

Grafičko korisničko sučelje

5/13

Grafičko korisničko sučelje

❑ Graphical User Interface (GUI)

❑ 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

❑ `System.Windows.Forms` prostor imena

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

❑ *WindowsForms* i *WebForms*

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

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

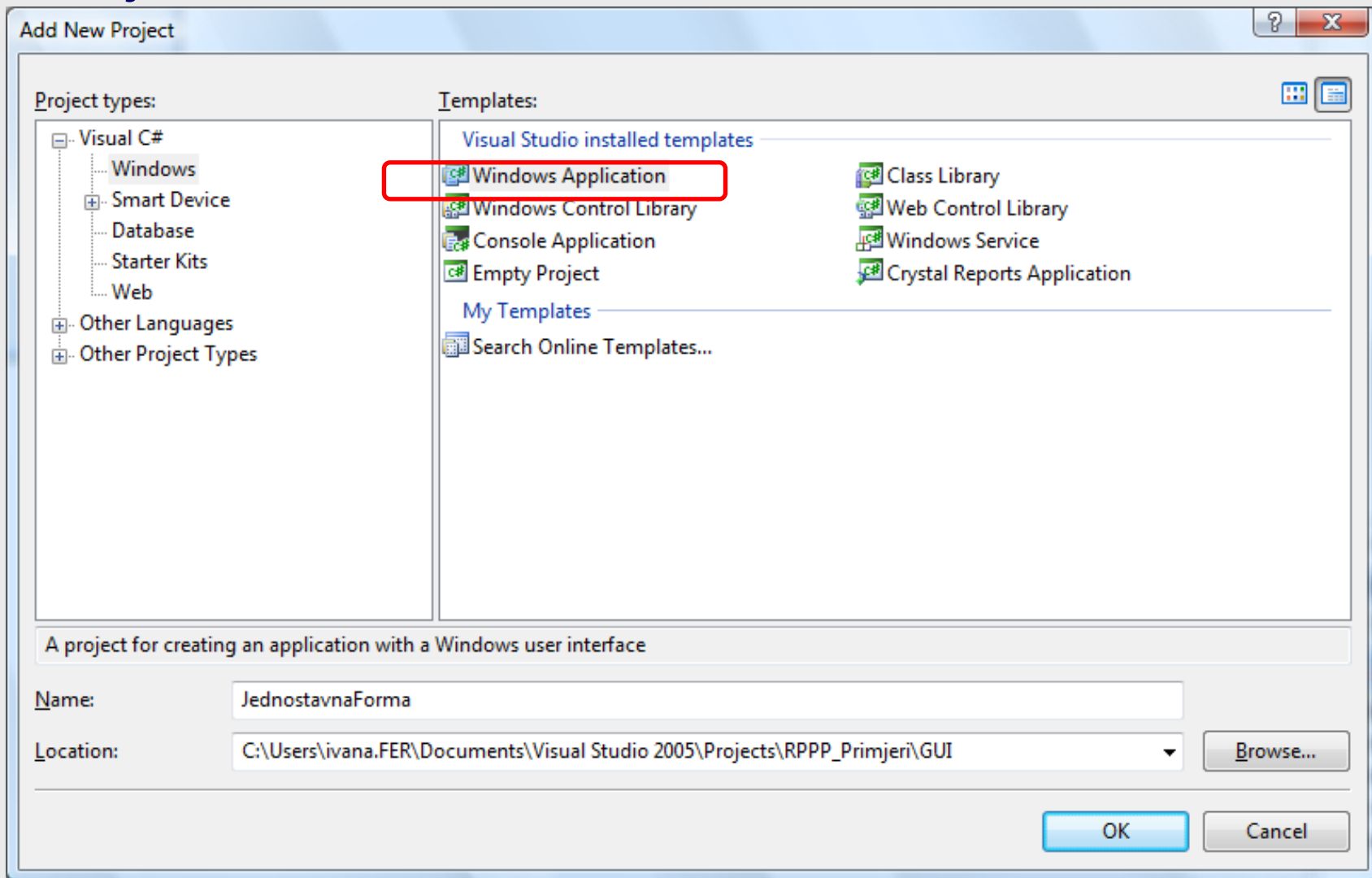
- funkcionalnost potrebna da bi kontrola sadržavala druge kontrole

