

Umístíme na okno tři tlačítka a dvě zaškrtávací políčka. Na stisknutí prvního tlačítka posuneme všechna tlačítka doprava a změníme zaškrtnutí políček.

```
private void buttonZmena_Click(object sender, EventArgs e)
        {
            foreach (Control c in Controls)
            {
                if (c is CheckBox)
                    (c as CheckBox).Checked = !(c as CheckBox).Checked;
                if (c is Button)
                    c.Left += 20;
                //není nutno přetypovat, vlastnost Left mají všichni
            }
```

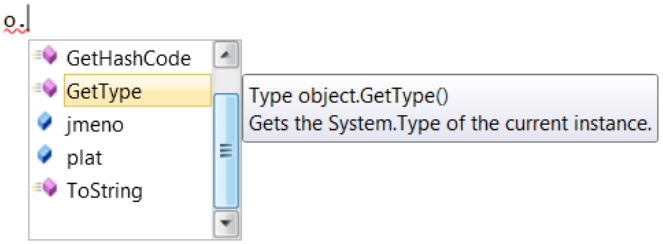
Praotec Object

Opět se podívejme do dokumentace – tentokrát na class object, která představuje vrchol dědičné hierarchie. Každá třída je jejím potomkem – neuvedeme-li bázovou třídu explicitně. Vložme do programu třídu osoba s položkami jméno (string) a plat (int), pro jednoduchost zatím bez vlastností a dalších metod.

A hned při vytváření její instance máme k dipozici její bezparametrický konstruktor – zděděný právě od třídy object.

```
Osoba o=new Osoba();
```

```
        public Form1()
        {
            InitializeComponent();
        }
```

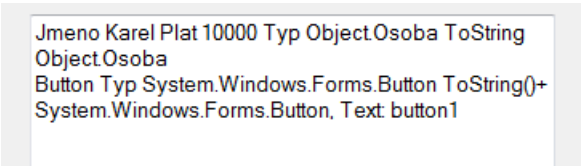


A také máme k dispozici metody, které jsme nedefinovali – mj. GetType a ToString.

V dokumentaci bychom našli, že jsou skutečně definovány u praotce Objektu.

Můžeme si je vyzkoušet v programu při výpisu do textového pole a dokonce si je můžeme vyzkoušet i pro tlačítko.

```
private void buttonOsoba_Click(object sender, EventArgs e)
        {o.jmeno = "Karel";   o.plat = 10000;
string vypis="Jmeno "+o.jmeno+" Plat
"+o.plat+" Typ
"+o.GetType().ToString()+" ToString
"+o.ToString();
            textBox1.Text+=vypis+Environment.NewLine;
            string vypis2 = "Button Typ " + button1.GetType().ToString() + "
ToString()+ "+ button1.ToString();
            textBox1.Text += vypis2 + Environment.NewLine;
        }
```



	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.
	GetHashCode	Serves as a hash function for a particular type.
	GetType	Gets the Type of the current instance.
	MemberwiseClone	Creates a shallow copy of the current **Object**.
	ReferenceEquals	Determines whether the specified **Object** instances are the same instance.
	ToString	Returns a string that represents the current object.

Je-li ovšem každá třída typu object, může být každý objekt přiřazen do proměnné tohoto typu – ano, to je ten tajemný sender v hlavičce většiny událostí ovládacích prvků:

```
private void buttonObjekty_Click(object sender, EventArgs e)
```

Příklad 2

Můžeme si pohrát s objekty různých druhů:

String, Color,int, Random:

```
  Osoba o=new Osoba();

        List<object> objekty = new List<object>();

        public Form1()

        {

            InitializeComponent();

            object barva = Color.Red;

            object cislo = 100;

            object slovo = "Ahoj";

            object nahoda = new Random();

            object clovek = new Osoba();

            (clovek as Osoba).jmeno = "Jarmila";

            (clovek as Osoba).plat = 20000;

            objekty.Add(barva);

            objekty.Add(cislo);

            objekty.Add(slovo);

            objekty.Add(nahoda);

            objekty.Add(clovek);
```

```
        }
```

…

```
private void buttonObjekty_Click(object sender, EventArgs e)
        {
            foreach(object ob in objekty)
                textBox1.Text += ob.ToString() + Environment.NewLine;
          }
```

Metoda ToString a ListBox

Metoda ToString je ve třídě object definována virtuální, ostatní třídy ji předefinovávají. Můžeme tak učinit i my u naší osoby a trochu pozměnit náš předchozí příklad:

```
 public  override string ToString()
{
    return " Jméno: " + jmeno + " Plat: " + plat.ToString();
}
```

Ano, teď je ve výpisu Jméno: Jarmila Plat: 20000.

Komponentu ListBox jsme zatím používali pro zobrazení řetězců – ale my do ní můžeme umístit libovolné objekty, které se v něm zobrazí svou metodou ToString.

