

Grafičko korisničko sučelje

5/13

Grafičko korisničko sučelje

☐ Graphical User Interface (GUI)

☐ *WindowsForms* i *WebForms*

- *WindowsForms* - namijenjene izradi klijentskih aplikacija (“debeli klijent”)
- *WebForms* - namijenjene izradi Web stranica (“tanki klijent”)

☐ `System.Windows.Forms` prostor imena za razrede Windows GUI

- Skup baznih razreda za podršku prozorima i kontrolama sučelja
- Svaka forma nasljeđuje `System.Windows.Forms`

☐ Forma (*form*)

- strukturirani prozor ili okvir koji sadrži prezentacijske elemente za prikaz i unos podataka

☐ Forma i prozor (*window*)

- Svaka forma je prozor, ali se uobičajeno podrazumijeva da je forma dijalog na kojem se nalazi određeni broj kontrola

Hijerarhija razreda

System.Object

System.MarshalByRefObject

System.ComponentModel.Component

System.Windows.Forms.Control

System.Windows.Forms.ScrollableControl

System.Windows.Forms.ContainerControl

System.Windows.Forms.Form

☐ MarshalByRefObject

- automatizira prenošenje objekata putem proxy-a preko različitih aplikacijskih domena (.NET Remoting)

☐ Component

- razred koji implementira sučelje **IComponent**
- omogućava dijeljenje objekata između aplikacija
- nema vidljivih dijelova

☐ Control

- osnovni bazni razred za kontrole s vizualnim sučeljem (npr. gumb)

☐ ScrollableControl

- osnovni razred za kontrole koje podržavaju auto-scrolling

☐ ContainerControl

- funkcionalnost potrebna da bi kontrola sadržavala druge kontrole

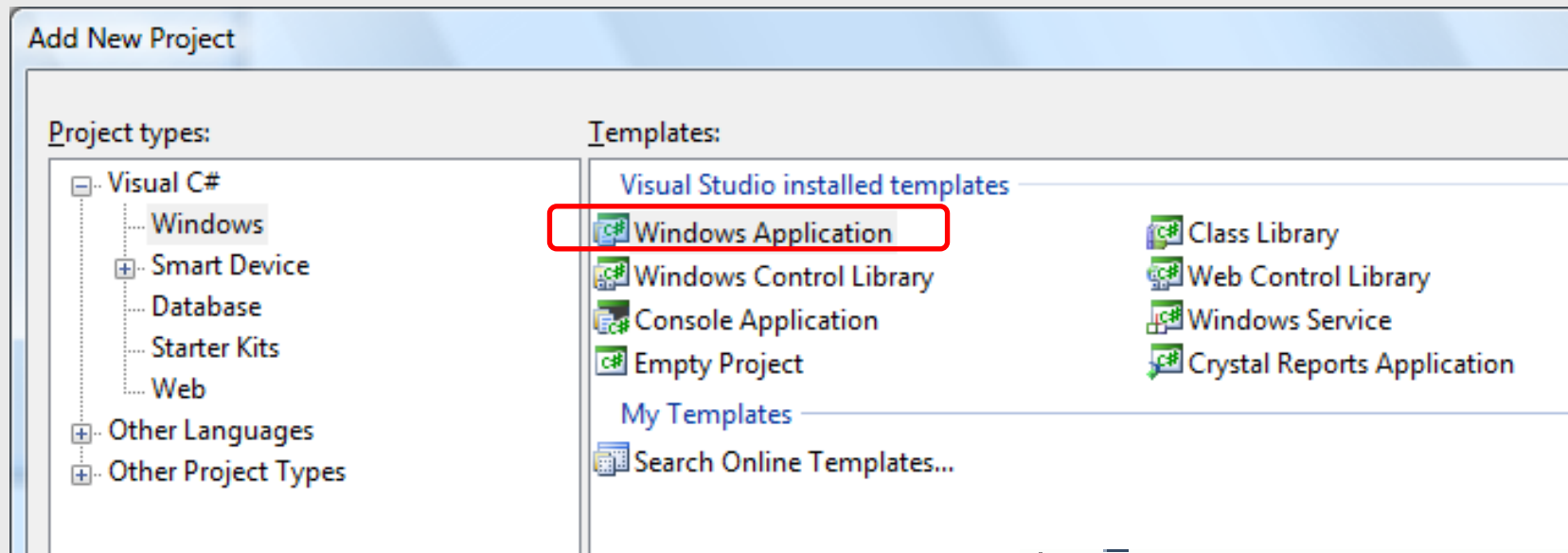
☐ Form

- razred koji predstavlja (praznu) formu ili dijalog kutiju i može sadržavati druge kontrole
- iz ovog razreda se najčešće izvodi razred u kojem se zatim implementiraju potrebne specifičnosti

Automatsko generiranje forme

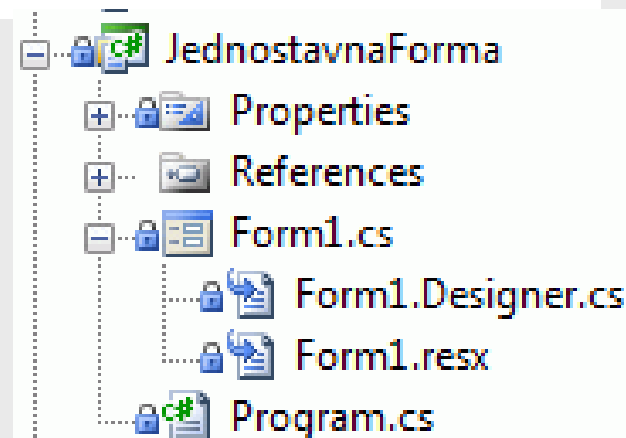
□ Primjer: GUI\JednostavnaForma

- File → New → Project – u odjeljku Windows odabrati Windows Application



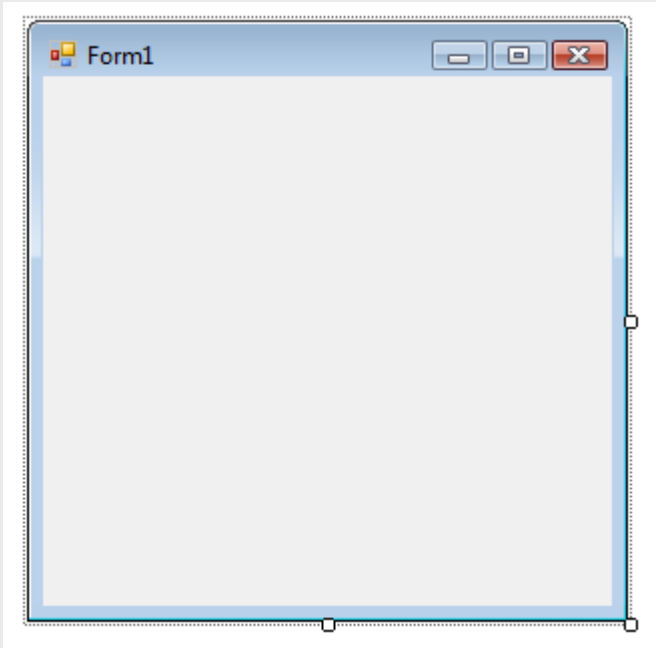
□ Stvara se projekt s nekoliko dokumenata:

- Form1.cs (programski kôd forme)
 - Form1.Designer.cs (definicija izgleda)
 - Form1.resx (resursi)
- Program.cs (glavni program)



Automatsko generiranje forme

❑ Form1.cs[Design] i Form1.cs



partial class – ostali
dijelovi razreda nalaze se u
drugim datotekama (npr.
Form1.designer.cs na
sljedećoj foliji)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

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

Automatsko generiranje forme

❑ Form1.Designer.cs

- `components` – instanca `System.ComponentModel.Container` razreda – komponenta, “spremnik” (eng. *container*) svih kontrola na formi
- `Dispose()` - “čišćenje” tzv. *unmanaged* resursa (koje ne čisti *Garbage Collector*) korištenih komponenti # nama do daljnjega neće trebati

```
partial class Form1
{
    private System.ComponentModel.IContainer components = null;
    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }
}
```

Automatsko generiranje forme

❑ Inicijalizacija: `InitializeComponent()`

- svako postavljanje svojstva forme ili postavljanje kontrole na formu u razvojnoj okolini generira kôd unutar `InitializeComponent()` funkcije
- nestručne ručne izmjene tijela `InitializeComponent` mogu uzrokovati da *Visual Designer* više ne bude u stanju sinkronizirati kôd i izgled forme

```
partial class Form1
{
    ...
    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.Text = "Form1";
    }
}
```

Automatsko generiranje forme

❑ Program.cs

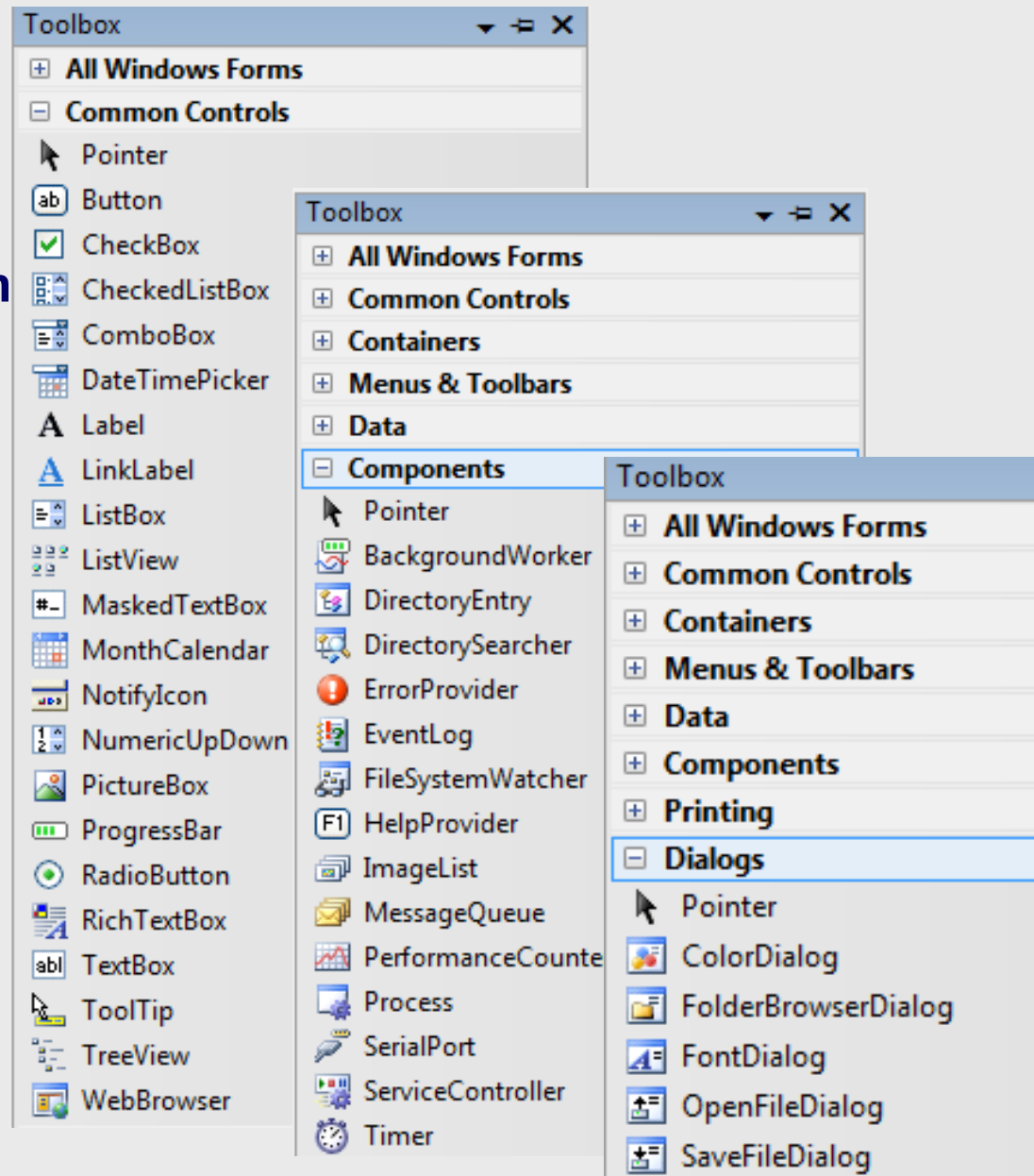
- Aplikacija kreće pokretanjem statičke funkcije *Main()*
- U *Main()* instanciramo i pokrenemo formu
- Grafičko sučelje dalje je vođeno događajima koji se zbivaju nad formom i objektima forme

