Číslo a název šablony	III/2 Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo didaktického materiálu	EU-OPVK-VT-III/2-ŠR-311
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech III
Téma didaktického materiálu	Objekty – proměnné
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	Objekt, třída, zapouzdření, modifikátory přístupu
Anotace	Studenti začínají programovat vlastní objekty – definují je a vytvářejí proměnné těchto tříd a jejich další zpracování
Použité zdroje	ELLER, Frank. *C# - začínáme programovat: podrobný průvodce začínajícího uživatele*. 1. vyd. Praha: Grada, 2002, 240 s. ISBN 80-247-0324-6. OCHRANOVÁ, Renata a Michal KOZUBEK. *Objektově orientované programování v Turbo Pascalu*. 1. vyd. Brno: Masarykova univerzita, 1993, 117 s. ISBN 80-210-0659-5. 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.
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. 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.

311. Objekty I – proměnné

Se třídami a objekty pracujeme v C# od začátku, ale teprve teď se naučíme vytvářet vlastní. Strukturované programování, kterým jsme se zabývali až dosud, se hodí spíš pro menší programové celky vytvářené jediným programátorem. Na tvorbě různých programových jednotek se sice může podílet více osob, ale při nezbytných úpravách hotových procedur se neobejdeme bez znalosti zdrojového kódu. Návrh datových struktur, procedur a funkcí je oddělen.

Základní myšlenka OOP se dá zhruba vyjádřit následujícím způsobem:

Program provádí určité akce s daty. Chceme-li, aby byl rozšiřitelný a opakovaně použitelný, je lepší jeho návrh založit spíš na datech než akcích. Data jsou obvykle dána a nepodléhají tolik změnám. Vývoj programu pak představuje úpravy starých a tvorbu nových akcí.

Když budeme chtít charakterizovat určitý objekt reálného světa – třeba kočku, uvědomíme si, že má nějaké vlastnosti – velikost, barvu, drápy… ale také něco umí. (chytat myši, pít mléko…) Samozřejmě také myš je objektem, (domácí, bílá, počítačová…) se svými metodami (prchnout před kočkou, prokousat se z bedýnky…) a myš a kočka při setkání komunikují – pomocí toho co umějí a mění své vlastnosti. (živá a mrtvá myš, sytá a hladová kočka …)

Nejprve tedy navrhujeme objekty s jejich vlastnostmi (vlastní data), dovednostmi (procedury a funkce) a teprve potom tyto metody implementujeme. (Zde se ovšem bez strukturovaného programování neobejdeme)

Program pak vzniká jako zápis komunikace mezi těmito objekty.

Základní rysy objektového programování:

Zapouzdření

Objekt obsahuje datové položky (proměnné, vlastnosti) a procedury a funkce (metody), které s těmito položkami pracují. Navenek vystupuje jako celek, k vlastnostem přistupujeme výhradně prostřednictvím metod.

Přístup k datům se dá omezit tak, aby k nim mohl přistupovat pouze oprávněný vlastník.

Dědičnost

Vlastnosti i metody se dědí z předka na potomka, nově vytvářený objekt může toto dědictví rozšiřovat a modifikovat, takže se vytváří hierarchický systém objektů.

Jednou vytvořené rozhraní lze tedy opakovaně používat, a tudíž se podstatně zvětšuje efektivita psaní programů.

Polymorfismus

Umožňuje zaměnitelnost objektů v rámci hierarchie. Určitá akce pak může být společná různým objektům hierarchie a její implementace pro různé objekty se může lišit.

Připomeňme si teď nám dobře známý objekt C# – okno. (Formulář)

Proměnné okna: textová políčka, tlačítka, panely…

Vlastnosti okna: Text (popisek), BackColor, DoubleBuffered…

Události: Load, Paint – to, co programujeme

Metody: Close, Refresh

Nejdůležitější součásti objektů, které najdeme ve všech objektových jazycích jsou přitom proměnné a metody, vlastnosti a události jsou pak typické pro platformu .NET a C#.