Přidejme tento ovládací prvek na náš formulář a vyzkoušejme nový výpis.

```
… foreach (object ob in objekty)
                listBox1.Items.Add(ob.ToString());
```

Jakou to má výhodu? Vybereme-li položku listboxu, máme k diposzici objekt v ní obsažený – můžeme ho tedy nějak použít:

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            string zprava;
            if (listBox1.SelectedIndex < 0)
                zprava = "není nic vybráno";
            else
            {
               object obj=listBox1.SelectedItem;
               zprava = obj.ToString();
               MessageBox.Show(zprava);
```

```
        }
    }
```

Metoda listBox1_SelectedIndexChanged se volá při změně zaškrtnutí položky. Zajímavější využití si vyzkoušíme ve cvičení.

Důležité

Do proměnné typu předka lze přiřadit jakéhokoliv potomka.

Všechny třídy v C# jsou odvozeny z bázové třídy **Object**, do instance této třídy lze tedy přiřadit prakticky cokoliv.

Při přetypování se používají opreátory is a as.

Třída **Control** je společným základem všech ovládacích prvků.

Metodu **ToString** definovanou pro třídu Object předefinovává mnoho potomků (můžeme i my), dá se vhdoně využít při zobrazení vlastních objektů v listboxu.

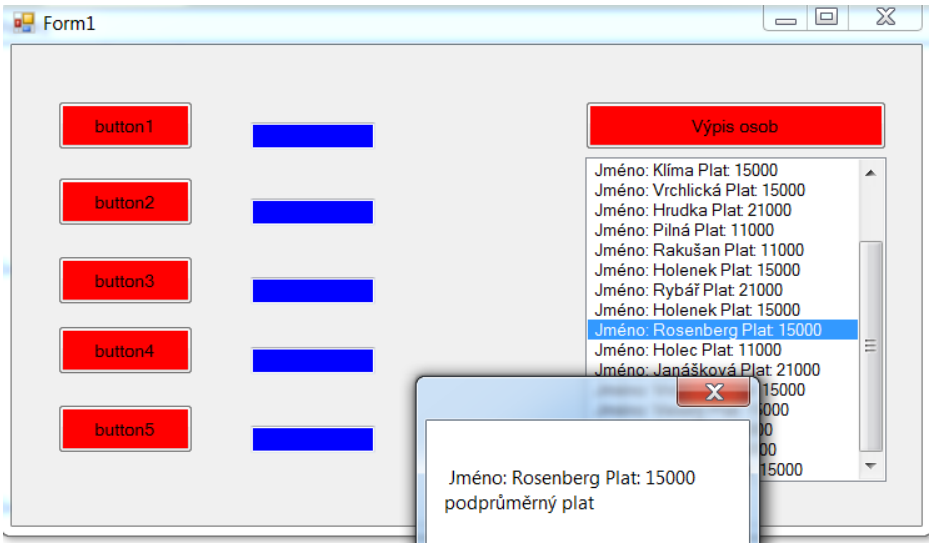
Pracovní list

Cvičení

1. Vložte na formulář několik textových políček a tlačítek a při dvojkliku na formulář se textová políčka obarví modře a tlačítka červeně.´

2. Připojte do třídy osoba bezparametrický konstruktor a konstruktor, jehož vstupem bude řetězec obsahující jméno a plat osoby oddělený středníkem.

3. Zobrazte do listboxu soubor Lide.csv a při kliknutí na položku zobrazit údaje o vybrané osobě a zdá má nadprůměrný nebo podprůměrný plat.



Řešení

```
class osoba
    {
        string jmeno;
        int plat;
        public osoba()
        {
        }
        public osoba(string s)
        {//načte hodnoty z řetězce, který obsahuje údaje, oddělené středníkem
            string[] hodnoty = s.Split(';');
            jmeno = hodnoty[0];
            plat = Convert.ToInt32(hodnoty[1]);
        }

        public string Jmeno
        {
            get {return jmeno;}
            set { value = jmeno; }
        }
        public int Plat
        {
            get { return plat; }
            set { value = plat; }
        }

        public override string ToString()
        {
            return " Jméno: " + jmeno + " Plat: " + plat.ToString();
        }
    }
```

```
public partial class Form1 : Form
    {
        double prumer = 0;
        List<osoba> osoby=new List<osoba>();
        public Form1()
        {
            InitializeComponent();
        }



        private void Form1_Load(object sender, EventArgs e)
        {
            StreamReader data = new StreamReader("Lide.csv",Encoding.Default);

            int pocet=0;
            string radek;
            while ((radek = data.ReadLine()) != null)
            {
                pocet++;
                osoba o = new osoba(radek);
                prumer += o.Plat;
                osoby.Add(o);
                listBox1.Items.Add(o.ToString());
            }
            prumer /= pocet;
            MessageBox.Show("počet osob: " + pocet.ToString() + Environment.NewLine +
                   "průměrný plat: " + prumer.ToString() + Environment.NewLine);
                     }

        private void Form1_Click(object sender, EventArgs e)
        {
```

```
        foreach (Control c in Controls)
        {
            if (c is Button) c.BackColor = Color.Red;
            if (c is TextBox) c.BackColor = Color.Blue;
        }
    }

    private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        string hodnoceni;
        osoba o = osoby[listBox1.SelectedIndex];
        if (o.Plat>prumer)  hodnoceni="nadprůměrný plat";
        else  hodnoceni="podprůměrný plat";
        MessageBox.Show(o.ToString()+Environment.NewLine+
                  hodnoceni);
    }
}
```