☐ 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

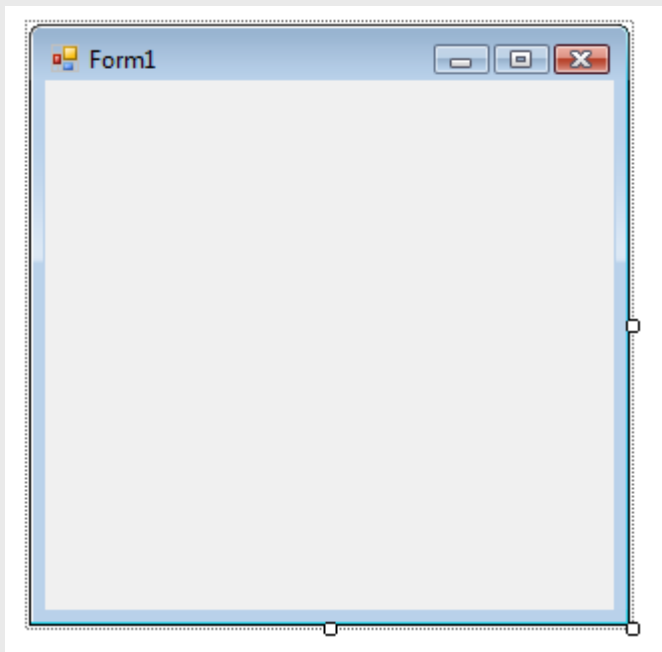
Automatsko generiranje forme

□ Primjer GUI\JednostavnaForma



Automatsko generiranje forme

❑ Form1.cs[Design] i Form1.cs



```
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();
        }
    }
}
```

partial class – ostali
dijelovi razreda nalaze se u
drugim datotekama

Automatsko generiranje forme

❑ Form1.Designer.cs

- `components` – instanca `System.ComponentModel.Container` razreda – komponenta (“spremnik” (eng. *container*) svih kontrola na formi)
- `Dispose()` - potrebno je obaviti `Dispose()` na svim kontrolama sadržanim na formi ukoliko neka kontrola drži *unmanaged* resurs, tj. takav koji nije pod kontrolom *Garbage Collector-a*

```
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

```
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

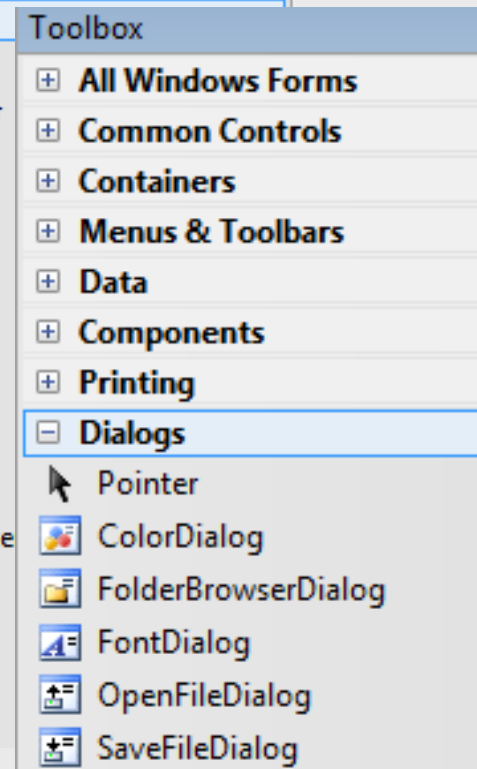
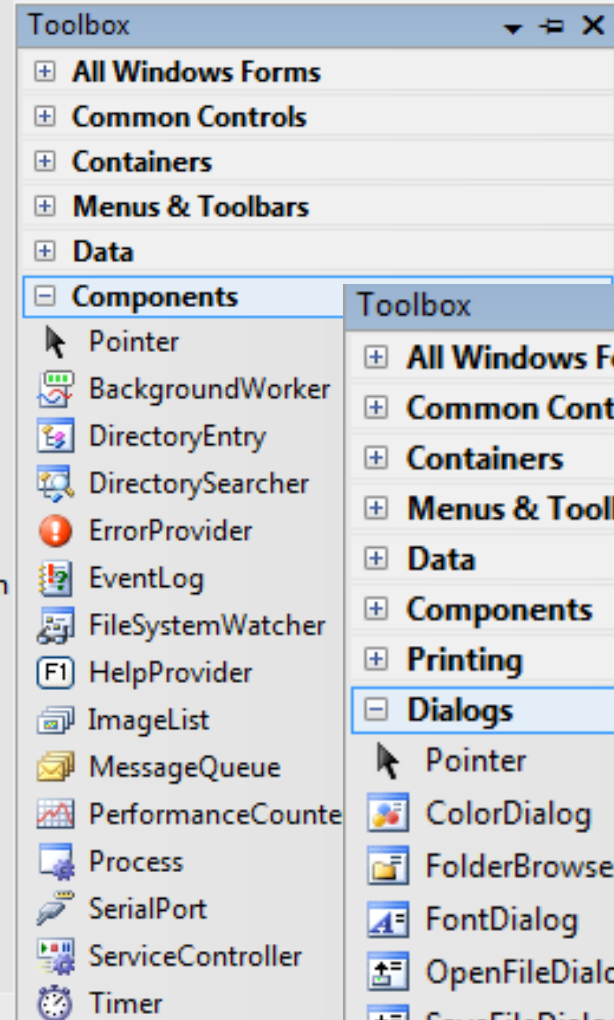
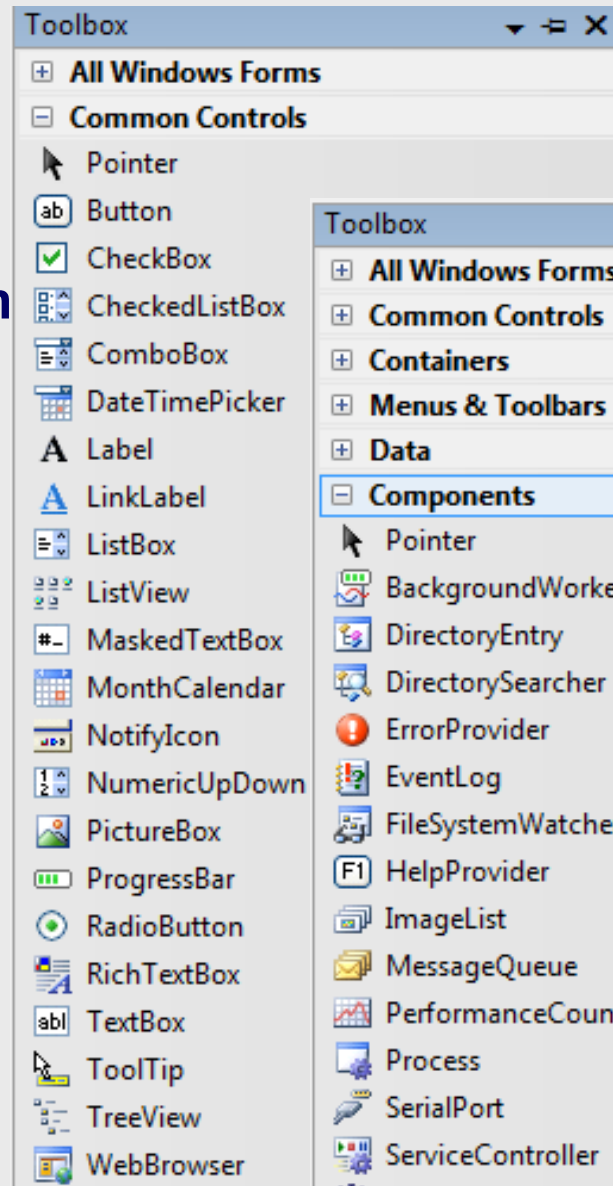
□ Program.cs