❑ Primjer: izvođenje programa naredbu po naredbu

```
static class Program
{
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```


Elementi grafičkog sučelja

- ❑ **Kontrole, komponente, dijalozi**
- ❑ **Dodajemo ih dovlačenjem (*drag-drop*) iz kutije s alatima (*Toolbox*)**
- ❑ **Dizajn forme i kontrola**
 - označavanje jedne ili više kontrola mišem, a zatim
 - premještanje, razvlačenje ili smanjivanje
 - Izbornik *Format*



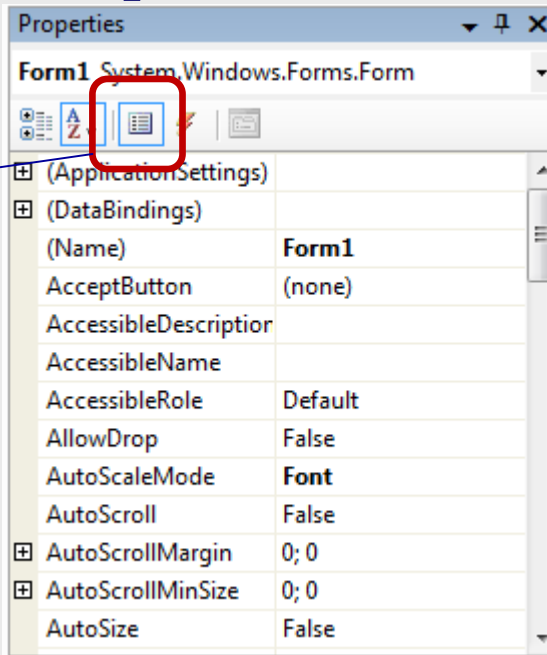
Svojstva i događaji

❑ Svojstva objekata (forme i ostalih kontrola)

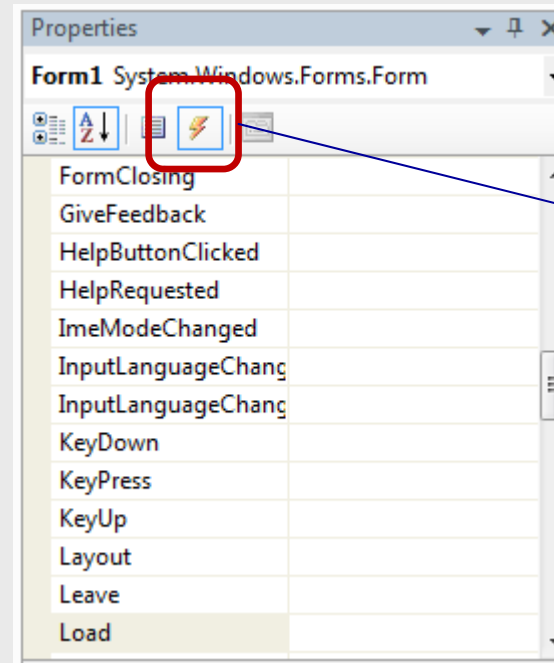
- inicijalno se postavljaju prilikom dizajna (u *Property Window*), označavanjem objekta i izbornikom *View – Properties Window* ili desnim klikom + *Properties*
- neka svojstva mogu biti i dinamički promijenjena, pri izvođenju programa

❑ Za svojstva početno postavljena u dizajnu generira se kod u `InitializeComponent()`

Svojstva



Događaji



Vrste (razredi) kontrola

- ☐ **Button** (gumb) – osnovna namjena je pokretanje akcija
- ☐ **Label** – prikazuje tekst, bez mogućnosti unosa
- ☐ **TextBox** – prikazuje tekst koji se može unositi
- ☐ **CheckBox** – predstavlja varijablu s dva stanja (true/false)
- ☐ **RadioButton** – predstavlja grupu da/ne izbora, stavlja ih se više u isti spremnik, a samo jedan gumb bude potvrđan

- ☐ **ListBox** i **CheckedListBox** – lista stavki, odabire se jedna ili više
- ☐ **ComboBox** – padajuća lista u kojoj se odabire samo jedna stavka

- ☐ **LinkLabel** – prikazuje jedan ili više hiperveza (hyperlink)
- ☐ **PictureBox** – prikazuje sliku

- ☐ **DataGridView, ListView, TreeView** – kontrole za prikaz kolekcije podataka

- ☐ **GroupBox i Panel** – spremnici kontrola

- ☐ ...

Standardna svojstva i postupci kontrola

□ Standardna svojstva i postupci

Svojstva	
<code>BackColor</code>	Boja pozadine
<code>BackgroundImage</code>	Slika pozadine
Enabled	Omogućena aktivnost (događaji)
<code>Focused</code>	Kontrola ima fokus (trenutno se koristi)
<code>Font</code>	Font svojstva <code>Text</code>
<code>ForeColor</code>	Boja prednjeg plana. Uobičajeno se odnosi na svojstvo <code>Text</code>
<code>TabIndex</code>	Redni broj kontrole, određuje redoslijed kojim ona dolazi u fokus (npr. tipkom <code>Tab</code>)
<code>TabStop</code>	Oznaka da kontrola dolazi u fokus postavljenim redoslijedom
Text	Tekst koji se prikazuje na kontroli
<code>TextAlign</code>	Poravnanje, horizontalno (left, center, right), vertikalno (top, middle or bottom).
Visible	Oznaka da je kontrola vidljiva
Postupci	
<code>Focus</code>	Prijenos fokusa na kontrolu
<code>Hide</code>	Skrivanje kontrole (postavlja Visible na false).
<code>Show</code>	Pojavljivanje kontrole (postavlja Visible na true).

Razred *Form*

□ Svojstva

Text	naslov forme
BackColor, ForeColor	boja pozadine i prednjeg plana
Font	koji font se koristi prilikom rada na formi
WindowState	stanja (Normal, Maximized, Minimized)
Size	dimenzije
Location, DesktopLocation	postavljanje pozicije forme na ekranu
MaximizeBox, MinimizeBox ControlBox	elementi upravljanja prozorom
FormBorderStyle	Fixed3D, FixedDialog, FixedSingle, ...
TopMost	da li 'e forma uvijek biti vidljiva (na vrhu)

□ Događaji

Activated	dogaća se kad forma dobije fokus
Deactivated	poziva se kad forma gubi fokus
Closing	poziva se prilikom zatvaranja forme (prije samog zatvaranja)
Closed	poziva se prilikom zatvaranja forme (nakon samog zatvaranja)
Load	poziva se prilikom prvog učitavanja forme

Svojstva forme

❑ Postupci načina prikaza

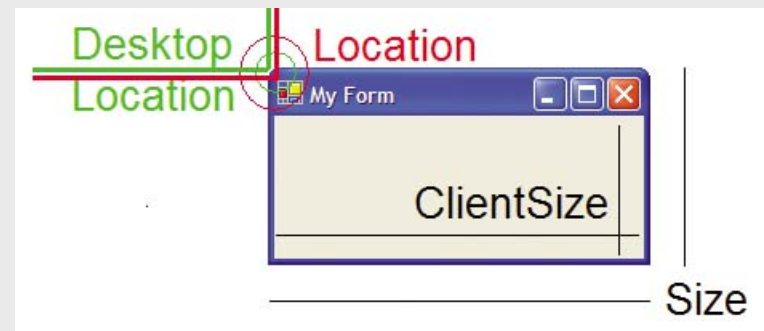
- `Show` – nemodalni prikaz
 - moguće je kliknuti na neku od više otvorenih formi
- `ShowDialog` – modalni prikaz
 - forma na vrhu ne dozvoljava da se aktiviraju donje
- `Hide` – sakriva formu

❑ Svojstva prikaza

- `Size` – veličina prozora
- `Location` – položaj lijevog-gornjeg ruba prozora u odnosu na spremnik
- `StartPosition` – početni položaj forme pri izvođenju
- `DesktopLocation` – dinamička promjena lokacije (programski)

❑ Događaji

- `LocationChanged`
- `SizeChanged`



Događaji

❑ Događaj - mehanizam razreda za dojavu zbivanja nad objektom

- primjeri događaja: klik mišem na gumb, unos teksta u kućicu formulara učitavanje/zatvaranje forme, ...
- povećava se modularnost programa
- osim na GUI, koriste se i šire

❑ Rukovatelj događajem (*Event handler*)

- Postupak koji obrađuje događaje
- Uobičajenog naziva *ControlName_EventName*
- Argument su dvije reference, objekt događaja i podaci događaja

❑ Primjer: GUIJednostavnaForma

- kako debugirati izvođenje događaja ?
- što kada se promijeni naziv objekta (npr. *button1* postane *gumb*) ?

```
private void button1_Click(object sender, EventArgs e)
{
    this.Text = "Pozdrav!";
    label1.Text = System.DateTime.Now.ToString();
}
```

Događaji i delegati događaja

❑ Delegati događaja pridruženi objektu

- delegat – referenca na postupak (isto što i pokazivač na funkciju u C/C++)
- postupci obrade događaja objekta registriraju se dodavanjem na listu poziva delegate objekta

```
this.button1.Click += new System.EventHandler(this.button1_Click);
```

Događaj

Delegat

Event handler

❑ **Event multicasting**

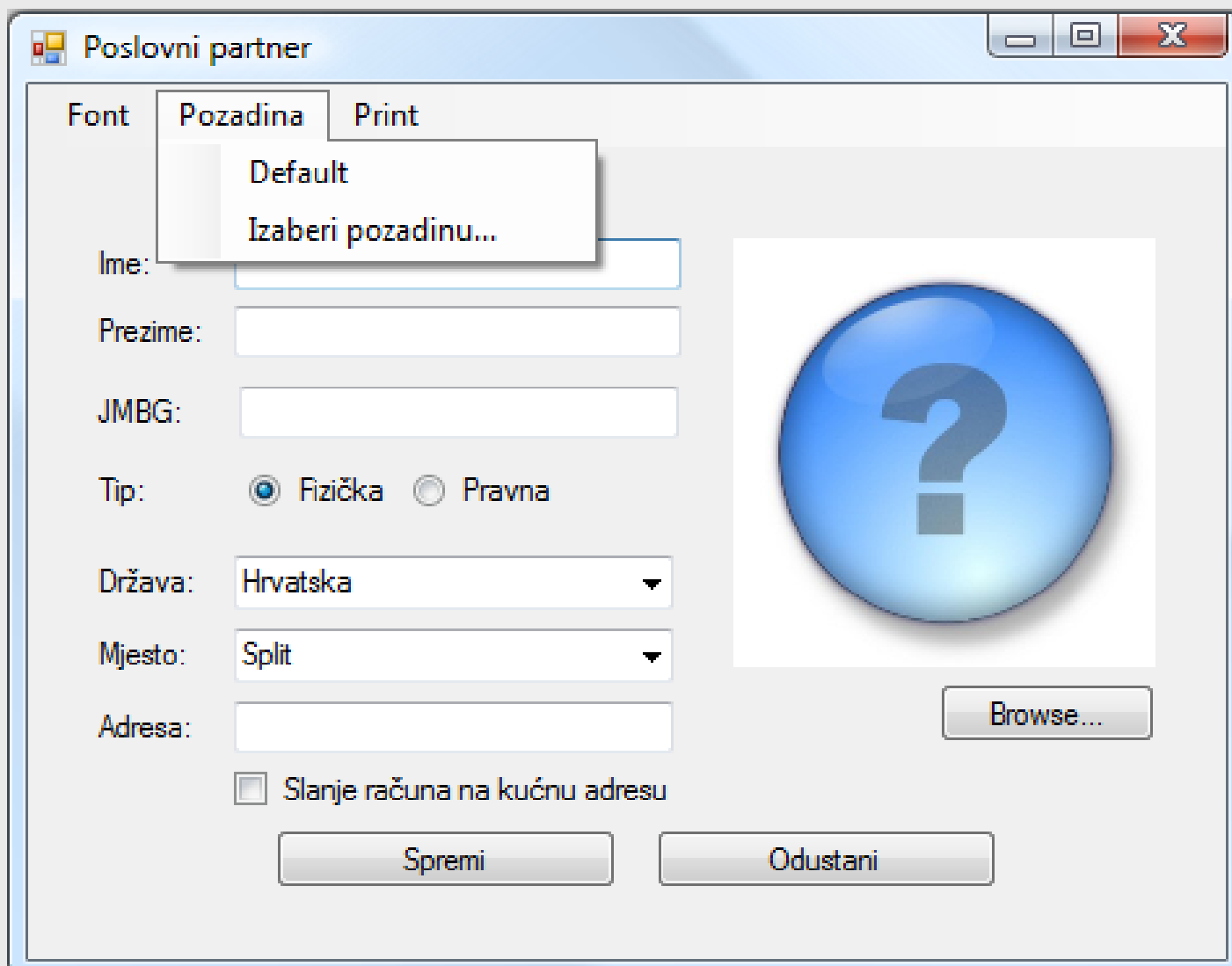
- može postojati više postupaka za obradu istog događaja
- za svaki rukovatelj (*handler*) kreira se novi delegat



❑ **Primjer: kako promijeniti naziv rukovatelja događaja ?**

Izrada zaslonske maske za unos podataka

□ Primjer: GUNPartner



Poslovni partner

Font Pozadina Print

Default
Izaberi pozadinu...

Ime:

Prezime:

JMBG:

Tip: ☒ Fizička ☐ Pravna

Država: Hrvatska ▼

Mjesto: Split ▼

Adresa:

☐ Slanje računa na kućnu adresu

Browse...

Spremi Odustani

Button, Label i TextBox

❑ Button

Svojstva	
FlatStyle	Plošni izgled
Image	Slika – ako nije postavljena, gumb prikazuje vrijednost svojstva Text
ImageList	Lista slika, od kojih jedna može u nekom trenutku biti prikazana
ImageIndex	Indeks trenutno prikazane slike
Događaji	
Click	Klik mišem ili <i>Enter</i> kada je kontrola u fokusu

❑ Label i TextBox

Svojstva	
AcceptsReturn	<i>True</i> – <i>Enter</i> umeće znak za novi redak
Multiline	prikaz teksta koji ima više redaka (uobičajeno se postavlja i ScrollBars)
PasswordChar	Znak koji će se pojavljivati pri unosu umjesto teksta koji se unosi. Ako se izostavi prikazuje se tekst koji se unosi
ReadOnly	<i>True</i> – pojavljuje se siva pozadina i ne dozvoljava unos
ScrollBars	Vrsta kliznih traka za <i>Multiline</i> : <i>none</i> , <i>horizontal</i> , <i>vertical</i> , <i>both</i> .
Text	Tekst
Događaji	
TextChanged	Primjena teksta (za svaki znak)

RadioButton, GroupBox i Panel

❑ **RadioButton**

- uobičajeno predstavlja grupu da/ne izbora
- kada se postavi u spremnik može se označiti samo jedna instanca kontrole

Svojstva	
Appearance	<i>Normal</i> ili <i>Button</i>
Checked	<i>true</i> ako je kontrola označena, inače <i>false</i>
AutoCheck	<i>true</i> : kontrola postavlja oznaku sukladno vrijednosti <i>Checked</i>
Događaji	
AppearanceChanged	promjena svojstva <i>Appearance</i>
CheckedChanged	promjena stanja <i>Checked</i>

❑ **GroupBox i Panel – spremnici kontrola s labelom**

- `Controls` – lista sadržanih kontrola
- `BorderStyle` – obrub (*Fixed3D*, *FixedSingle*, *None*)
- `AutoScroll` – automatska pojava kliznika kad se ne vide kontrole (*Panel*)
- `Text` – labela na vrhu *GroupBox* kontrole (samo *GroupBox*)

❑ **Primjer, premještanje panela u dizajnu**

ListBox, CheckedListBox, ComboBox

- ❑ **ListBox i CheckedListBox** – lista stavki, odabire se jedna ili više
- ❑ **ComboBox** – padajuća lista u kojoj se odabire samo jedna stavka

Svojstva	
Items	<i>ListBoxItems.ObjectCollection</i> objekt s listom svih stavki
MultiColumn	Prikaz stavki u više stupaca (samo ListBox)
SelectedIndex	indeks selektirane stavke ili -1 kada nijedna nije odabrana
SelectedItem	Object referenca na selektiranu stavku
Sorted	True: stavke su sortirane
Metode	
Add, Clear, Insert, Remove, ...	Rukovanje stavkama (sjetimo se kolekcija)
ClearSelected	briše sve selektirane stavke
FindString,	nalazi prvu stavku u listi koja počinje sa zadanim stringom
FindStringExact	nalazi prvu stavku u listi koja točno odgovara zadanom stringu
Sort	sortira stavke u listi
Događaji	
SelectedIndexChanged	podigne se kad se promijeni SelectedIndex

Padajuće liste

❑ Padajuća lista (*ComboBox*): Mjesto i Država

- Kada odaberemo državu želimo ponuđene samo gradove te države

