Číslo a název šablony	III/2 Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo didaktického materiálu	EU-OPVK-VT-III/2-ŠR-104
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech I
Téma didaktického materiálu	Ovládací prvky a události
Vyučovací předmět	Seminář z informatiky
Cílová skupina (ročník)	Žáci ve věku 16–17 let
Úroveň žáků	začátečníci
Časový rozsah	1–2 vyučovací hodiny
Klíčová slova	Ovládací prvky, události, tabulátor, knihovny podprogramů
Anotace	Studenti se seznamují s dalšími událostmi, rozlišují syntaktické a sémantické chyby a učí se na ně reagovat, poznávají knihovny podprogramů.
Použité zdroje	DRÓZD, Januš a Rudolf KRYL. *Začínáme s programováním*. 1.vyd. Praha: Grada, 1992, 306 VYSTAVĚL, Radek. *Moderní programování: sbírka úloh k učebnici pro začátečníky*. 2. vyd. Ondřejov: moderníProgramování, 2008, 2 sv. ISBN 978-80-903951-5-2. VYSTAVĚL, Radek. *Moderní programování: učebnice pro začátečníky*. Ondřejov: moderníProgramování s.r.o, 2007, 2 sv. ISBN 978-80-903951-0-7.
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) 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.

104. Ovládací prvky a události

Popisek (Label)

Text, který můžeme umístit kamkoliv do okna programu, nejčastěji se používá k popisu ovládacích prvků. Je to pasivní ovládací prvek, neumí přijmout vstup od uživatele.

Důležité vlastnosti:

Name

Text

TextAlign

Pořadí tabulátoru

Klávesa tabulátoru slouží k pohybu mezi komponentami, které mohou přijímat vstup z klávesnice. (Např. při vyplňován textových polí ve formulářích je to pro uživatele rychlejší než použití myši) Přitom se předává zaměření (**fokus**), v zaměřené komponentě bliká kurzor. Toto pořadí je dáno číselnou vlastností **TabIndex**, systém toto číslo přiřazuje automaticky podle pořadí, ve kterém vkládáme objekty na formulář. Nastavení se dá změnit. (V okně Properties, View/TabOrder nebo i za běhu programu – viz další příklad).

Label sice vstup z klávesnice nepřijímá, ale předává zaměření komponentě s nejbližším vyšším číslem

Pokud chceme komponentě předat zaměření přímo, můžeme to udělat příkazem:

```
textBox1.Focus();
```

Příklad 1

Umístěte na formulář tři textová políčka textBox1,textBox2, textBox3, před ně popisky Label1, Label2, Label3, jejichž text budou pouze číslice 1., 2., 3. a dvě tlačítka – buttonVpred (text:Dopředu) a buttonVzad (text:Dozadu). Při stisknutí tlačítka buttonVzad bude tabulátor předávat zaměření v pořadí 3, 2, 1, 3, 2…po stisknutí buttonVpred se obnoví pořadí normální.

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

```
{
    textBox1.Focus();
    textBox1.TabIndex = 3;
    textBox2.TabIndex = 5;
    textBox3.TabIndex = 7;
}
private void buttonVzad_Click(object sender, EventArgs e)
{
    textBox3.Focus();
    textBox3.TabIndex = 3;
    textBox2.TabIndex = 5;
    textBox1.TabIndex = 7;
}
```

Abychom zabránili tabulátoru „poskakovat“ po tlačítkách, potřebovali bychom znát podmiňovací příkaz.

Všimněte si, že příkaz `textBox2.TabIndex = 5;` jsme mohli vynechat:.

Události

Programy s grafickým uživatelským rozhraním (Windows) jsou řízeny událostmi. Známe již událost klepnutí na tlačítko a budeme používat další – stisknutí klávesy, stisknutí tlačítka myši, ale také impuls, který umí pravidelně vysílat časovač.