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

```
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*)
- ❑ Postavljanje i razmještaj kontrola
 - Preddefinirane kontrole
 - Korisnički definirane kontrole
 - Razmještaj u dizajnu
 - Izbornik *Format*



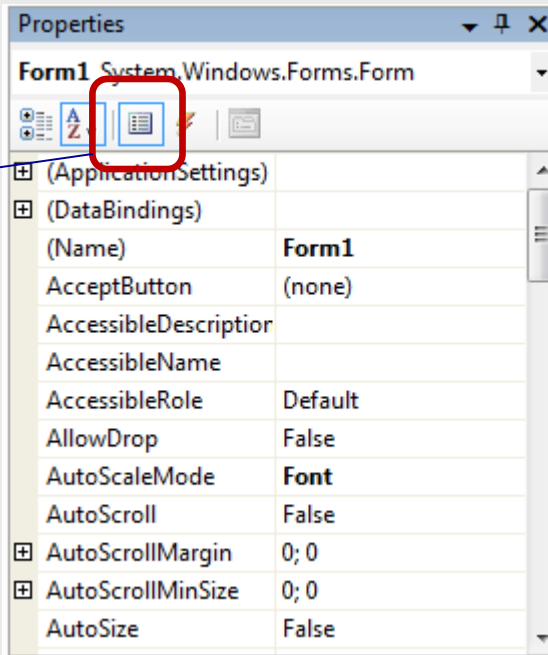
Svojstva i događaji

❑ Svojstva objekata (forme i ostalih kontrola)

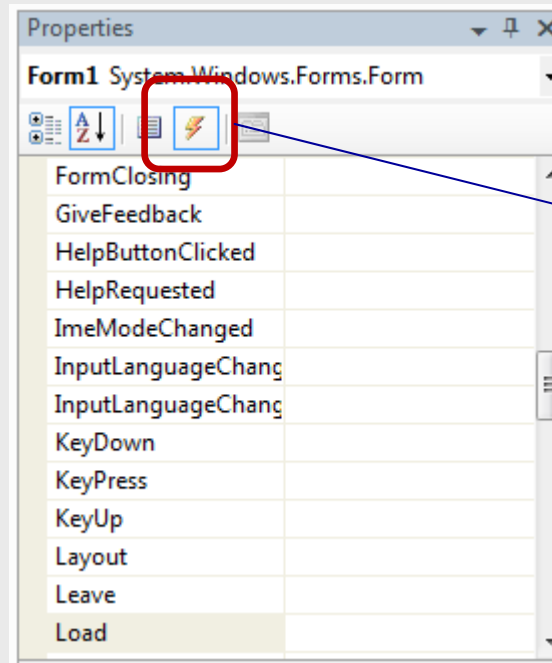
- inicijalno se postavljaju prilikom dizajna (u *Property Window*),
- neka mogu biti i dinamički promijenjena, pri izvođenju programa

❑ Inicijalno postavljena svojstva generiraju kod u `InitializeComponent()`


Svojstva



Događaji



Događaji

- ❑ **Mehanizam kojim razred dojavljuje klijentima tog razreda da se nešto događa s objektom razreda.**
 - tipična primjena je interaktivno grafičko sučelje, ali su događaji korisni i šire
 - između ostalog, povećava se modularnost programa
- ❑ **Primjeri događaja: klik mišem na gumb, unos teksta u textbox, učitavanje/zatvaranje forme, ...**
- ❑ **Rukovatelj događajem (*Event handler*)**
 - Postupak koji obrađuje događaje
 - Uobičajenog naziva *ControlName_EventName*
 - Argument su dvije reference: **object**, **EventArgs**
- ❑ **Primjer:  GUI\JednostavnaForma**

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

Događaji

❑ Pridruženi delegati

- delegat – referenca na postupak (isto što i pokazivač na funkciju u C/C++)
- sadrže listu referenci na postupke
- postupci se registrišu 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