```
listaDrzava[0] = "Hrvatska";  
listaDrzava[1] = "Njemačka";  
...  
listaGradova[0] = new string[] { "Split", "Zagreb", "Dubrovnik" };  
listaGradova[1] = new string[] { "Koeln", "Berlin", "Frankfurt"};  
...  
comboBoxDrzava.DataSource = listaDrzava; // ekvivalent Items u dizajnu  
comboBoxDrzava.SelectedIndex = 0;  
comboBoxMjesto.DataSource =  
    listaGradova[comboBoxDrzava.SelectedIndex];
```

```
private void comboBoxDrzava_SelectedIndexChanged(  
    object sender, EventArgs e)  
{  
    comboBoxMjesto.DataSource =  
        listaGradova[comboBoxDrzava.SelectedIndex];  
}
```

PictureBox

□ Svojstva

- **Image** – slika
- **BorderStyle** – obrub (Fixed3D, FixedSingle, None)
- **SizeMode** – veličina i položaj bitmape (enum PictureBoxSizeMode)
 - **Normal (default)** – slika u gornjem lijevom kutu kontrole
 - vidi se samo dio slike ako je slika veća od kontrole
 - **CenterImage** – slika u sredini kontrole
 - vidi se samo dio slike ako je slika veća od kontrole
 - **StretchImage** – prilagodba veličine slike prema veličini kontrole
 - **AutoSize** – prilagodba veličine kontrole prema veličini slike
 - **Zoom** – promjena veličine kontrole uz zadržavanje omjera visina:širina

Izbornici

❑ **MainMenu** – korijen sustava izbornika

❑ **MenuItem** – stavka izbornika

Svojstva	
Checked	da li se kraj stavke nalazi kvačica
Text	tekst stavke, vrijednost "-" prikazuje separator
Shortcut	definira tipkovničku kraticu za pozivanje stavke
Događaji	
Click	podigne se kad se klikne na stavku ili utipka tipkovničku kraticu
Metode	
GetMainMenu	vraća <i>MainMenu</i> objekt kojeg je dio ta stavka
CloneMenu	kreira kopiju izbornika (podrazumijeva dinamički rad s izbornicima)
MergeMenu	spaja dva izbornika u jedan
OnClick	podigne <i>Click</i> događaj
PerformClick	generira <i>Click</i> događaj nad stavkom (simulira akciju korisnika)

❑ **ToolBar** – traka s alatima

■ `ToolBarButton` kontrola

❑ **Contextbar** – izbornik zavisan o kontekstu

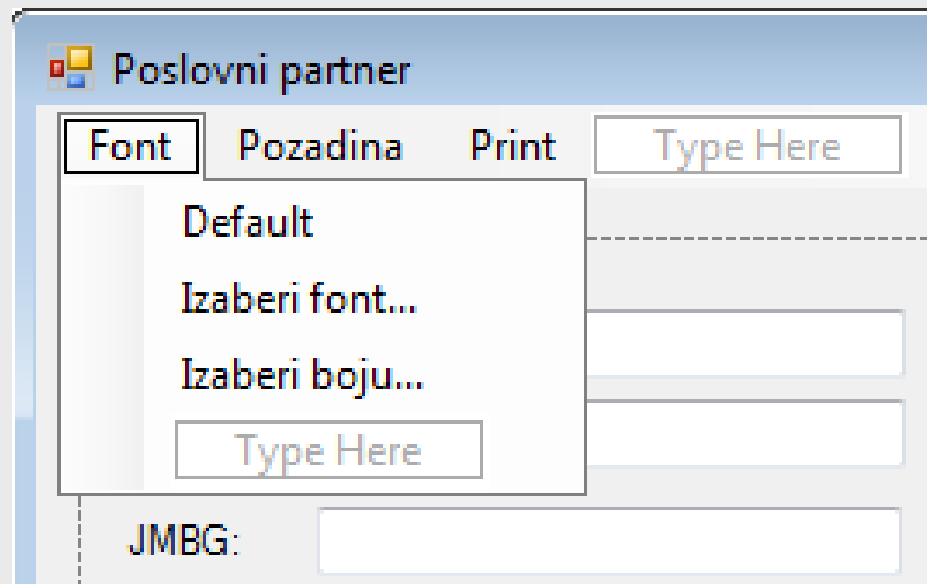
Primjer izbornika

❑ Primjer: GUI\Partner

- izgled izbornika u dizajnu
- korijen sustava izbornika (MenuStrip) i stavke (ToolStripMenuItem)
- desni klik + Edit items

Događaji izbornika

- Primjer, klik mišem na stavku izaberi font...



```
System.Windows.Forms.MenuStrip menuStrip;
...
this.MainMenuStrip = this.menuStrip;
...
private void izaberiFontToolStripMenuItem_Click(
    object sender, EventArgs e)
{
    //...
}
```


Dijalozi

- ❑ **Dijalog** – posebna, preddefinirana vrsta forme za jednostavnu interakciju i odabir vrijednosti korisnika u specifičnim situacijama

- ❑ **Vrste dijaloga, npr.**
 - promjena tipa pisma (*FontDialog*), boje slova,
 - promjena boje pozadine (*ColorDialog*)
 - tisak (*PrintDialog* i *PrintDocument*)
 - dijalog za obradu datoteka (*FileOpenDialog*)

- ❑ **Svi ugrađeni dijalozi imaju postupke**
 - `ShowDialog` – prikaz dijaloga
 - `Dispose` – oslobađa dijalogom zauzete resurse
 - `ToString` – vrijednost objekta
 - za dijaloge ispisa koristi se `using System.Drawing.Printing;`

Predefinirani dijalozi

❑ Predefinirani dijalozi i najvažnija svojstva (pogledati Help)

- OpenFileDialog - CheckFileExists, FileName, Filter, InitialDirectory, ShowReadOnly
- SaveFileDialog - CheckFileExists, FileName, Filter, InitialDirectory, OverwritePrompt
- ColorDialog - Color, AnyColor, FullOpen
- FontDialog - Font, FontMustExist
- PageSetupDialog - Document (*PrintDocument*), PageSettings
- PrintDialog - Document (*PrintDocument*), PrinterSettings
- PrintPreviewDialog - Document (razred *PrintDocument*)

❑ Nalaze se u *System.Windows.Forms*

Primjeri dijaloga

❑ OpenFileDialog – odabir slike

```
if (openFileDialogSlika.ShowDialog() != DialogResult.Cancel) {  
    string NazivSlike = openFileDialogSlika.FileName;  
}
```

❑ FontDialog – odabir tipa pisma

- `this.Font = fontDialog.Font;`

❑ PrintDialog – odabir opcija za zapisivanje

- `(System.Drawing.Printing.)PrintDocument`
 - dokument za zapisivanje
- `System.Drawing.Printing.PrintPageEventArgs.Graphics`
 - Dohvaća grafiku za iscrtavanje stranice (dokumenta) za zapisivanje

```
printDocument.PrinterSettings = printDialog.PrinterSettings;  
printDocument.Print();  
...  
e.Graphics.DrawString(tekstZaPrintanje, this.Font, Brushes.Black,  
    new PointF(180, 50));
```

Poruke i resursi

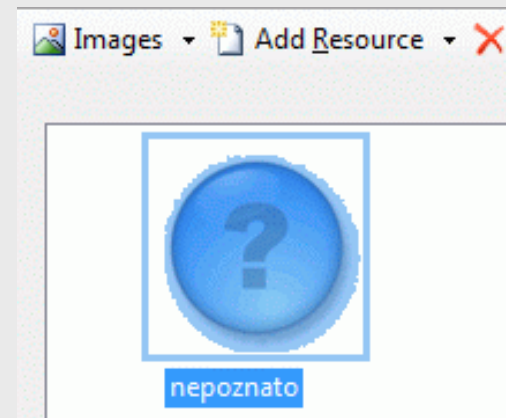
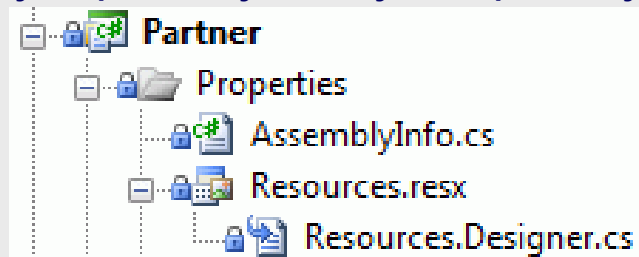
❑ Razred MessageBox

- Postupak **show** – prikaz poruke

```
private void buttonSpremi_Click(object sender, EventArgs e)
{
    MessageBox.Show("Spremljeno!");
}
```

❑ Resources (više o resursima naknadno)

- spremište stringova, datoteka, slika, ...
- *Form1.resx* – spremište resursa forme *Form1*
- Properties projekta \ *Resources.resx*
 - spremište koje upotrebljava cijela aplikacija



- Dohvat objekata iz spremišta *Resources.resx* :
[nazivImenika].Properties.Resources.nazivPostupka

