| | |
|---|---|
| Číslo a název šablony | III/2 Inovace a zkvalitnění výuky prostřednictvím ICT |
| Číslo didaktického materiálu | EU-OPVK-VT-III/2-ŠR-316 |
| Druh didaktického materiálu | DUM |
| Autor | RNDr. Václava Šrůtková |
| Jazyk | čeština |
| Téma sady didaktických materiálů | Programování v C# v příkladech III |
| Téma didaktického materiálu | Objekty IV – vlastnosti |
| Vyučovací předmět | Seminář z informatiky |
| Cílová skupina (ročník) | Žáci ve věku 17–18 let |
| Úroveň žáků | Mírně pokročilí |
| Časový rozsah | 1–2 vyučovací hodiny |
| Klíčová slova | Třídy, metody, proměnné, vlastnosti |
| Anotace | Studenti se seznamují s koncepcí vlastností v C# a programují třídy, kde vlastnosti využívají k zpracování tabulkových dat. |
| Použité zdroje | ELLER, Frank. *C# - začínáme programovat: podrobný průvodce začínajícího uživatele*. 1. vyd. Praha: Grada, 2002, 240 s. ISBN 80-247-0324-6.<br><br>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.<br><br>VYSTAVĚL, Radek. *Moderní programování: učebnice pro pokročilé*. Ondřejov: moderníProgramování, 2011, 149 s. ISBN 978-80-903951-7-6.<br><br>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.<br>Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním).<br>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:<br>ohodnocení samostatné práce během hodiny např. podle volby a počtu úloh a elaborace řešení (efektivnost, komentáře…). |

**Metodický list k didaktickému materiálu**

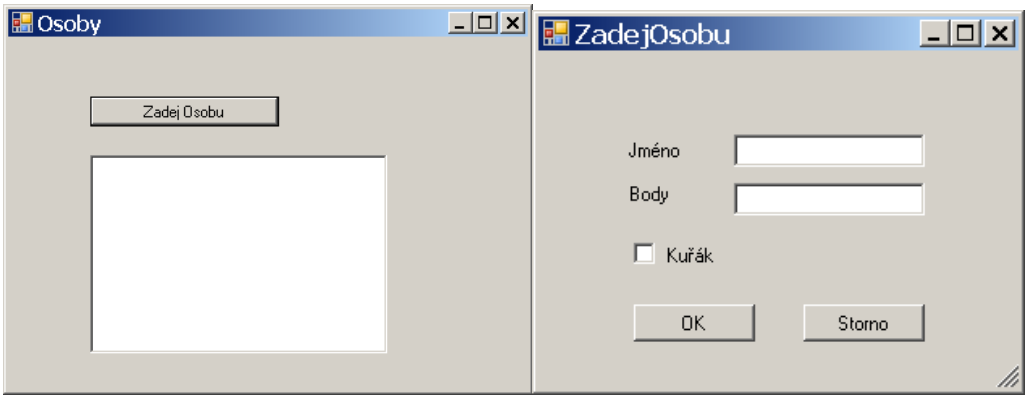
**Prohlášení autora**

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.

# 316. Objekty IV – vlastnosti

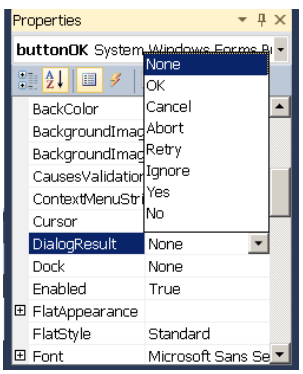
Připomeneme si práci s dalším oknem – tentokrát budeme chtít mít data z druhého okna v dispozici v prvním okně. Prostřednictvím druhého okna (dialogového) budeme zadávat údaje o osobách – jméno, body, je nebo není kuřák, při stisknutí tlačítko OK je zatím přepíšeme do prvního okna.




Druhé okno (vlastně jeho třídu) vložíme do projektu přes **Project/Add Windows Form,** nazveme ho ZadejOsobu a vybavíme potřebnými ovládacími prvky – textBoxJmeno, textBoxBody, checkBoxKurak s adekvátními popisky a tlačítky buttonOK a buttonStorno.´

Okno ZadejOsobu bychom mohli zobrazit:

```
private void buttonZadej_Click(object sender, EventArgs e)
        {
            ZadejOsobu Zadej = new ZadejOsobu();
            Zadej.ShowDialog();
        }
```




Budeme ale potřebovat zavřít okno Zadejosobu tlačítky OK a Storno a také umět rozlišit, kterým tlačítkem bylo vlastně uzavřeno. C# k tomu má

elegantní prostředek – vlastnost **DialogResult** formuláře. Pomocí editoru vlastností nastavíme oběma tlačítkům hodnoty vlastnosti DialogResult – OK a Cancel.