Vrste 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, pa 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 hiperlinkova (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

□ Način prikaza

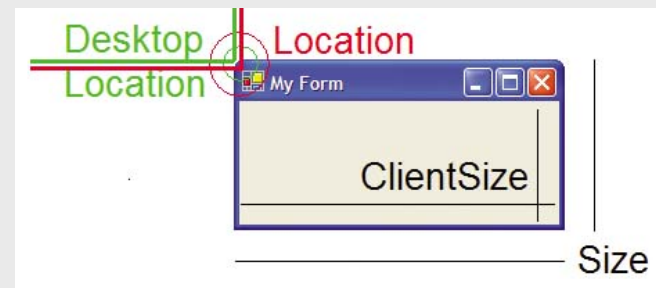
- `Show` – nemodalni prikaz
 - moguće je kliknuti na neku od više otvorenih formi
- `ShowDialog` – modalni prikaz
 - forma na vrhu ne dozvoljava da se aktiviraju donje
- `Hide` – sakriva formu

□ 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)

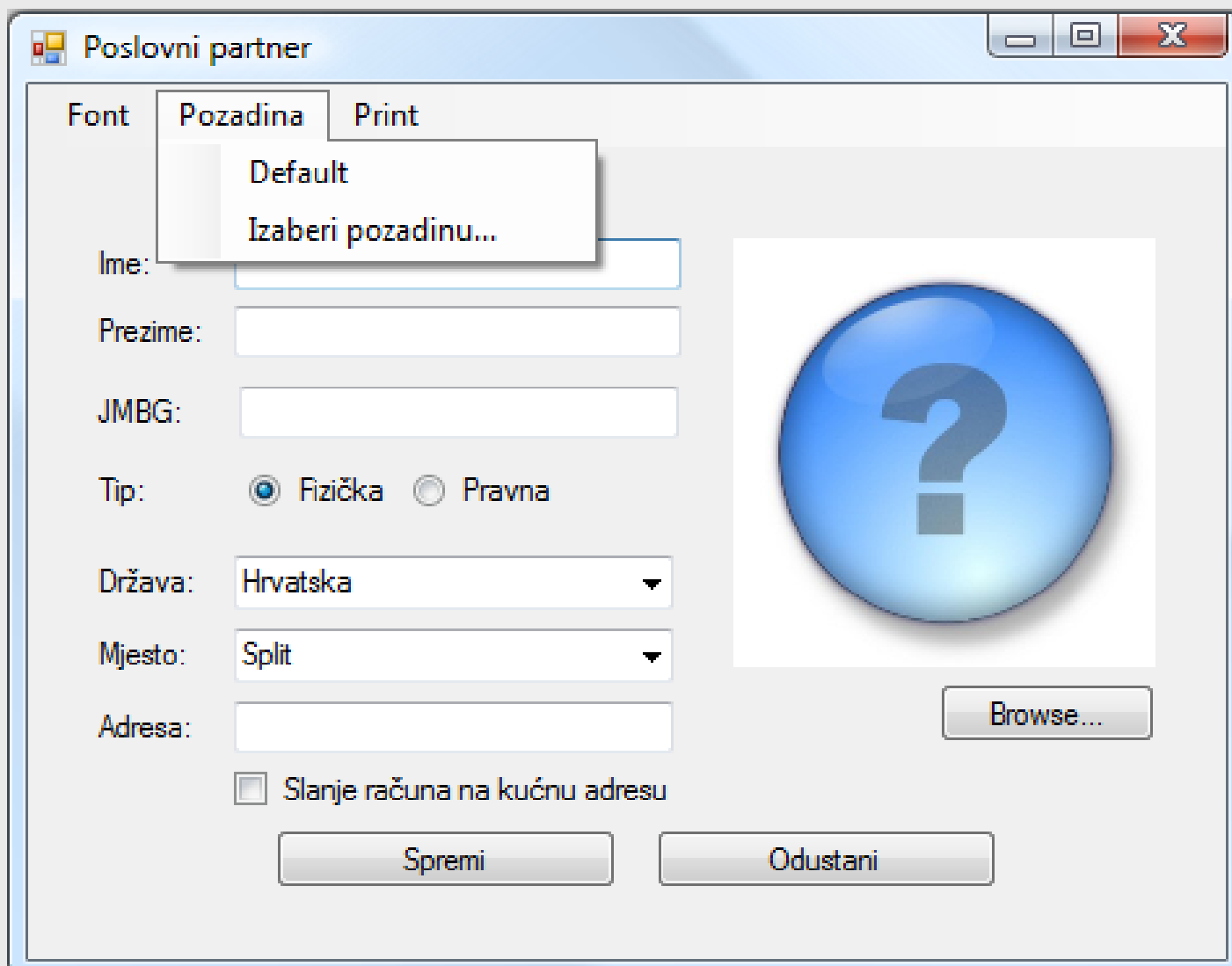
□ Događaji

- `LocationChanged`
- `SizeChanged`



Izrada zaslonske maske za unos podataka

□ Primjer GUI\Partner



Poslovni partner

Font Pozadina Print

Default
Izaberi pozadinu...

Ime:

Prezime:

JMBG:

Tip: ☒ Fizička ☐ Pravna

Država:

Mjesto:

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
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
ClearSelected	briše sve slektirane 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;  
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

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

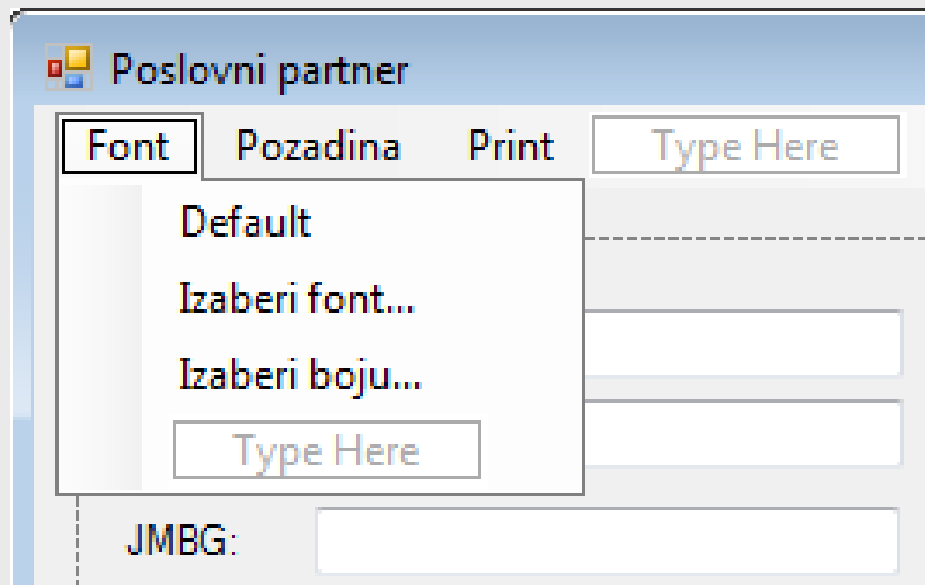
■ `ToolBarButton` kontrola

❑ **Contextbar** – izbornik zavisan o kontekstu

Primjer izbornika

❑ Primjer GUI\Partner

- Izgled izbornika u dizajnu
- Korijen sustava izbornika (MenuStrip) i stavke (ToolStripMenuItem)



❑ Događaji izbornika

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