```
pictureBoxSlika.Image = Properties.Resources.nepoznato;
```

MessageBox

❑ Javni razred MessageBox

■ Postupak Show – prikaz poruke, više oblika, najčešći su:

- `DialogResult Show(string text);`
- `DialogResult Show(string text, string caption);`
- `DialogResult Show(string, string, MessageBoxButtons);`
- `DialogResult Show(string, string, MessageBoxButtons, MessageBoxIcon);`





❑ MessageBoxButton i MessageBoxIcon

- `MessageBoxButtons.OK`
- `MessageBoxButtons.OKCancel`
- `MessageBoxButtons.YesNo`
- `MessageBoxButtons.YesNoCancel`
- `MessageBoxButtons.RetryCancel`
- `MessageBoxButtons.AbortRetryIgnore`

❑ enum DialogResult

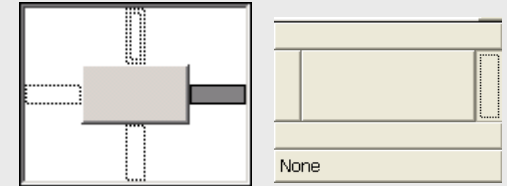
- `Abort, Cancel, Ignore, No, None, OK, Retry, Yes`

❑ Primjer: buttonSpremi_Click

MessageBox Icons	Icon
<code>MessageBoxIcon.Exclamation</code>	
<code>MessageBoxIcon.Information</code>	
<code>MessageBoxIcon.Question</code>	
<code>MessageBoxIcon.Error</code>	

Svojstva za dinamičko razmještanje kontrola

❑ Veličina, položaj i razmještaj u pogonu



❑ Zadatak za vježbu

- Proučiti svojstva za dinamičko razmještanje kontrola (`Anchor`, `Dock`, `DockPadding`, `Location`, `Size`, `MinimumSize`, `MaximumSize`).

Anchor	sidrenje rubova kontrole prema rubu spremnika: <code>Top</code> , <code>Left</code> , <code>Bottom</code> , <code>Right</code> obavlja se dinamički kada se mijenja veličina spremnika, po mogućnosti kontrola mijenja veličinu
Dock	Automatsko prislanjanje kontrole na rub spremnika: <code>Top</code> , <code>Left</code> , <code>Bottom</code> , <code>Right</code> , <code>Fill (Center)</code> , <code>None</code>
DockPadding	Razmak od ruba (samo za spremnike), standardna vrijednost = 0.
Location	Položaj gornjeg-lijevog kuta kontrole, relativno u odnosu na spremnik.
Size	Veličina: Size struktura sa svojstvima Height i Width .
MinimumSize	Minimalna veličina forme
MaximumSize	Maksimalna veličina forme

Validacija unosa podataka

❑ Događaji kontrole

- `Validating (object sender, CancelEventArgs e)` – okida provjeru ispravnosti podataka u trenutku kad kontrola treba biti napuštena
 - postavljanje `e.Cancel = false` omogućuje napuštanje kontrole
 - postavljanje `e.Cancel = true` blokira napuštanje kontrole u slučaju pogrešnog unosa
 - problem: ako postoje neispravni podaci, nije moguće poduzeti nikakvu akciju sve dok se pogreška ne popravi pa čak ni zatvoriti formu
 - rješenje: u `(Form1_FormClosing` postaviti `e.Cancel = false`
- `Validated` – nakon što je provjera obavljena

❑ Svojstva kontrole

- `CausesValidation` – `false`: ne aktivira `Validating` i `Validated`

❑ Postupak forme

- `this.ValidateChildren()` – pokreće niz poziva postupaka `Validating` i `Validated` za kontrole koje forma sadrži

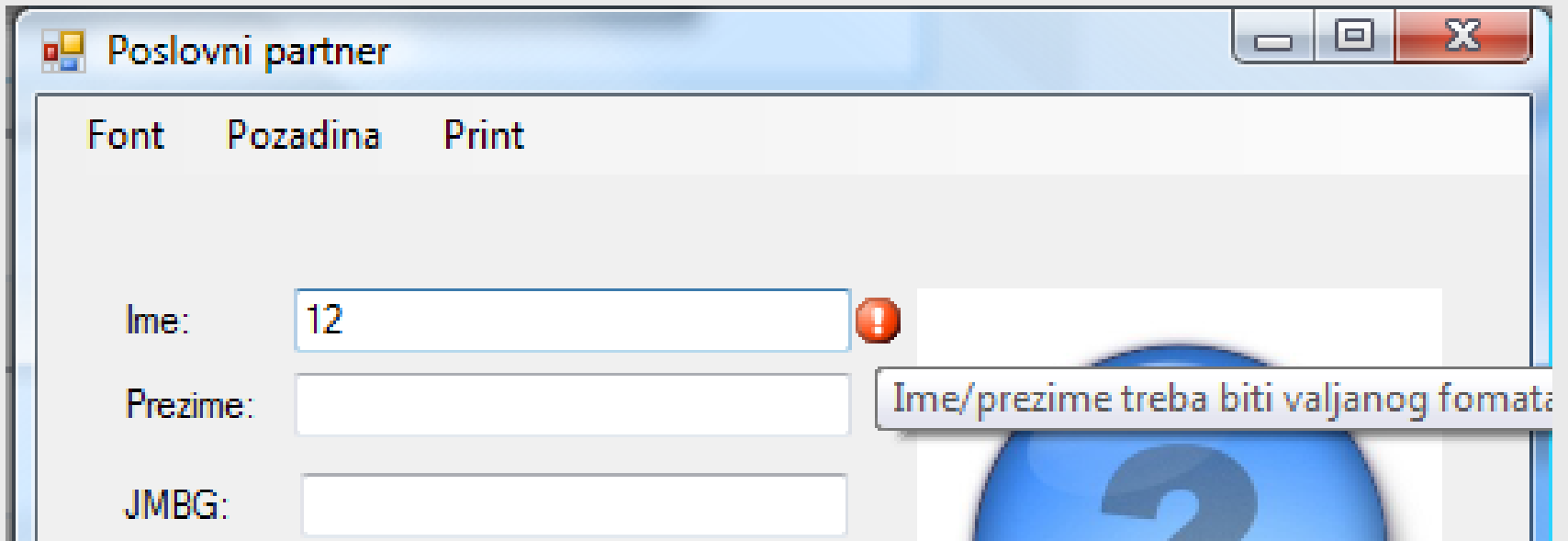
❑ Komponenta `ErrorProvider` – poruka o pogrešnom unosu

- ikona uskličnika se prikazuje dok unos nije valjan
- `SetError(Control, string)` – postavlja poruku o pogrešci uz kontrolu

Primjer validacije pri unosu u zaslonsku masku

❑ Primjer: GUI\Partner

- želimo upozoriti na unos neispravnih podataka
 - Ime i prezime moraju biti uneseni i ne smiju sadržavati brojeve
 - JMBG mora biti niz brojeva duljine 13 znakova
 - ...



ErrorProvider

❑ **ErrorProvider** – prikaz informacije o pogrešci

❑ **Svojstva**

- `BlinkRate` – učestalost treptanja
- `BlinkStyle` – `BlinkIfDifferentError`, `AlwaysBlink`, `NeverBlink`

❑ **Postupci**

- `SetError(Control, string)` – kontrola i objašnjenje

Primjer postupka za validaciju kontrole

❑ Postupak ValidacijaNaziv

- provjera valjanosti formata za ime/prezime
- postavljanje poruke za `errorProvider`

```
private bool ValidacijaNaziv(TextBox textBox) {  
    if (textBox.Text == "") {  
        errorProvider.SetError(textBox, "Unesite ime/prezime.");  
        return false;  
    }  
    else if (System.Text.RegularExpressions.Regex.IsMatch(  
        textBox.Text, "[0-9]")) {  
        errorProvider.SetError(textBox, "Ime/Prezime treba biti...");  
        return false;  
    }  
    else { // OK  
        errorProvider.SetError(textBox, "");  
        return true;  
    }  
}
```

Validacija kontrole i validacija forme

❑ Implementacija događaja *Validating* na kontroli `textBoxIme`

- `e.Cancel` omogućava/onemogućava napuštanje kontrole

```
private void textBoxIme_Validating(object sender, CancelEventArgs e)
{
    if (ValidacijaNaziv((TextBox)sender)) {
        e.Cancel = false; // validacija ok, dozvoliti napuštanje kontrole
    }
    else {
        e.Cancel = true; // validacija neuspješna, forsirati popravak
    }
}
```

❑ Klikom na gumb *Spremi* pozivamo postupak za validaciju

- ako su svi podaci ispravni, `this.ValidateChildren()` vraća *true*

```
if (this.ValidateChildren()) {
    MessageBox.Show("Spremljeno!");
    OcistiFormu();
}
```

ToolTip

❑ **ToolTip – podsjetnik kontrole kojoj je ToolTip pridružen**

■ Svojstva

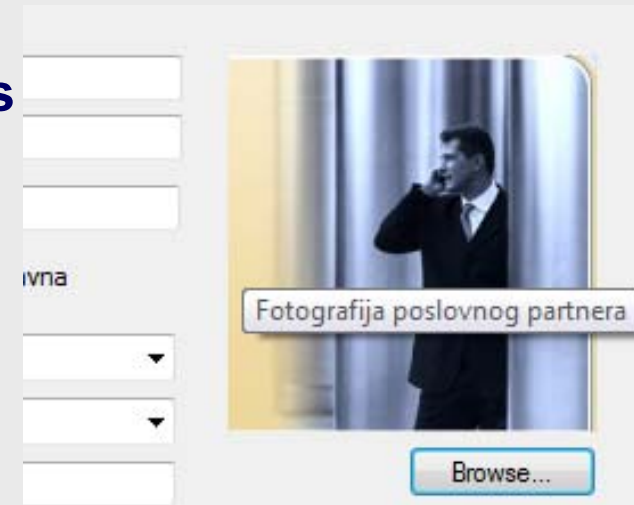
- `AutomaticDelay` – vrijednost za automatsku postavku ostalih vremena
- `InitialDelay` – vrijeme do prvog prikaza (= `AutomaticDelay`)
- `AutoPopDelay` – vrijeme trajanja prikaza (= `AutomaticDelay*10`)
- `ReshowDelay` – vrijeme do ponovnog prikaza (= `AutomaticDelay/5`)
- `ShowAlways` – aktivan i kad kontrola kojoj je pridružen nije u fokusu

■ Postupci (uobičajeno u forma.Designer.cs)

- `SetToolTip(Control, string)` – kontroli postavlja objašnjenje, u dizajnu se vidi u popisu svojstava kao "ToolTip on control"
- `SetToolTip(Control)` – vraća objašnjenje kontrole u pogonu

❑ **Primjer: GUI \ Partner \ Form1.Designer.cs**

```
ToolTip toolTip = new  
    ToolTip(this.components);  
  
...  
toolTip.SetToolTip(pictureBoxSlika,  
    "Fotografija poslovnog partnera");
```



Nasljeđivanje formi

❑ Jednostavno nasljeđivanje formi

```
public partial class Form2 : Partner.Form1
{
    public Form2()
    {
        InitializeComponent();
    }
    private void InitializeComponent()
    {
        this.Text = this.Text + " naslijeđena";
        this.BackColor = System.Drawing.Color.Yellow;
    }
}
```

❑ Nasljeđivanje u dizajnu

- Izvedena forma dodaje se s *Project* → *Add* → *New Item* → *InheritedForm*
- Mogu se postavljati svojstva izvedenoj formi ali ne njezinim kontrolama (bolji primjeri nasljeđivanja formi slijede na narednim predavanjima)

Dinamičko kreiranje kontrola

❏ Primjer GUI\KreiranjeKontrola

■ Kreiranje gumba klikom na formu

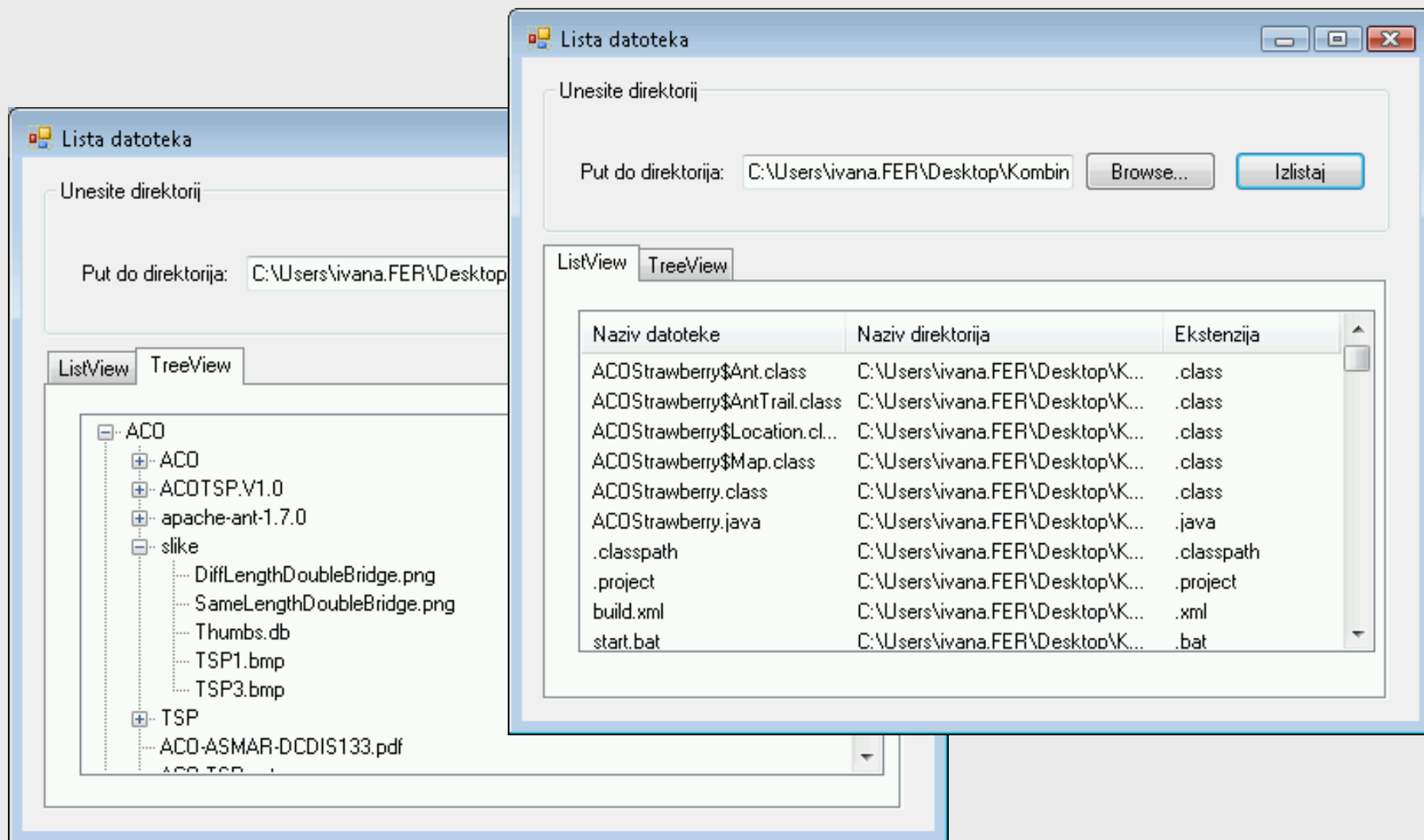
```
private void Form1_Click(object sender, EventArgs e)
{
    Button b = new Button();
    b.Text = "Gumb " + this.Controls.Count;
    Point mousePoint = PointToClient(MousePosition);
    b.Location = new Point(mousePoint.X, mousePoint.Y);

    b.Click += new System.EventHandler(this.ButtonClick);
    this.Controls.Add(b);
}

private void ButtonClick(object sender, System.EventArgs e)
{
    MessageBox.Show(sender.ToString());
}
```

Primjer kontrola *TabControl*, *ListView*, *TreeView*

❑ Primjer: GUI\ListaDatoteka



TabControl

❑ **TabControl** - Kontrola s karticama (*tabovima*)

■ Svojstva

- Alignment – poravnanje tabova (Top, Bottom, Left, Right)
- Appearance – izgled tabova (Normal, Buttons, FlatButtons)
- HotTrack – dinamičko isticanje (highlight) taba pri prolasku miša
- Images – kolekcija bitmapa za tabove
- ItemSize – veličina tabova
- Multiline – tabovi u više redaka pri sužavanju kontrole
- SizeMode – automatsko podešavanje širine tabova (Normal, Fixed, FilledToRight)
- TabPages – kolekcija tabova
- TabCount – broj tabova
- SelectedIndex – indeks odabranog taba

■ Događaji

- SelectedIndexChanged – promjena svojstva SelectedIndex

❑ **TabPage**

■ Svojstva # otprije poznata

- AutoScroll, BackgroundImage, ImageIndex, Text

Primjer *TabControl*

❑ *TabControl* kontrola