Pak můžeme rozlišovat:

```
if (Zadej.ShowDialog() == DialogResult.OK)
```

…

Jak se dostat k hodnotám druhého okna? Našeptávač je nenabízí, teoreticky bychom k nim mohli přistupovat, kdyby měli modifikátor public, (Můžete vyzkoušet např. u textBoxuJmeno – vlastnost Modifiers v okně Properties – potom už máte k dispozici Zadej.textBoxJmeno.Text) ale prakticky je to nevýhodné. Jakmile bychom totiž v okně Zadejosobu cokoliv změnili, museli bychom to přepracovat v celém programu. Proto také systém všude standardně používá modifikátory private Lepší je použít soukromé (private nebo neoznačené) proměnné a veřejné metody a přistupovat pak k proměnným pouze pomocí metod.

```
public ZadejOsobu()
        {
            InitializeComponent();
        }
        public string ZjistiJmeno()
        {
            return textBoxJmeno.Text;
        }
public int ZjistiBody()
        {
            return Convert.ToInt32(textBoxBody.Text);
        }
        public bool ZjistiKuraka()
        {
            return (checkBoxKurak.Checked);
        }
```

Ve výchozím okně potom:

```
private void buttonZadej_Click(object sender, EventArgs e)
        {
            ZadejOsobu Zadej = new ZadejOsobu();
```

```
            if (Zadej.ShowDialog() == DialogResult.OK)
            {
                string vyst = Zadej.ZjistiJmeno() + " " +
Zadej.ZjistiBody().ToString();
                if (Zadej.ZjistiKuraka())
                    vyst += " Kuřák";
                else
                    vyst += " Nekuřák";
                textBoxOsoby.Text += vyst + Environment.NewLine();
            }
            else
                MessageBox.Show("Storno");
        }
```

C# má ale ještě lepší možnost řešení veřejného rozhraní – **vlastnosti**.

Vlastnost se sice používá jako proměnná, ale v podstatě je dvojicí metod.

**Get –** metoda, která za slovem **return** vrací hodnotu vlastnosti

**Set –** provádí přiřazení hodnoty do vlastnosti**,** přiřazovaná hodnota je **value**.

Chybí-li část get, je vlastnost jen k zápisu, chybí-li set, pouze ke čtení.

Přesnou syntaxi si ukážeme na našem příkladu.

ZadejOsobu

```
public string Jmeno
        {
            get
            {
                return textBoxJmeno.Text;
            }
            set
            {
                textBoxJmeno.Text = value;
```

```
            }
        }
        public int Body //jen ke čtení
        {
            get
            {
                return Convert.ToInt32(textBoxBody.Text);
            }
          }
        public bool Kurak //jen ke čtení
        {
            get
            {
                return (checkBoxKurak.Checked);
            }
        }
```

FormOsoby

…

```
ZadejOsobu Zadej = new ZadejOsobu();
            Zadej.Jmeno = "Zadejte jméno osoby";
            if (Zadej.ShowDialog() == DialogResult.OK)
            {
                string vyst = Zadej.Jmeno + " " + Zadej.Body.ToString();
                if (Zadej.Kurak)
              vyst += " Kuřák";
            else
              vyst += " Nekuřák";
                textBoxOsoby.Text += vyst + Environment.NewLine;
```

…

Nabízí se ještě připravit si třídu osoba a vybavit ji stejnými vlastnostmi a metodou převodu na řetězec.

```
class Osoba
    {
         string jmeno;
         int body;
         bool kurak;
        public string Jmeno
        {
            get
            {
                return jmeno;
            }
            set
            {
                jmeno = value;
            }
        }
        public int Body
        {
            get
            {
                return body;
            }
            set
            {
                body = value;
            }
        }
        public bool Kurak
        {
            get
            {
```

```
            return kurak;
        }
        set
        {
            kurak = value;
        }
    }
    public string OsobaRet()
    {
        string vyst = jmeno + " " + body.ToString();
        if (kurak)
            vyst += " Kuřák";
        else
            vyst += " Nekuřák";
        return vyst;
    }
  }
}
```

V C# se používá konvence, kdy se soukromé proměnné a odpovídající veřejné vlastnosti jmenují stejně, ale proměnné začínají malým písmenem a vlastnosti velkým.