```
System.Windows.Forms.MenuStrip menuStrip;  
System.Windows.Forms.ToolStripItem izaberiFontToolStripMenuItem;  
  
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*)

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 (Form1.resx)**

- spremište stringova, datoteka, slika, ... koje upotrebljava cijela aplikacija
 - » više o resursima naknadno

```
System.ComponentModel.ComponentResourceManager resources = new
    System.ComponentModel.ComponentResourceManager(typeof(Form1));
...
this.pictureBoxSlika.Image = ((System.Drawing.Image)
    (resources.GetObject("nepoznato")));
```

MessageBox

❑ Javni razred MessageBox

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





- `public static DialogResult Show(string);`
- `public static DialogResult Show(string, string);`
- `public static DialogResult Show(string, string, MessageBoxButtons);`
- `public static DialogResult Show(string, string, MessageBoxButtons, MessageBoxIcon);`

❑ MessageBoxButton i MessageBoxIcon

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

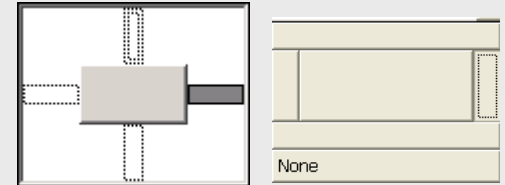
❑ enum DialogResult

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

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 prilikom izvođenja



❑ Zadatak

- Proučiti svojstva za dinamičko razmještanje kontrola (`Anchor`, `Dock`, `DockPadding`, `Location`, `Size`, `MinimumSize`, `MaximumSize`). Dopunite primjer Partner tako da forma “lijepo izgleda” kada se mijenja veličina.

Anchor	sidrenje rubova kontrole prema rubu spremnika: <code>Top</code> , <code>Left</code> , <code>Bottom</code> , <code>Right</code> oavlja se dinamički kada se mijenja veličina spremnika, po mogućnosti kontrola mijenja veličinu
Dock	Automatsko prislanjanje kontrole prema rubu 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



Domaća zadaća

☐ Domaća zadaća – nefunkcionalni prototip

- Organizacija grafičkog sučelja za projekt
- Svaki član treba oblikovati zaslonsku masku prozor za unos podataka temeljem scenarija prepoznatog pri izradi zapisnika s intervjua
- Svaki član treba isporučiti izvorni kod nefunkcionalne zaslonske maske

☐ Zadaća je bila zadana i isporučena kao dio 1.ciklusa !