```
private void PrikaziSveDokumente() {  
    if (tabControlLista.SelectedIndex == 0)  
        PrikaziDokumenteListView(textBoxPut.Text);  
    else PrikaziDokumenteTreeView(textBoxPut.Text, null);  
}
```

❑ Rekurzivna obrada hijerarhije datoteka u poddirektorijima

- Korištenje prostora imena `System.IO` – detaljnije u narednim predavanjima

```
private void PrikaziDokumenteListView(string imeDirektorija)  
{  
    DirectoryInfo directoryInfo = new DirectoryInfo(imeDirektorija);  
    FileSystemInfo[] listFiles = directoryInfo.GetFileSystemInfos();  
    foreach (FileSystemInfo fileSystemInfo in listFiles)  
    {  
        if (fileSystemInfo is FileInfo) ... //dodaj u listu  
        if (fileSystemInfo is DirectoryInfo)  
            PrikaziDokumenteListView(fileSystemInfo.FullName);  
    }  
}
```

ListView

❑ Prikaz kolekcije elemenata u mreži

■ Svojstva

- View – enum { Details, LargeIcon, List, SmallIcon }
- SmallImageList, LargeImageList – liste slika za *SmallIcon* i *LargeIcon*
- AllowColumnReorder, AutoArrange, GridLines – prilagodba prikaza
- CheckBoxes – *CheckBox* elementi
- FullRowSelect – označava se cijeli redak odabranog elementa
- Multiselect – odabir više elemenata
- Columns – lista zaglavlja stupaca,
 - ColumnHeader razred sa svojstvima: Index, Text, Width
- Items – lista ListViewItem elemenata
 - Svojstva: Count,
 - Postupci: Add, Clear, Insert, Remove
- CheckedItems, SelectedItems – lista označenih, lista odabranih

■ Događaji

- SelectedIndexChanged, StyleChanged, ...

❑ ListViewItem element

■ Svojstva:

- Index – indeks elementa u listi
- ImageIndex – indeks slike u SmallImageList, odnosno LargeImageList
- Selected – element je označen

Primjer *ListView*

❑ Prikaz u *ListView*

```
DirectoryInfo directoryInfo = new DirectoryInfo(imeDirektorija);
FileSystemInfo[] listFiles = directoryInfo.GetFileSystemInfos();

foreach (FileSystemInfo fileSystemInfo in listFiles)
{
    if (fileSystemInfo is FileInfo)
    {
        ListViewItem listItem = new ListViewItem(fileSystemInfo.Name);
        listItem.SubItems.Add(((FileInfo)fileSystemInfo).DirectoryName);
        listItem.SubItems.Add(fileSystemInfo.Extension);
        listView.Items.Add(listItem);
    }
    if (fileSystemInfo is DirectoryInfo)
    {
        PrikaziDokumenteListView(fileSystemInfo.FullName);
    }
}
```

TreeView

❑ **TreeView – stablasta lista čvorova**

■ Svojstva

- CheckBoxes – *CheckBox* čvorovi
- Nodes – *TreeNodeCollection*
- ShowLines – prikaz poveznica između čvorova
- ShowPlusMinus – prikaz znaka za širenje/skupljanje čvora
- TopNode – čvor na vrhu

■ Postupci

- CollapseAll, ExpandAll – širenje i skupljanje prikaza

■ Događaji

- BeforeSelect, AfterSelect
- BeforeCollapse, AfterCollapse
- BeforeExpand, AfterExpand

❑ **Nodes (TreeNodeCollection) - kolekcija čvorova**

■ Postupci

- Add, Remove ...

❑ **TreeNode – čvor**

■ Svojstva

- Text
- FullPath – put do čvora
- Index – indeks čvora
- Nodes – kolekcija djece
- FirstNode, LastNode - djeca
- NextNode, PrevNode – braća
- IsExpanded, IsSelected

■ Postupci

- Collapse, Expand – širenje i skupljanje prikaza odabranog

Primjer *TreeView*

❑ Prikaz u *TreeView*

```
DirectoryInfo directoryInfo = new DirectoryInfo(imeDirektorija);
FileSystemInfo[] listFiles = directoryInfo.GetFileSystemInfos();

foreach (FileSystemInfo fileInfo in listFiles)
{
    TreeNode treeItem = new TreeNode(fileInfo.Name);

    if (parentNode != null)
        parentNode.Nodes.Add(treeItem);
    else
        treeView.Nodes.Add(treeItem);

    if (fileInfo is DirectoryInfo)
        PrikaziDokumenteTreeView(fileInfo.FullName, treeItem);
}
```

Prijenos parametara među formama

- ❑ Forma poziva drugu formu i prosljeđuje joj parametre
- ❑ Izmjene na drugoj formi koriste se za ažuriranje na pozivajućoj formi

The diagram illustrates the transfer of parameters between two forms. The first form, titled "Unos podataka o partneru", contains the following fields: "Ime:" (Ivo), "Prezime:" (Ivić), "JMBG:" (1111), and "Tip:" with radio buttons for "Fizička" (selected) and "Pravna". Below these is a section titled "Adresa" with fields for "Država:", "Mjesto:", and "Ulica i broj:". A "Unesi..." button is located at the bottom right of this section. The second form, titled "Unos adrese", is a modal dialog that appears over the first. It contains fields for "Država:" (Hrvatska), "Mjesto:" (Split), and "Ulica i broj:" (Vukovarska 15). It also has "Spremi" and "Odustani" buttons. A curved arrow points from the "Adresa" section of the first form to the "Unos adrese" dialog, and another curved arrow points from the "Unos adrese" dialog back to the "Unesi..." button of the first form, indicating the flow of data.

Prijenos parametara među formama (2)

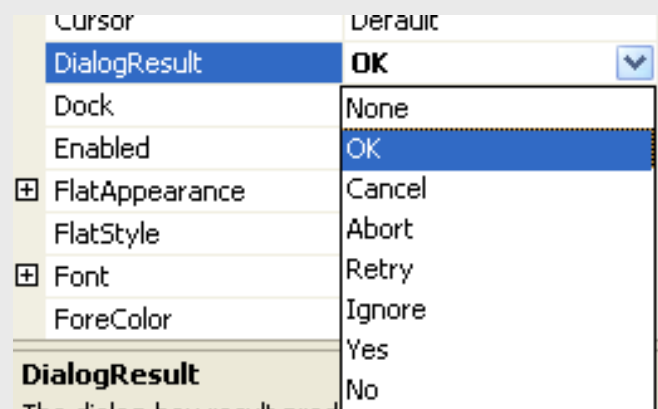
❑ **Primjer:**  GUI \ PrijenosParametara

❑ **Prijenos parametara iz pozvane forme u pozivajuću – FormaAdresa**

- FormaAdresa ima svojstva Drzava, Mjesto i Ulica koje su omotač oko pripadnih kontrola za unos

```
public string Drzava {  
    get {  
        if (comboBoxDrzava.SelectedItem == null) return null;  
        else return comboBoxDrzava.SelectedItem.ToString();  
    }  
    set { comboBoxDrzava.SelectedItem = value; }  
}
```

- Gumbi *Spremi* i *Odustani* imaju postavljena svojstva DialogResult na *OK*, odnosno *Cancel*



Prijenos parametara među formama (3)

❑ Prijenos parametara iz pozvane forme u pozivajuću – FormaUnos

- FormaUnos instancira formu FormaAdresa i otvara je kao dijalog

```
private void buttonUnos_Click(object sender, EventArgs e){  
    FormAdresa f = new FormAdresa();  
    ...  
    if (f.ShowDialog() == DialogResult.OK)  
    {  
        textBoxDrzava.Text = f.Drzava;  
        textBoxMjesto.Text = f.Mjesto;  
        textBoxAdresa.Text = f.Ulica;  
    }  
}
```

- Ako je na formi FormaAdresa odabran gumb *Spremi*, FormaUnos dohvaća vrijednosti Drzava, Mjesto i Ulica forme FormaAdresa i postavlja ih kao vrijednosti svojim pripadajućim kontrolama
- Ako je odabran gumb *Odustani* vrijednosti se ne dohvaćaju

Prijenos parametara među formama (4)

❑ Prijenos parametara iz pozivajuće forme u pozvanu – FormaUnos

- Prijenos parametara u pozivajuću formu može se izvesti slanjem parametara u konstruktor pozvane forme
- `FormaAdresa f = new FormaAdresa(textBoxDrzava.Text);`
 - U primjeru, forma `FormaUnos` može mijenjati svojstva forme `FormaAdresa` pa je prijenos napravljen postavljanjem svojstava preko reference

```
private void buttonUnos_Click(object sender, EventArgs e) {  
    FormaAdresa f = new FormaAdresa();  
    f.Drzava = textBoxDrzava.Text;  
    f.Mjesto = textBoxMjesto.Text;  
    f.Ulica = textBoxAdresa.Text;  
  
    if (f.ShowDialog() == DialogResult.OK) { ... }  
}
```

- Rješenje je napravljeno po uzoru na predefinirane dijaloge (npr. *OpenFileDialog*)

Višenitnost

❑ Nit ili dretva (eng. *thread*)

- Operacijski sustav razdvaja izvođenje aplikacija u procese
- Niti su osnovne jedinice u kojima OS raspodjeljuje procesorsko vrijeme
- Ime niti dolazi od 'thread of execution'.

❑ Višenitnost (*multithreading*)

- Više niti može se izvoditi unutar jednog procesa
- Višenitni program simulira istovremeno izvođenje različitih niti (pseudo-paralelizam).

❑ Prednost

- Bolja iskoristivost procesorskog vremena, naročito kod programa sa sporim periferijama

❑ Primjer:

- poslužitelj (web, baze podataka), koji "istovremeno" obrađuje zahtjeve klijenata

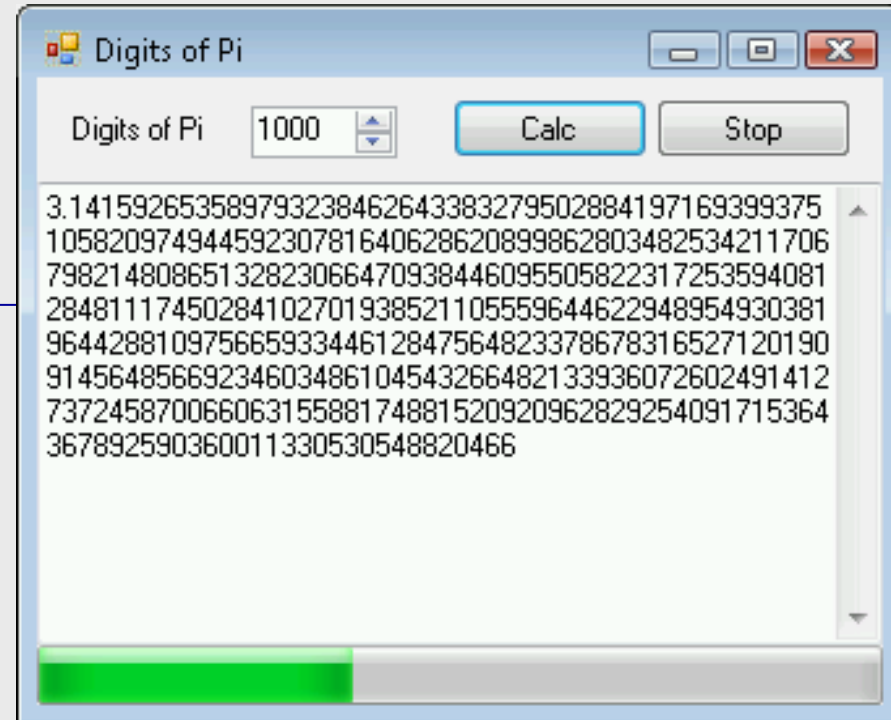
Primjer višenitnog programa

❑ Primjer GUI\RacunanjePi

- problem istovremenog prikaza rezultata i statusa za veći broj znamenki

```
void CalcPi(int digits)
{
    ...
    for (int i = 0; i < digits; i += 9)
    {
        ...//računanje sljedećih decimala
        ShowProgress(pi.ToString(), digits,
            i + digitCount); //ProgressBar
    }
}

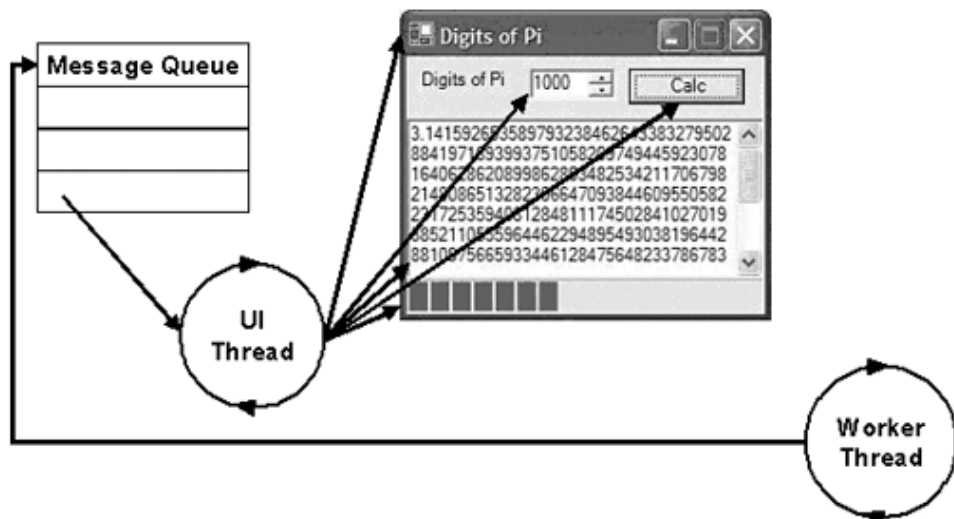
void buttonCalc_Click(object sender, EventArgs e)
{
    CalcPi(digitsToCalc);
}
```



Pokretanje niti

- ❑ **Prostor imena `System.Threading`**
- ❑ **Da bi se koristila nit, treba instancirati objekt tipa `Thread`.**
 - `public Thread(ThreadStart entryPoint)`
 - `ThreadStart` - delegat
 - `entryPoint` – postupak koji započinje nit
- ❑ **Kreirana nit se pokreće postupkom `Start()`**