Třídy

Jak souvisí pojem objekt a třída? Můžeme si představit, že třída je jakási šablona, podle které pak vytváříme objekty. Můžeme mít několik objektů téže třídy, o nichž pak hovoříme jako o jejích instancích. Vždy je nejprve třeba vytvořit třídu, jako datový typ, potom její objekty – instance. Třída může mít i statické složky, které se nepoužívají k vytváření instancí. Běžné složky se určují jako: Jmeno_Objektu.složka_Objektu – např. textBoxVstup.Text, zatímco ke statickým přistupujeme přes Jmeno_Tridy.složka_Tridy – např. Convert.ToString.

Příklad

Budeme pracovat s objektem kolečko – který budeme charakterizovat souřadnicemi středu, poloměrem, barvou a tloušťkou kreslicí čáry. Všechny tyto charakteristiky budou tentokrát součástí definice kolečka – hovoříme o **zapouzdření –** místo nezávislých proměnných pracujeme s jedním celkem.

C# každou novou třídu umísťuje do nového zdrojového textu, v novém projektu zvolíme **Project/Add Class** a zdrojový text pojmenujeme kolo.cs.

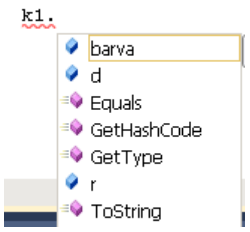
```
using System.Drawing;

//abychom mohli použít Color

namespace Objekty1

{

    class Kolo

    {

        public int x, y, r, d;

        public Color barva;

        //x,y - střed, r - poloměr, d - tloušťka pera

    }

}
```

Klíčové slovo **public** na začátku deklarace je **modifikátor přístupu** – říká, že k těmto proměnným můžeme přistupovat z kterékoliv části programu. Pokud bychom ho neuvedli, předpokládá se přístup **private,** kdy je složka přístupná pouze z objektu samotného.

Ve zdrojovém kódu Form1.cs pak můžeme deklarovat členskou proměnnou `Kolo k1 a pak vytvořit jeho instanci voláním konstruktoru`

```
k1=new(Kolo);
```

Našeptávač nám nabídne vlastnosti, které jsme definovali my a také vlastnosti, které má v C# jakýkoliv objekt.



Ke složkám můžeme přistupovat tečkovanou notací nebo je inicializovat ve složených závorkách při volání konstruktoru, jak ukazuje následující výpis.

```
public partial class Form1 : Form

    {

      Kolo k1, k2,pk; //deklarace koleček, pk bude to, které se bude
      pohybovat

        public Form1()

        {

            InitializeComponent();

        }
```