Validacija unosa podataka

❑ Događaji kontrole

- `Validating (object sender, CancelEventArgs e)` – okida u trenutku kad kontrola treba biti napuštena
 - postavljanjem `e.Cancel = true` zabranjuje se napuštanje kontrole
 - nezgodno, jer nije moguće poduzeti nikakvu akciju sve dok se pogreška ne popravi (npr. nije moguće zatvoriti formu)
- `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

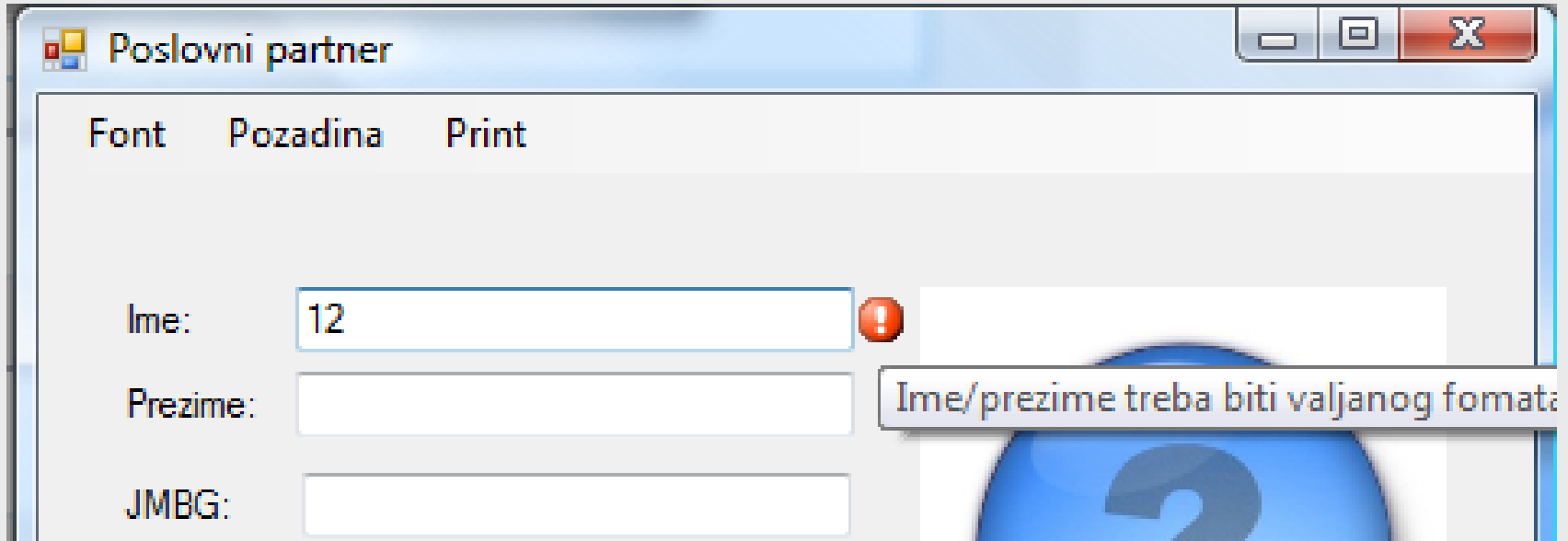
❑ Komponenta `ErrorProvider` – poruka o pogrešnom unosu

- ikona uskličnika se prikazuje dok unos nije valjan
- `SetError(Control, string)` – postavlja poruku o pogreški na 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
 - ...



Poslovni partner

Font Pozadina Print

Ime: 12

Prezime:

JMBG:

Ime/prezime treba biti valjanog fomata

ErrorProvider

❑ **ErrorProvider** – prikaz informacije o pogrešci

■ Svojstva

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

■ Postupci

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

Primjer validacije pri unosu u zaslonsku masku

❑ 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 {  
        errorProvider.SetError(textBox, "");  
        return true;  
    }  
}
```

Validacija forme za unos poslovnog partnera

❑ 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;
    }
    else {
        e.Cancel = true;
    }
}
```

❑ Klikom na gumb *Spremi* pozivamo postupak za validaciju

- Ako je validacija kontrola uspješno prošla, `this.ValidateChildren()` vraća *true*