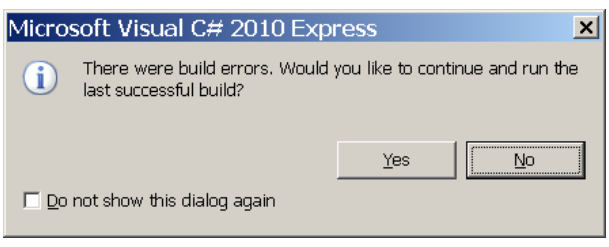
Událost komponenty vybíráme v Editoru vlastností a událostí, vývojové prostředí pak otvírá zdrojový kód Form1.cs a vkládá do něj prázdný obslužný program, jehož kód už zapisuje programátor. Obslužný program začíná hlavičkou: `private void buttonVpred_Click(object sender, EventArgs e)`

Hlavička obsahuje název události (buttonVpred_Click), zpřesnění jejích vlastností (private void) a parametry, pomocí kterých podprogram komunikuje se svým okolím. `(object sender, EventArgs e)`

Za hlavičkou následuje tělo, které se zapisuje mezi otvírací a uzavírací složenou závorku.

Chyba v programu

Pokud se program nespustí a vývojové prostředí hlásí nějaké chyby, patrně vznikla chyba, takže se program nepodařilo sestavit. Většinou je to **chyba syntaktická** – něco špatně zapsaného, případně něco na špatném místě.



Na hlášku vždycky odpovídejte **NO** a snažte se odstranit první chybu v **Error Listu** ve spodní

části okna návrhu programu. (Některé další mohou být umělým důsledkem té první)

Knihovny podprogramů

Příkaz MessageBox.Show, který zobrazuje zprávu, je ve skutečnosti podprogram z knihovny podprogramů, které jsou součástí každé vývojářské platformy. Knihovny jsou dnes převážně soubory s příponou .dll. Aby se knihovna mohla v projektu použít, musí na ni vést odkaz z projektu, toto pro nás zatím řešilo vývojové prostředí – volbou šablony WindowsFormApplication se připojuje několik knihoven. Prohlédnout si je můžeme v Průzkumníku řešení – položka Reference. Seznam také najdeme na začátku program Form.cs.

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;
```

Chceme-li podprogram použít, voláme ho. (Používám nepřesně termín příkaz) Volání obsahuje jméno podprogramu a má-li parametry, uvedou se v závorkách. Pokud je bez parametrů, jsou závorky prázdné. (Vyzkoušejte si podprogram **Close()**, který zavírá okno.) Např. v příkazu MesssageBox.Show("Je parametrem text v závorce"). Všechny příkazy ukončuje středník.

Bližší určování

V příkazu MesssageBox.Show("Je parametrem text v závorce") je MessageBox třídou, která určuje skupinu příbuzných podprogramů a Show potom podprogram z této třídy.

Odstraňování prázdných událostí

Pokud na objekt v Designeru poklepeme omylem, vzniká událost, kterou nebudeme vyplňovat. Rozhodně ji nemažeme (zůstávala by vazba na obslužnou metodu a program by se nesestavil). Nejjednodušší je nevšímat si jí a nechat ji tak. Pokud ji chcete smazat, použijte Editor vlastností a událostí. Zobrazte si tuto událost (Vybrat např. Click) a z její místní nabídky zvolte **Reset**. (V kódu je někdy třeba smazat ručně) Podobně se dělá přejmenování: volba **Refactor/Rename**. (Zde vybíráme název události ve zdrojovém kódu.

Událost MouseHover

vzniká, když se myš zastaví nad objektem. Můžeme ji vyzkoušet v následujícím příkladu

Příklad 2

Připravíme program s jediným tlačítkem ButtonJá a textem Já a když se nad ním zastaví myš, objeví se zpráva “Jsem nad tlačítkem ButtonJá”. Událost MouseHover vyberte v okně vlastností a událostí tlačítka, po zobrazení můžete okno zavřít příkazem Close().

```
private void buttonJa_MouseHover(object sender, EventArgs e)
        {
            MessageBox.Show("Jsem nad tlačítkem ButtonJá");
            Close();
        }
```

Postupné zpracování programu

Příklad 3

Připravte si program se třemi textovými poli a jedním tlačítkem. Tlačítko bude buttonAkce s textem Kopíruj, první pole pojmenujeme textBoxtZdroj, druhé textBoxKopie a třetí textBoxZalohaKopie. Při klepnutí na tlačítko budeme chtít text ze zdroje zkopírovat do kopie a co bylo v kopii do záložní kopie.

Mohlo by to tedy fungovat takhle:

```
textBoxKopie.Text = textBoxZdroj.Text;
textBoxZaloha.Text = textBoxKopie.Text;
```

Jenže když to vyzkoušíme, text se neposouvá, jak by měl. Zdrojový text se dostane do kopie, ale ta se tím přepíše a v záloze se pak objeví opět zdroj.

Chybám tohoto typu říkáme **sémantické,** vývojové prostředí je nedokáže odhalit a obvykle se projevují tak, že program dělá něco jiného, než chceme. Nicméně pamatujme, že vždycky dělá pouze a přesně to, co jsme naprogramovali.

Pořadí příkazů je třeba vyměnit:

```
private void buttonAkce_Click(object sender, EventArgs e)
        {
            textBoxZaloha.Text = textBoxKopie.Text;
            textBoxKopie.Text = textBoxZdroj.Text;
        }
```

Prvním příkazem se uschová Kopie do zálohy a druhým se do ní přesune zdroj. Pokud bychom chtěli zdroj vymazat a nastavit do něj kurzor k dalšímu pokračování, doplníme ještě příkazy:

```
textBoxZdroj.Text = null;
textBoxZdroj.Focus();
```

Null je jakási univerzální nula C#, podobně by zde fungoval prázdný řetězec – dvě uvozovky vedle sebe.

Důležité:

Popisek Label – užíváme k popisu komponent, především vlastnost **Text**.

TabIndex – vlastnost všech komponent, která určuje pořadí tabulátoru

Chyba při sestavení programu – na hlášku odpovídáme No a snažíme se odstranit první chybu v ErrorListu

Close() – příkaz zavře okno

MouseHover – událost komponenty, která vzniká, když se nad ní zastaví myš.

Příkazy se za sebou vykonávají v tom pořadí, jak je zapíšeme. (Program vždycky dělá to, co do něj zapíšeme, nikoliv to, co bychom si přáli)

Syntaktické chyby – chyby v zápisu kódu

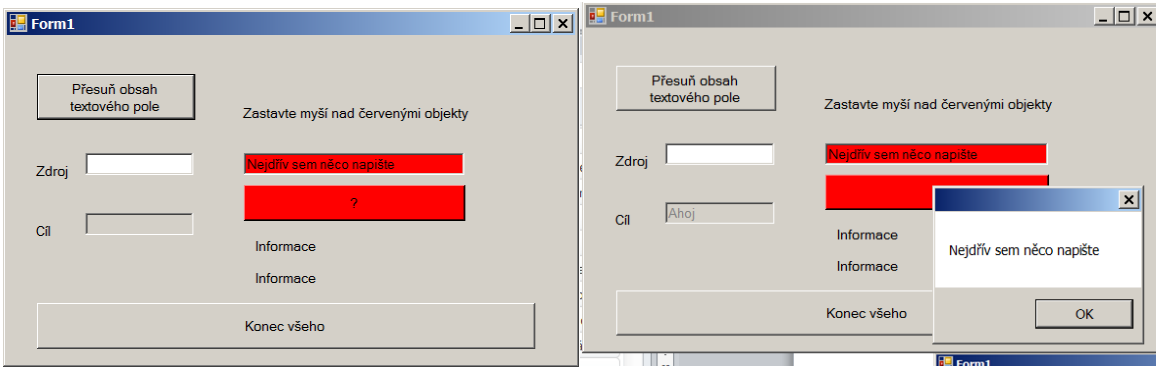
Sémantické chyby – logické chyby v programu

null – univerzální nulový prvek

Pracovní list

Cvičení

1. Vyzkoušejte si program Cvičení 4.exe a zkuste podle něj udělat svůj.



2. Vyzkoušejte si v dalším projektu nějaké další vlastní nápady..

Řešení

```
namespace Cvičení_4

{

public partial class Form1 : Form

{

public Form1()

{
```

```
InitializeComponent();
}
private void buttonPresun_Click(object sender, EventArgs e)
{
textBoxCil.Text = textBoxZdroj.Text;
textBoxZdroj.Text = "";
textBoxZdroj.Focus();
}
private void buttonKonec_Click(object sender, EventArgs e)
{
MessageBox.Show("Program končí a vy už s tím nic nenaděláte");
Close();
}
private void textBox1_MouseHover(object sender, EventArgs e)
{
MessageBox.Show(textBoxRed.Text);
}
private void buttonRed_MouseHover(object sender, EventArgs e)
{
buttonRed.BackColor = Color.Aquamarine;
}
private void buttonRed_Click(object sender, EventArgs e)
{
labelInfo2.Text = "Když se myš zastaví nad červeným labelem, ";
labelInfo3.Text = "objeví se zpráva a nestihne se psaní";
}
}
}
```

2. Individuální řešení