```
Thread piThread = new Thread(  
    new ThreadStart(CalcPiThreadStart));  
piThread.Start();
```



Form1
Class
Form

Fields

Methods

- buttonCalc_Click
- buttonStop_Click
- CalcPi
- CalcPiThreadStart
- Dispose
- Form1
- InitializeComponent
- ShowProgress

Nested Types

ShowProgressDelegate
Delegate


Računanje znamenki broja Pi

❑ Rad niti u primjeru treba biti asinkron s računanjem znamenki

- Invoke – postupak za sinkroni poziv (ne koristimo u primjeru)
- **BeginInvoke**, **EndInvoke** – postupci za asinkroni poziv delegata
- **InvokeRequired** – pozivatelj treba "prizvati" postupak jer je na drugoj niti

```
delegate void ShowProgressDelegate(  
    string pi, int totalDigits, int digitsSoFar);  
  
void ShowProgress(string pi, int totalDigits, int digitsSoFar)  
{  
    if (this.InvokeRequired) // poziv postupka iz glavne niti  
    {  
        ShowProgressDelegate showProgress = new  
            ShowProgressDelegate(ShowProgress);  
        this.BeginInvoke(showProgress,  
            new object[] { pi, totalDigits,digitsSoFar });  
    }  
    else {...} // izvršenje na drugoj niti  
}
```

Ostalo o nitima ...

- ❑ **Statički postupak `sleep`** - uzrokuje privremeno zaustavljanje izvršavanja niti iz koje je pozvana za navedeni broj milisekundi.
 - `Thread.Sleep(n)`
- ❑ **svojstvo `Priority`** - Postavljanje prioriteta pojedine niti
 - `public ThreadPriority Priority{ get; set; }`
- ❑ **svojstvo `ThreadState`** - stanje niti
- ❑ `Thread.CurrentThread` - referenca na trenutnu nit
- ❑ `Thread.Suspend()` – privremeno zaustavlja izvršavanje niti
- ❑ `Thread.Resume()` – ponovno pokreće privremeno zaustavljenu nit
- ❑ `Thread.Abort()` – uzrokuje `ThreadAbortException` na niti na kojoj se izvrši, što uzrokuje završetak niti
 - Primjer  `GUI\RacunanjePi`

```
private void buttonStop_Click(object sender, EventArgs e)
{
    piThread.Abort();
}
```

NumericUpDown

□ **NumericUpDown**

■ Svojstva:

- `UpDownAlign` – položaj upravljačkih *Button* kontrola
- `TextAlign` – poravnanje prikazanog sadržaja
- `Minimum` – najmanja vrijednost kontrole
- `Maximum` – najveća vrijednost kontrole
- `Increment` – korak vrijednosti
- `DecimalPlaces` – broj prikazanih decimala
- `ThousandSeparator` – prikaz oznake za tisućice
- `Value` – sadržana vrijednost

■ Događaj `ValueChanged`

ProgressBar

❑ **Kontrola za informiranje o napredovanju nekog procesa**

❑ **Svojstva**

- **Minimum, Maximum** – najmanja i najveća vrijednost
- **Value** – trenutni položaj oznake
- **Step** – korak promjene položaja oznake pri pozivu PerformStep postupka

❑ **Postupci**

- **Increment** – promjena vrijednosti za navedenu veličinu
- **PerformStep** – promjena vrijednosti za veličinu Step

❑ **Primjer**  **GUI\RacunanjePi**

```
void ShowProgress(string pi, int totalDigits, int digitsSoFar)
{
    if (this.InvokeRequired) { ... }
    else{
        piTextBox.Text = pi;
        piProgressBar.Maximum = totalDigits;
        piProgressBar.Value = digitsSoFar;
    }
}
```


Vlastite kontrole

❑ Vlastite kontrole (*custom controls*)

- Elementi sučelja koje stvara korisnik razvojnog pomagala – programer
- Nakon toga može ih se koristiti u različitim programima, kao i druge preddefinirane kontrole, odabirom iz kutije s alatima

❑ Tipovi vlastitih kontrola:

❑ **User kontrole ("korisničke")**

- Nasljeđivanje `System.Windows.Forms.UserControl`
- Kombinacija više postojećih kontrola u logičku cjelinu

❑ **Owner-draw kontrole ("nacrtane")**

- Nasljeđivanje `System.Windows.Forms.Control`
- Crtanje kontrole ispočetka

❑ Naslijeđene kontrole

- Nasljeđivanje kontrole koja je najbližnja onoj koju želimo napraviti
- Dodavanje novih svojstava i postupaka

Primjer vlastite "korisničke" kontrole

❑ **Windows Forms Control Library** projekt

- rezultat je .dll datoteka (*class library*)



❑ **Kontrola nasljeđuje razred UserControl**

- `public partial class VlastitiTextBox : UserControl`

❑ **Primjer** **GUI\VlastiteKontrole – razred VlastitiTextBox**

- Implementiramo funkcionalnosti koje želimo (npr. rukovatelj događajem izmjene veličine kontrole)

```
private void CustomTextBox_Resize(object sender, EventArgs e) {  
    textBox.Width = (2 * this.Width / 3);  
    labelText.Width = (this.Width / 3);  
}
```

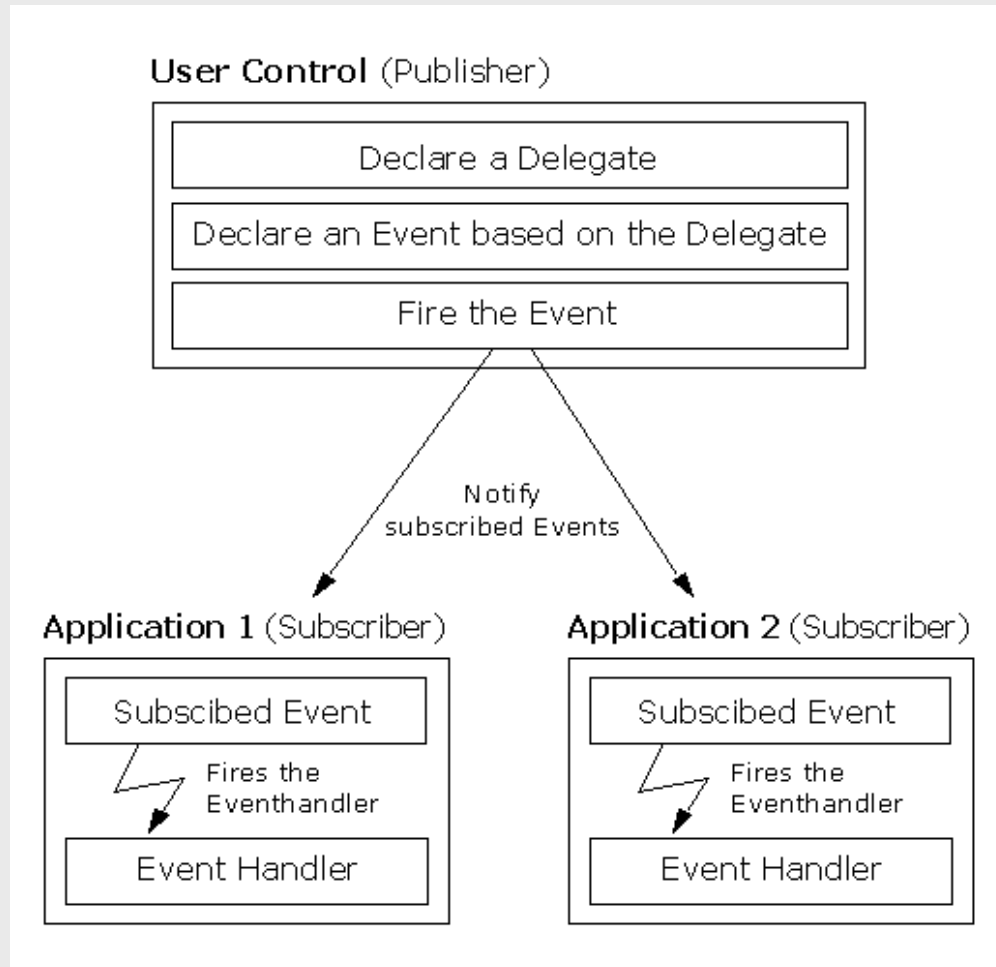
- Implementiramo svojstva vlastite kontrole kojima pristupamo iz aplikacije

```
public string LabelaTekst {  
    get { return labelText.Text; }  
    set { labelText.Text = value; }  
}
```

Događaji na vlastitoj kontroli

❑ Model izdavač-pretpatnik (publisher-subscriber)

- Objekt “objavljuje” događaj, a aplikacija se na njega “pretplaćuje”
- U ovom modelu vlastita kontrola je izdavač



Primjer: vlastita *TextBox* kontrola

❑ Delegat i događaji pridruženi delegatu

- primjer, promjene aktivnosti *TextBox* kontrole

```
public delegate void TextBoxActivityHandler();  
public event TextBoxActivityHandler TextBoxEntered;  
public event TextBoxActivityHandler TextBoxLeaved;
```

❑ Rukovatelj događajem

- Događaj `textBox_Enter` – *TextBox* postaje aktivna kontrola forme
- provjera postoji li pretplatnik na događaj i okidanje događaja (način obrade događaja bit će definiran u aplikaciji koja koristi kontrolu, kasnije)