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

Nasljeđivanje formi

❑ Jednostavno nasljeđivanje formi

```
public class Form2 : Form1
{
    public Form2()
    {
        this.Text = this.Text + " Inherited";
        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

Dinamičko kreiranje kontrola

❑ Primjer GUI\KreiranjeKontrola

■ Kreiranje gumba klikom na formu

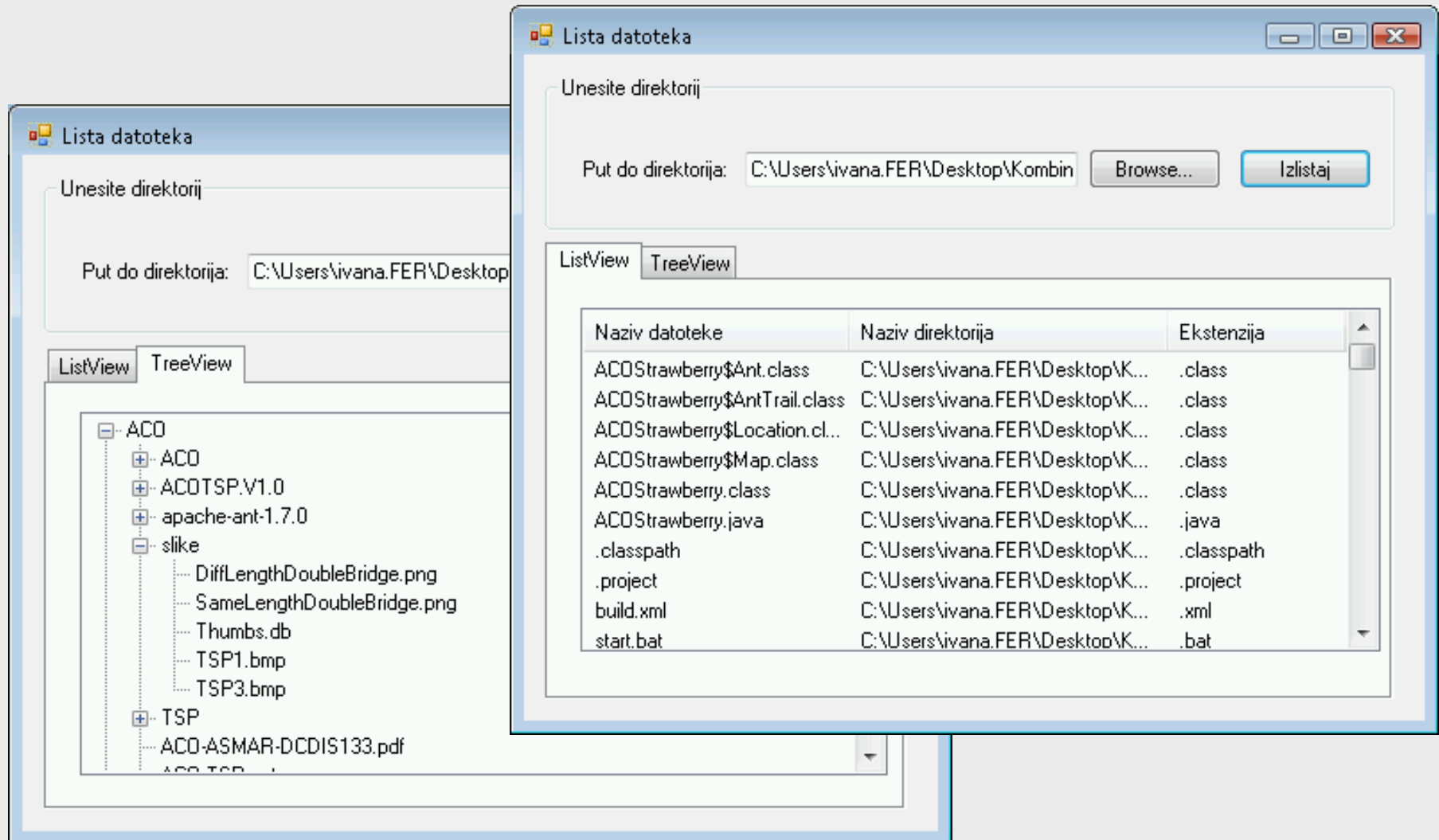
```
private void Form1_Click(object sender, EventArgs e)
{
    Button b = new Button();
    b.Text = "Gumb " + this.Controls.Count;
    Point mousePoint = PointToClient(MousePosition);
    b.Location = new Point(mousePoint.X, mousePoint.Y);

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

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

Lista datoteka

❑ Primjer GUI\ListaDatoteka



TabControl

❑ **TabControl** - Kontrola s tabulatorima (jahačima)

■ 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 podešavanja širine tabova (Normal, Fixed, FilledToRight)
- TabPages – kolekcija tabova
- TabCount – broj tabova
- SelectedIndex – indeks odabranog taba

■ Događaj

- SelectedIndexChanged – promjena svojstva SelectedIndex

❑ **TabPage**

■ Svojstva # otprije poznata

- AutoScroll, BackgroundImage, ImageIndex, Text

Lista datoteka

❑ **TabControl kontrola**

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

❑ **Rekurzivna obrada hijerarhije datoteka u poddirektorijima**

- Korišćenje 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

Lista datoteka

❑ 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
- Sorted – sortirani čvorovi
- 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

Lista datoteka

❑ 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);