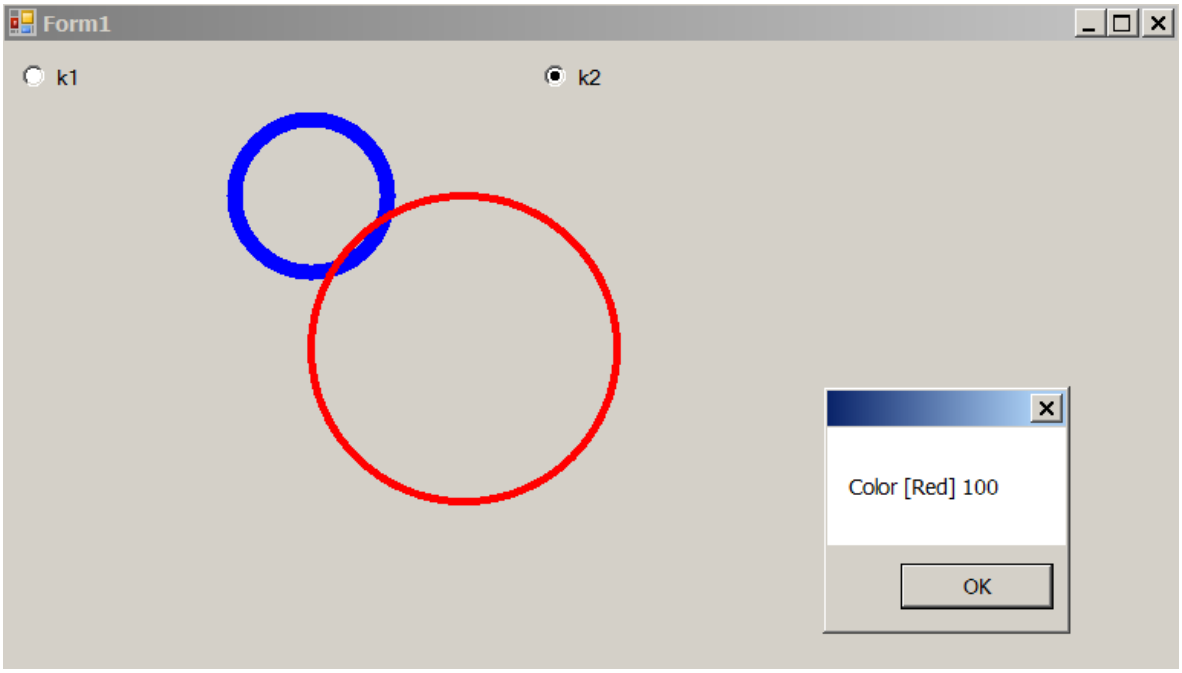
```
        private void Form1_Load(object sender, EventArgs e)
        {
              //vytvoření koleček
              pk = k1;
              k1=new Kolo();
              k1.r = 50;
              k1.x = 200;
              k1.y = 100;
              k1.barva = Color.Blue;
              k1.d = 10;
              k2 = new Kolo() { r = 100, x = 200, y = 200, d = 2, barva =
             Color.Red };
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            //kreslení koleček
            Graphics kp=e.Graphics;
            Pen p1=new Pen(k1.barva,k1.d);
            kp.DrawEllipse(p1, k1.x - k1.r, k1.y - k1.r, k1.r*2, k1.r*2);
            Pen p2 = new Pen(k2.barva, k2.d);
            kp.DrawEllipse(p2, k2.x - k2.r, k2.y - k2.r, k2.r*2, k2.r*2);
        }
    }
}
```

Kreslení koleček s využíváním jejich objektových vlastností je ovšem poněkud komplikované, tady by bylo skvělé mít nějakou metodu Kolo.kresli, která by se o všechno postarala sama – vytvářet metody jako součást objektů se naučíme v příští kapitole.

Na závěr příkladu si ještě připomeneme animace – podle volby uživatele budeme pohybovat prvním nebo druhým kolečkem. Událost bude společná pro oba radioButtony, pomocí kterých se kolečko volí.

Můžeme si také vytisknout informaci o vybraném kolečku pomocí metody ToString:

```
MessageBox.Show(pk.barva.ToString()+" "+pk.r.ToString());
```




```
private void radioButtonK1_CheckedChanged(object sender, EventArgs e)
        {
            //výběr pohybujícího se kolečka
            if (radioButtonK1.Checked)
                pk = k1;
            if (radioButtonK2.Checked)
                pk = k2;
            for (int i = 0; i < 100; i++)
            {
                pk.x++;
                Thread.Sleep(20);
                Refresh();
            }
        }
```

Všimněte si, že nevytváříme další instanci pk třídy Kolo – je to dáno tím, že proměnné objektových typů obsahují pouze odkaz na objekt.

Důležité

Zapouzdření

Objekt obsahuje datové položky a metody, které s těmito položkami pracují. Navenek vystupuje jako celek. Třída je jakási šablona, podle které pak vytváříme objekty. Můžeme mít několik objektů téže třídy, o nichž pak hovoříme jako o jejích instancích.

Nejprve je třeba vytvořit třídu, jako datový typ, potom její objekty – instance.

Třída může mít i statické složky, které se nepoužívají k vytváření instancí.

Běžné složky se určují jako: **Jmeno_Objektu.složka_Objektu**

Statické složky: **Jmeno_Tridy.složka_Tridy**.