```
private void textBox_Enter(object sender, EventArgs e)  
{  
    if (TextBoxEntered != null)  
    {  
        TextBoxEntered(); // definiran u aplikaciji  
    }  
}
```

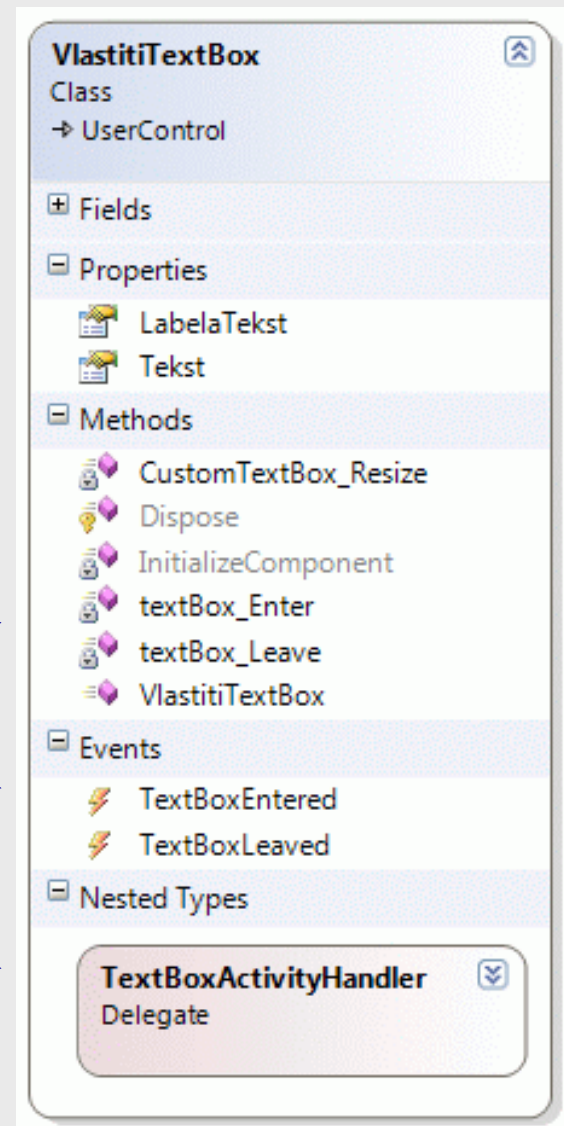
Obrada događaja na vlastitoj kontroli

☐ Obrada događaja na vlastitoj kontroli

- ☐ **Rukovatelj događajem**
 - lokalno na vlastitoj kontroli

- ☐ **Događaj pridružen delegatu**
 - vidljiv aplikaciji koja koristi kontrolu

☐ Delegat



Upotreba vlastite kontrole

❑ Dodavanje vlastite kontrole (VlastiteKontrolle.dll) u projekt

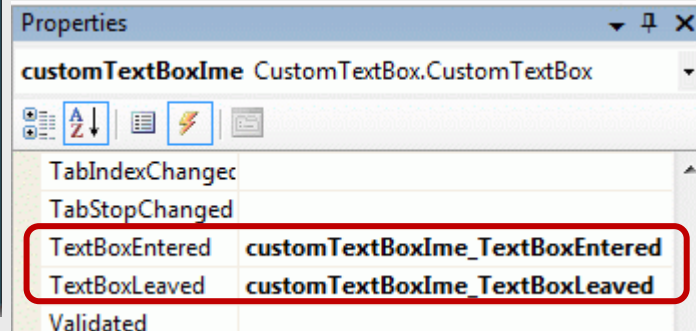
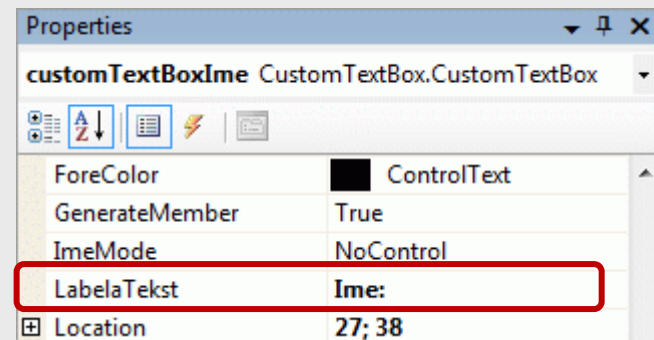
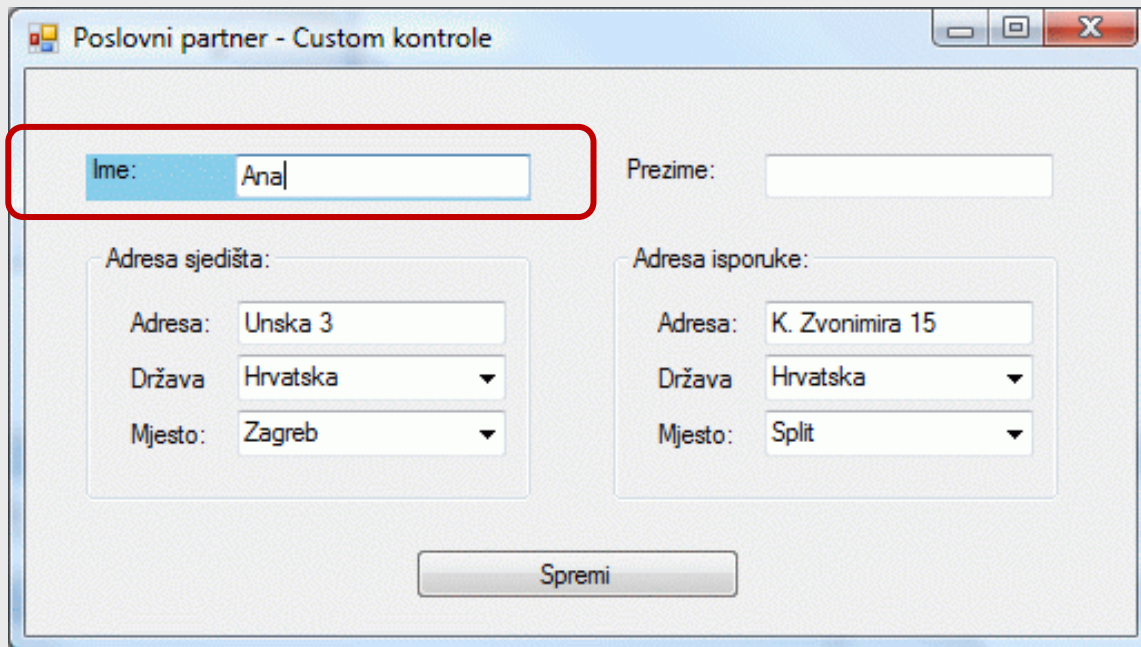
❑ Primjer  GUI\Custom

- Desni klik na *Toolbox* → *Choose Items* → *Browse...* *VlastiteKontrolle.dll*

- Ili, desni klik na projekt – *Add reference* - *Project* - *VlastiteKontrolle*

❑ Dodamo kontrolu s *ToolBoxa* i postavljamo joj svojstva

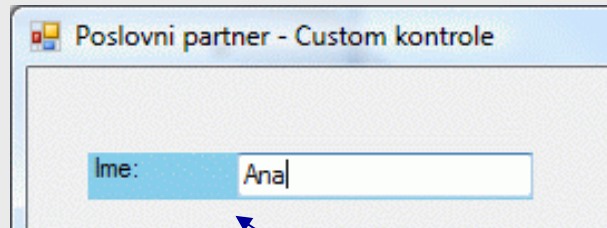
- Na popisu su svojstva i događaji koje smo sami definirali



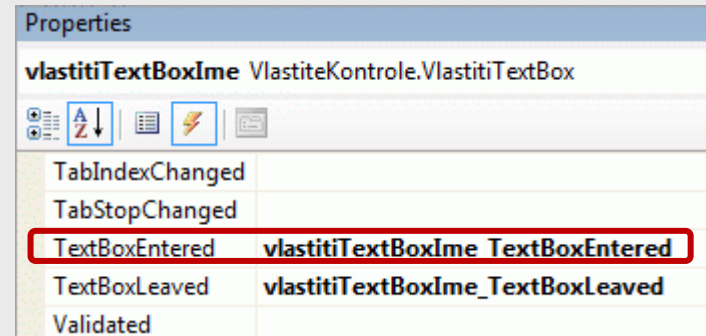
Obrada događaja vlastite kontrole

❑ Implementiramo rukovatelj događajem

- događaj `TextBoxEntered` u aplikaciji koja koristi kontrolu



`vlastitiTextBoxIme`



- Generira se kôd koji postavlja `customTextBoxIme_TextBoxEntered` za rukovatelja događajem `TextBoxEntered`

```
this.vlastitiTextBoxIme.TextBoxEntered += new  
    VlastiteKontrola.VlastitiTextBox.TextBoxActivityHandler  
    (this.vlastitiTextBoxIme_TextBoxEntered);
```

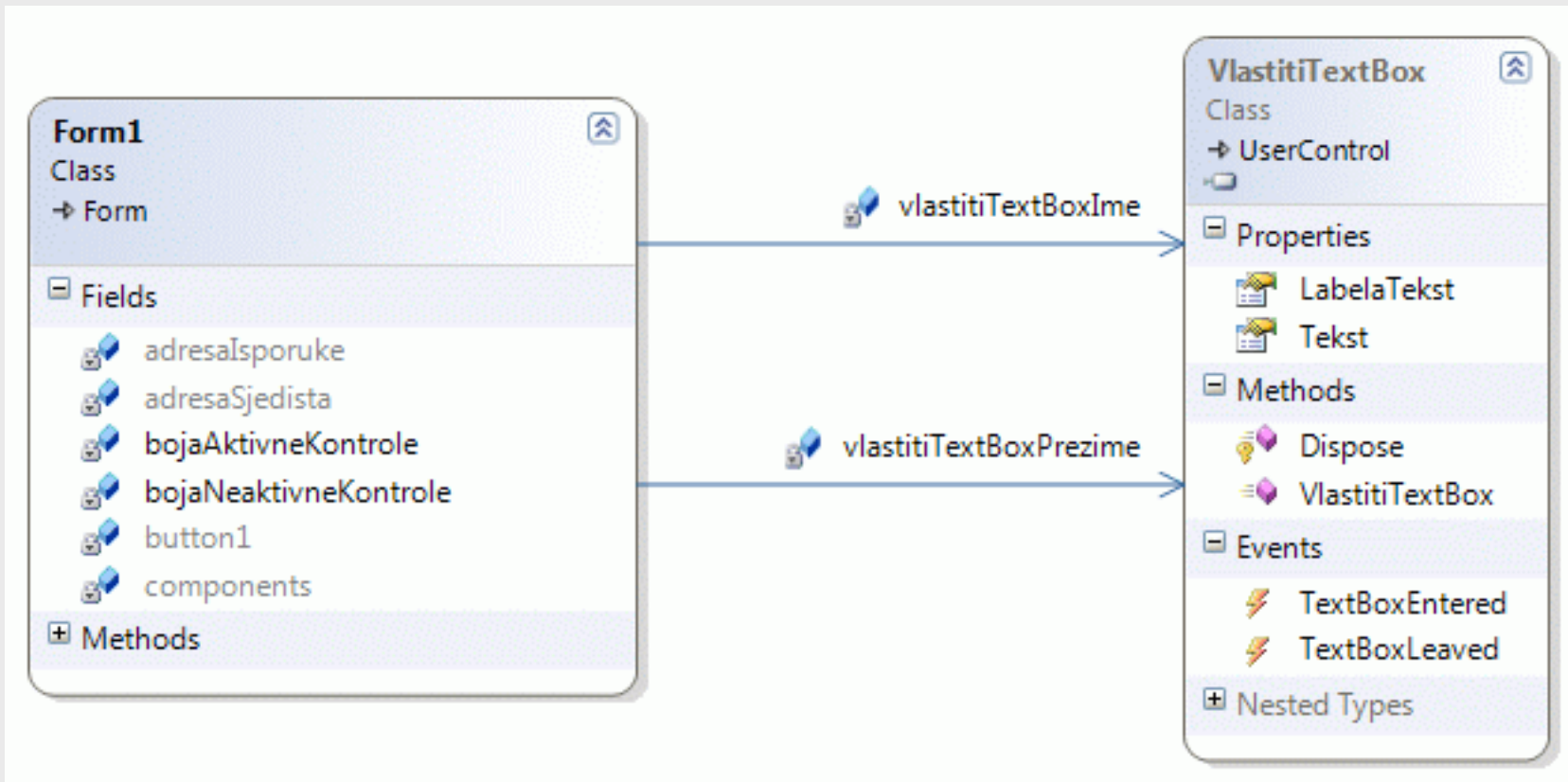
- Rukovatelj događajem: `vlastitiTextBoxIme_TextBoxEntered`

```
Color bojaAktivneKontrola = Color.SkyBlue;  
private void vlastitiTextBoxIme_TextBoxEntered() {  
    vlastitiTextBoxIme.BackColor = bojaAktivneKontrola;  
}
```


Dijagram razreda aplikacije i vlastite kontrole

❑ Dijagram razreda za primjer GUI\Custom

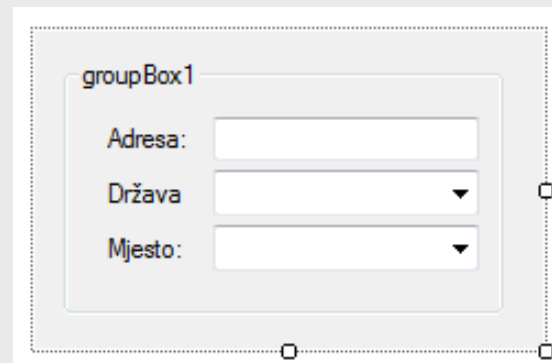
- Pretplatnik `Form1` i izdavač `VlastitiTextBox`
- Svojstva `vlastitiTextBoxIme` i `vlastitiTextBoxPrezime` prikazana su kao asocijacija



Još jedan primjer vlastite kontrole ...

❑ Primjer GUI\VlastiteKontrole – razred Adresa

❑ Korisnička kontrola koja omogućava unos adrese (ulica, država, mjesto)



- Osnovna svojstva (dohvat upisane adrese, države, mjesta, naslov *GroupBox* kontrole, ...)

```
public string AdresaTekst{
    get { return textBoxAdresa.Text; }
    set { textBoxAdresa.Text = value; }
}
public string DrzavaTekst { get { ... } set { ... } }
...
```

- Prilikom promjene države, mijenja se ponuđena lista gradova!
 - O tome treba brinuti korisnička kontrola
 - Aplikacija koja kontrolu koristi, jednom (za jednu instancu) postavlja za *DataSource* – listu država i pripadnih gradova

Još jedan primjer vlastite kontrole ... (2)

- ❑ Definiramo razred **Drzava** koji će **sadržavati naziv i listu gradova**

```
public class Drzava{  
    private string nazivDrzave;  
    private List<string> gradovi;  
}
```

- ❑ **Korisnička kontrola ima svojstvo DrzavaDataSource**

```
public List<Drzava> DrzavaDataSource { get { ... } set { ... } }
```

- ❑ **Prilikom promjene države u padajućem izborniku, mijenja se i ponuđena lista gradova**

```
private void comboBoxDrzava_SelectedIndexChanged(object sender,  
EventArgs e){  
    comboBoxMjesto.DataSource =  
        ((Drzava)comboBoxDrzava.SelectedValue).Gradovi; }  
}
```

- ❑ **Primjer korištenja**  **GUI\Custom**

```
private VlastiteKontrole.Adresa adresaSjedista;  
List<Drzava> listaDrzava;  
  
adresaSjedista.DrzavaDataSource = listaDrzava;  
adresaIsporuke.DrzavaDataSource = listaDrzava;
```

Još jedan primjer vlastite kontrole ... (3)

❑ Postavljanje različitih vrijednosti na različitim padajućim izbornicima

- Različiti izbornici nad istim izvorom podataka
 - Izborom jedne vrijednosti na jednom, mijenja se vrijednost i drugom
- Potrebno je padajućim izbornicima postaviti različiti `BindingContext`:
- Više o `BindingContext` u narednim predavanjima

❑ U primjeru, postavljanje `BindingContext`-a može se obaviti prilikom inicijalizacije kontrole