FormOsoby

…

```
Osoba o = new Osoba();
            ZadejOsobu Zadej = new ZadejOsobu();
            Zadej.Jmeno = "Zadejte jméno osoby";
            if (Zadej.ShowDialog() == DialogResult.OK)
            {
                o.Jmeno = Zadej.Jmeno;
                o.Body = Zadej.Body;
                o.Kurak = Zadej.Kurak;
                string pom = o.OsobaRet();
```

```
MessageBox.Show(pom);
```

## Důležité

Z hlediska modifikovatelnosti programu je vhodné proměnné deklarovat jako soukromé (**private)** a přistupovat k nim pomocí veřejných (**public)** metod nebo vlastností.

**Vlastnost** se sice používá jako proměnná, ale v podstatě je dvojicí metod.

**Get –** metoda, která za slovem **return** vrací hodnotu vlastnosti

**Set –** provádí přiřazení hodnoty do vlastnosti**,** přiřazovaná hodnota je **value**.

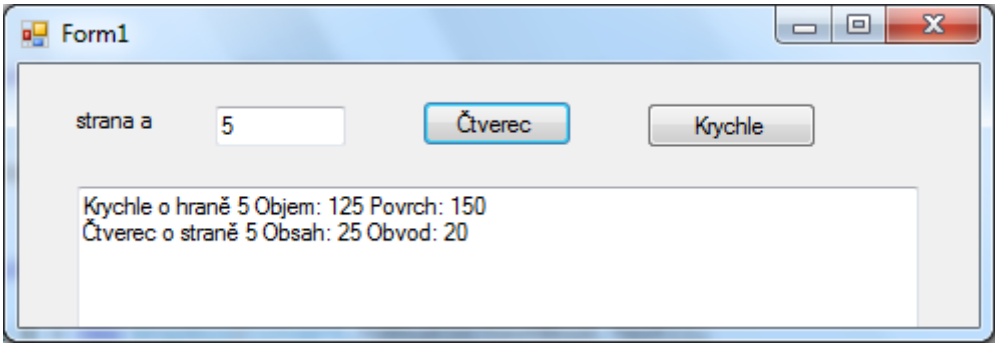
Chybí-li část get, je vlastnost jen k zápisu, chybí-li set, pouze ke čtení.

## Pracovní list

### Cvičení

Připravte nový projekt s třídami čtverec a krychle. Obě budou mít proměnnou (strana, hrana) a odpovídající vlastnost, bezparametrický a parametrický konstruktor, metody pro výpočet obvodu a obsahu, případně objemu a povrchu a metodu Info, která převede veškeré údaje na řetězec.

Otestujte třídy v programu podle vzoru:




### Řešení

```
namespace Vlastnosti_Cviceni
{
    class Ctverec
    {
        double a;       //strana
        public double A //vlastnost strana
        {
            get
            {
                return a;
            }
            set
            {
                a = value;
            }
```

```
        }
        public Ctverec() //bezparam. konstruktor
        {
        }
        public Ctverec(double a) //param. konstruktor
        {
            this.a = a;
        }
        public double Obsah()
        {
            return a * a;
        }
        public double Obvod()
        {
            return a * 4;
        }
        public string Info()
        {
            return "Čtverec o straně " + a.ToString() + " Obsah: "
                + Obsah().ToString() + " Obvod: " +
                Obvod().ToString();
        }
    }
class Krychle
    {
        double a;       //hrana
        public double A //vlastnost hrana
        {
            get
            {
                return a;
            }
            set
            {
                a = value;
            }
        }
         public Krychle() //bezparam. konstruktor
        {
        }
        public Krychle(double a) //param. konstruktor
        {
```

```
            this.a = a;
        }
        public double Povrch()
        {
            return 6*a * a;
        }
        public double Objem()
        {
            return a * a * a;
        }
        public string Info()
        {
            return "Krychle o hraně " + a.ToString() + " Objem: "
                + Objem().ToString() + " Povrch: " +
                Povrch().ToString();
        }
    }

public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonCtverec_Click(object sender, EventArgs e)
        {
            Ctverec c=new Ctverec(Convert.ToDouble(textBoxA.Text));
            textBoxVystup.Text += c.Info() + Environment.NewLine;
        }

        private void buttonKrych_Click(object sender, EventArgs e)
        {
            Krychle k = new Krychle(Convert.ToDouble(textBoxA.Text));
            textBoxVystup.Text += k.Info() + Environment.NewLine;

        }
    }
```