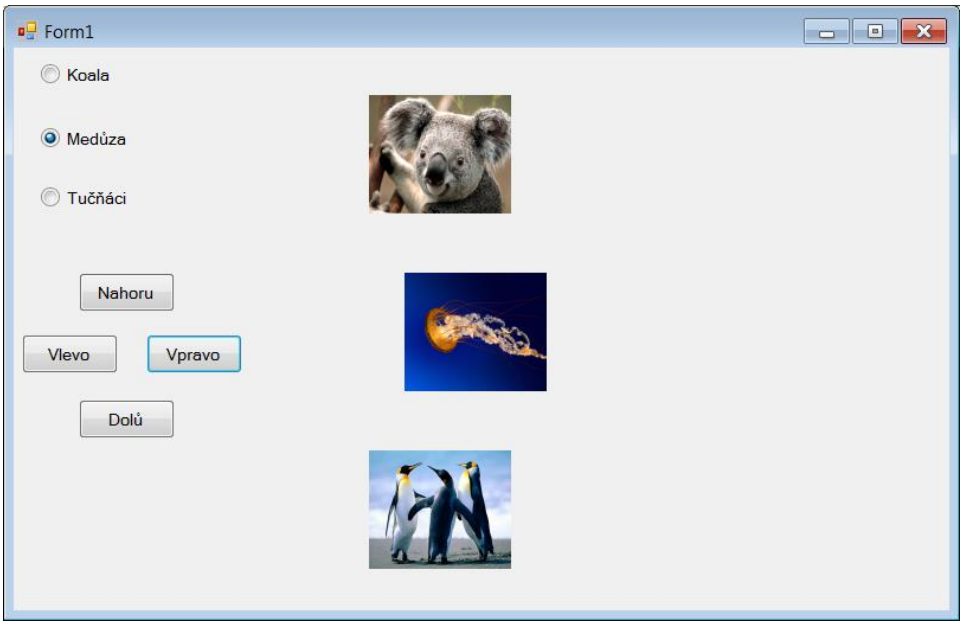
Přístup k datům se dá omezit tak, aby k nim mohl přistupovat pouze oprávněný vlastník (modifikátory public, private)

C# každou novou třídu umísťuje do nového zdrojového textu, vytvoření: Project/Add Class

```
class Jmeno_Tridy
{
    public Typ_promenne Promenna;
    private Typ_promenne Promenna;
   …  }
```

Pracovní list

Cvičení



Vypracujte nový projekt, kdy vytvoříte třídu Obraz – bude obsahovat obrázek a jeho souřadnice. (Použijte vestavěné obrázky) Vytvořte několik instancí tohoto typu a pohybujte s nimi pomocí tlačítek, případně si vymyslete nějakou animaci.

Řešení

```
namespace _311_Cvičení
{
    class Obraz
    {
        public int x, y;
        public Image obr;
    }
}
```

```
namespace _311_Cvičení
{
    public partial class Form1 : Form
    {
       Obraz Koala = new Obraz() { x = 300, y = 40, obr = Properties.Resources.Koala };
       Obraz Meduza = new Obraz() { x = 300, y = 190, obr =
       Properties.Resources.Jellyfish };
       Obraz Tucnaci = new Obraz() { x = 300, y = 340, obr =
       Properties.Resources.Penguins };
       Obraz ten; // se kterým se po dosazení bude pohybovat
        …

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
           //vykreslení obrázků
            Graphics kp = e.Graphics;
            kp.DrawImage(Koala.obr, Koala.x, Koala.y,120,100);
            kp.DrawImage(Meduza.obr, Meduza.x, Meduza.y,120,100);
            kp.DrawImage(Tucnaci.obr, Tucnaci.x, Tucnaci.y,120,100);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            ten = Koala;
        }

        private void Vyber_obraz(object sender, EventArgs e)
        {
            //výběr objektu
            if (radioButtonKoala.Checked) ten = Koala;
            if (radioButtonMeduza.Checked) ten = Meduza;
            if (radioButtonTucnaci.Checked) ten = Tucnaci;
        }
```

```
        private void buttonUp_Click(object sender, EventArgs e)
        {
            //posuv vybraného objektu nahoru
            ten.y -= 5;
            Refresh();
        }

        private void buttonRight_Click(object sender, EventArgs e)
        {
            //posuv vybraného objektu doprava
            ten.x += 5;
            Refresh();
        }

        private void buttonLeft_Click(object sender, EventArgs e)
        {
            //posuv vybraného objektu doleva
            ten.x -= 5;
            Refresh();
        }

        private void buttonDown_Click(object sender, EventArgs e)
        {
            //posuv vybraného objektu dolů
            ten.y += 5;
            Refresh();
        }
    }
}
```