```
public Adresa()  
{  
    InitializeComponent();  
    comboBoxDrzava.BindingContext = new BindingContext();  
    comboBoxMjesto.BindingContext = new BindingContext();  
}
```

Primjeri drugih kontrola

ImageList

❑ Razred s listom *Image* objekata

- `Images` – lista *Image* objekata
 - Svojstva: `Count`, `Empty` – svojstva liste
 - Postupci: `Add`, `Clear`, `Remove` – rukovanje elementima liste
- `ImageSize` – veličina bitmape pojedinog *Image* objekta u listi
 - standardno (16;16), najviše (255;255)

❑ `Button` i neke druge kontrole imaju svojstva

- `ImageList` referenca
- `ImageIndex` – indeks *ImageList* člana

❑ Definiranje bitmapa u dizajnu:


- `imageList1.Images` (Properties ... collection)

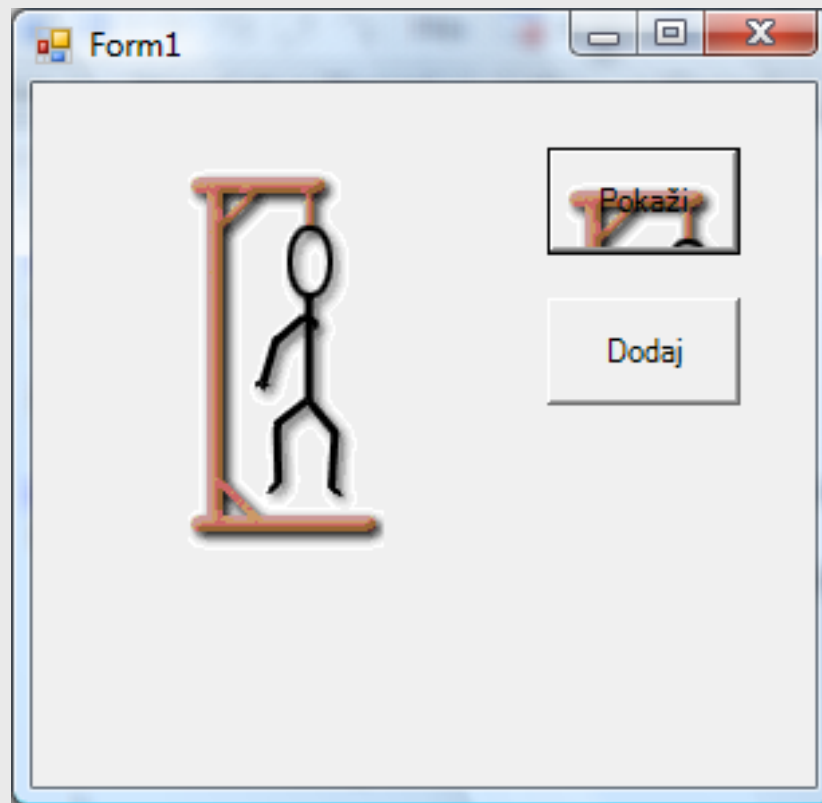
❑ Dodavanje bitmape programski:

- `imageList1.Images.Add(Image.FromFile(openFileDialog1.FileName));`

Zadaci za vježbu

❑ Napraviti formu za prikaz vješala za igru Vješala.

- Unaprijed pripremljen niz slika prikazati u *ImageList*
- Klikom na gumb Prikaži dodaje se sljedeća slika
- Klikom na gumb Dodaj se može dodati nova slika
- Primjer:  GUI\Vjesala



Timer

❑ Kontrola koja izaziva događaj u korisnički definiranim intervalima

❑ Svojstva

- `Interval` – vrijeme (ms) između 2 otkucaja, vrijednost mora biti ≥ 1
 - broj sekundi proteklih između 2 otkucaja izračunava se kao $\text{Interval}/1000$
 - $\text{Interval} = 100$ – mjeri desetinke, $\text{Interval} = 10$ – mjeri stotinke
 - izraz $1000/\text{Interval}$ = broj manjih vremenskih jedinica u sekundi
- `Enabled` – `true`:pokrenut, `false`:zaustavljen

❑ Događaj

- `Tick` – otkucaj

❑ Primjer: GUI \ Timer

RichTextBox

❑ **Jača verzija TextBox-a (mini word procesor)**

❑ **Svojstva i postupci**

- `RichTextBoxStreamType` – vrsta teksta
 - `PlainText`, `RichText`, `UnicodePlainText`, itd.
- `LoadFile` - Čitanje dokumenta
- `SaveFile` - Pisanje dokumenta
- `SelectedText`, `SelectedRTF` – označeni tekst
- `SelectionLength` – duljina označenog teksta
- `SelectionFont` – Font označenog teksta
- `SelectionColor` – Boja označenog teksta

❑ **Primjer: GUI \ RichTextBoxPad**

ScrollBar

❑ Klizne trake, pomične trake

❑ *HScrollBar* – vodoravna klizna traka

❑ *VScrollBar* – okomita klizna traka

❑ Svojstva:

- `Minimum` – najmanja vrijednost kliznika kontrole
- `Maximum` – najveća vrijednost kliznika kontrole
- `SmallChange` – korak vrijednosti za male pomake kliznika ili klik na rub
- `LargeChange` – vrijednost inkrementa kad korisnik klikne unutar kontrole
- `Value` – vrijednost kontrole, predstavlja položaj kliznika

❑ Primjer:  **GUI \ ScrollBars**

TrackBar

❑ Tračna traka, "Tračnica"

❑ Svojstva:

- `Minimum` – najmanja vrijednost kliznika kontrole
- `Maximum` – najveća vrijednost kliznika kontrole
- `SmallChange` – korak vrijednosti za male pomake kliznika
- `LargeChange` – korak vrijednosti kad korisnik klikne unutar kontrole
- `Value` – vrijednost kontrole, predstavlja položaj kliznika
- `TickFrequency` – broj vrijednosti za koje se postavlja oznaka vrijednosti
- `TickStyle` – položaj oznaka vrijednosti

❑ Primjer: GUI \ Trackbars

DomainUpDown

□ **DomainUpDown**

■ Svojstva:

- `Items` - kolekcija objekata s vrijednostima
- `SelectedItem` – referenca na odabrani objekt
- `Sorted` – *bool* oznaka sortiranih vrijednosti
- `Text` – sadržana vrijednost

■ Događaj `SelectedItemChanged`

□ **Primjer:** **GUI \ DomainUpDown**

Događaji pri radu mišem

❑ Mouse Events (Delegate EventHandler, EventArgs e)

- `MouseEnter` – značka miša ulazi u područje kontrole
- `MouseHover` – značka lebdi iznad kontrole
- `MouseLeave` – značka miša izlazi iz područja kontrole

❑ Postupak `PointToClient(MousePosition)`

- računa koordinate značke u odnosu na kontrolu, pri čemu
- `MousePosition` - položaj značke u koordinatama zaslona

❑ Mouse Events (Delegate MouseEventArgs e)

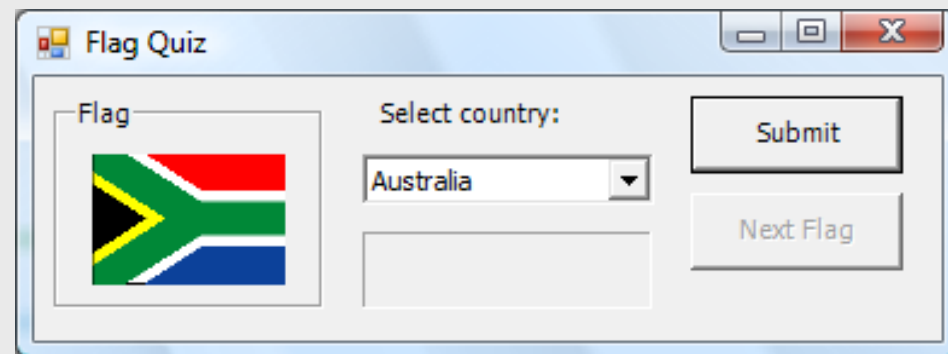
- `MouseDown` – pritisnut gumb miša u području kontrole
- `MouseMove` – pomicanje miša unutar područja kontrole
- `MouseUp` – gumb miša otpušten unutar područja kontrole
- `MouseWheel` – okrenut kotačić miša
- *MouseEventArgs* razred ima svojstva
 - `Button` – pritisnuti gumb miša (left, right, middle or none)
 - `Clicks` – broj klikova The mouse pressed
 - `Delta` – broj koraka okretanja kotačića miša
 - `X, Y` – koordinate događaja, relativno u odnosu na komponentu

❑ Primjer: GUI \ MouseEvents

Zadaci za vježbu


Zadaci za vježbu

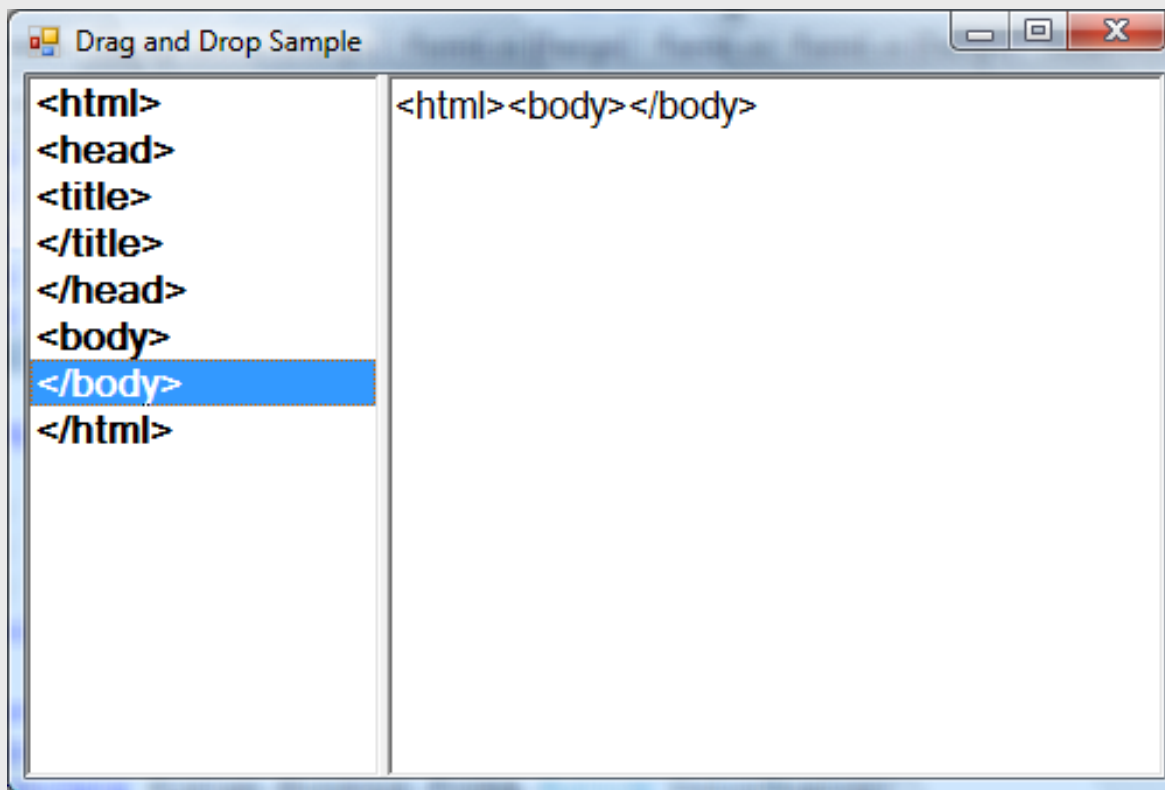
- ❑ **Napraviti zaslonsku masku iz vlastitog problemskog područja**
 - Ugraditi svojstva, postupke i obradu događaja, uključujući provjeru valjanosti podataka (validaciju).
 - Po potrebi napraviti vlastitu kontrolu (User control).
- ❑ **Napraviti formu za uređivanje slike**
 - Mogućnosti mijenjanja boje, svjetline, ...
 - Vrijednosti su mijenjaju uz pomoć NumericUpDown kontrole
 - Omogućiti spremanje promijenjene slike
- ❑ **Doraditi formu Partner radio gumbima za različite postavke rastezanja slike.**
- ❑ **Napraviti formu za pogađanje države čija je zastava prikazana**
 - Odabir u padajućem izborniku
 - Poruka o uspješnosti
 - Primjer  GUIZastave
- ❑ **Napraviti formu za unos artikla**
 - Artikl ima šifru, naziv, cijenu i sliku
 - Validirati podatke prije “spremanja”



Kako napraviti drag & drop ?

❑ Napraviti formu kao na slici

- S lijeve strane nalazi lista html oznaka (*tagova*)
- S desne nalazi se *TextBox* u koji će se zapisati oznake koje se mišem dovuku i ispuste (*drag&drop*) iz liste s lijeva
- Primjer:  GUI\Html



Automatizirano povlačenje i ispuštanje (1)

❑ Automatizirano povlačenje i ispuštanje (Drag&Drop)

❑ Podrazumijeva se postojanje dvije kontrole

- izvor (drop source)
- cilj (drop target)

❑ Cilj mora imati svojstvo

- `AllowDrop` – postavljeno na `true`

❑ Postupak kojim izvor inicira Drag&Drop operaciju

- `DoDragDrop(object data, DragDropEffects allowedEffects);`
- pri tome najavljuje dozvoljenu posljedicu

```
enum DragDropEffects {  
    Copy, // Take a copy of the data  
    Move, // Take ownership of the data  
    Link, // Link to the data  
    Scroll, // Scrolling is happening in the target  
    All, // All of the above  
    None, // Reject the data  
}
```


Automatizirano povlačenje i ispuštanje (2)

❑ Događaji (object sender, DragEventArgs e)

- DragEnter - miš ulazi u područje cilja, koji dojavljuje da li prihvaća podatke
- DragOver - prolazak miša preko cilja
- DragLeave - miš napušta područje cilja
- DragDrop - podaci ispušteni na cilj

❑ Članovi razreda DragEventArgs

- AllowedEffect – učinci koje dozvoljava izvor
- Data – podaci koji se prenose
 - GetDataPresent – postupak kojim se provjerava da li podaci odgovaraju željenom tipu
 - GetData – postupak uzimanja podataka u željenom tipu
- Effect – postavljanje ili uzimanje učinka na cilju
- KeyState – oznaka da je pritisnuta neka od upravljačkih tipki
- X, Y – screen koordinate značke

Reference

Resursi

- csharp_ebook

Dodatna literatura:

- Windows Forms Programming in C#, *Addison Wesley*
- Programming Microsoft Windows with C#, *Microsoft Press*