}
```

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

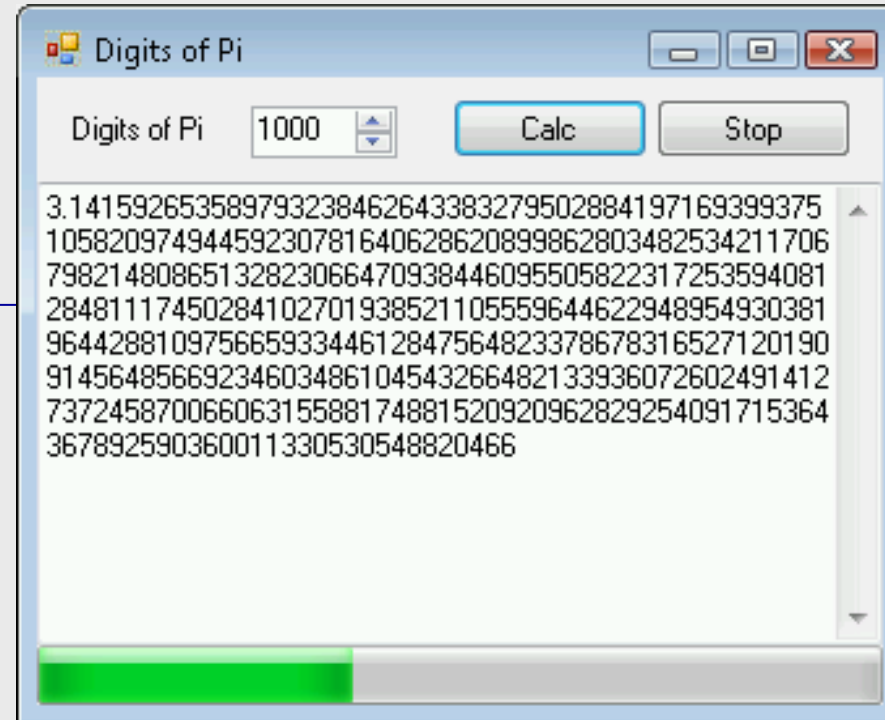
Primjer višenitnog programa

❑ 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
    }
}

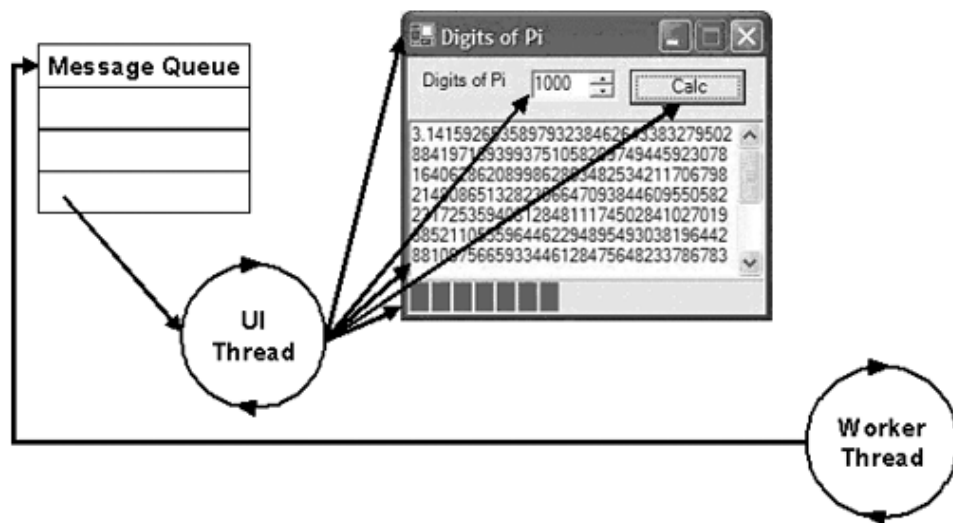
void buttonCalc_Click(object sender, EventArgs e)
{
    CalcPi(digitsToCalc);
}
```



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)
    { // Ponovni poziv istog postupka, ali iz glavne niti
        ShowProgressDelegate showProgress = new
            ShowProgressDelegate(ShowProgress);
        this.BeginInvoke(showProgress,
            new object[] { pi, totalDigits, digitsSoFar });
    }
    else {...} //kao prije
}
```

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
- Način izrade različitih vrsta naveden je u nastavku

❑ *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

Kreiranje vlastite kontrole

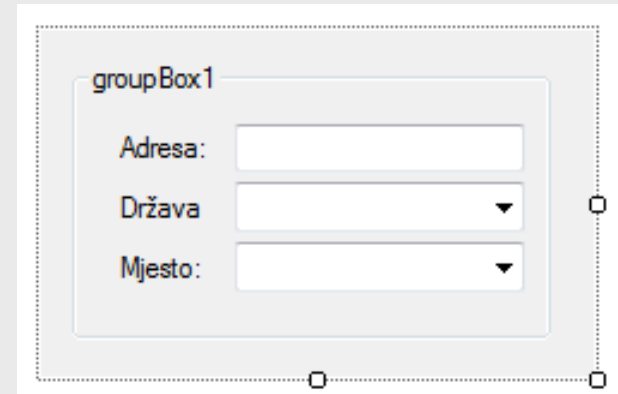
❑ **Windows Control Library projekt**

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

❑ **Vlastita kontrola nasljeđuje razred UserControl**

- `public partial class UserControlAdresa : UserControl`

❑ **Vlastitu kontrolu stvaramo dodavanjem postojećih kontrola**



❑ **Primjer GUI\Adresa**

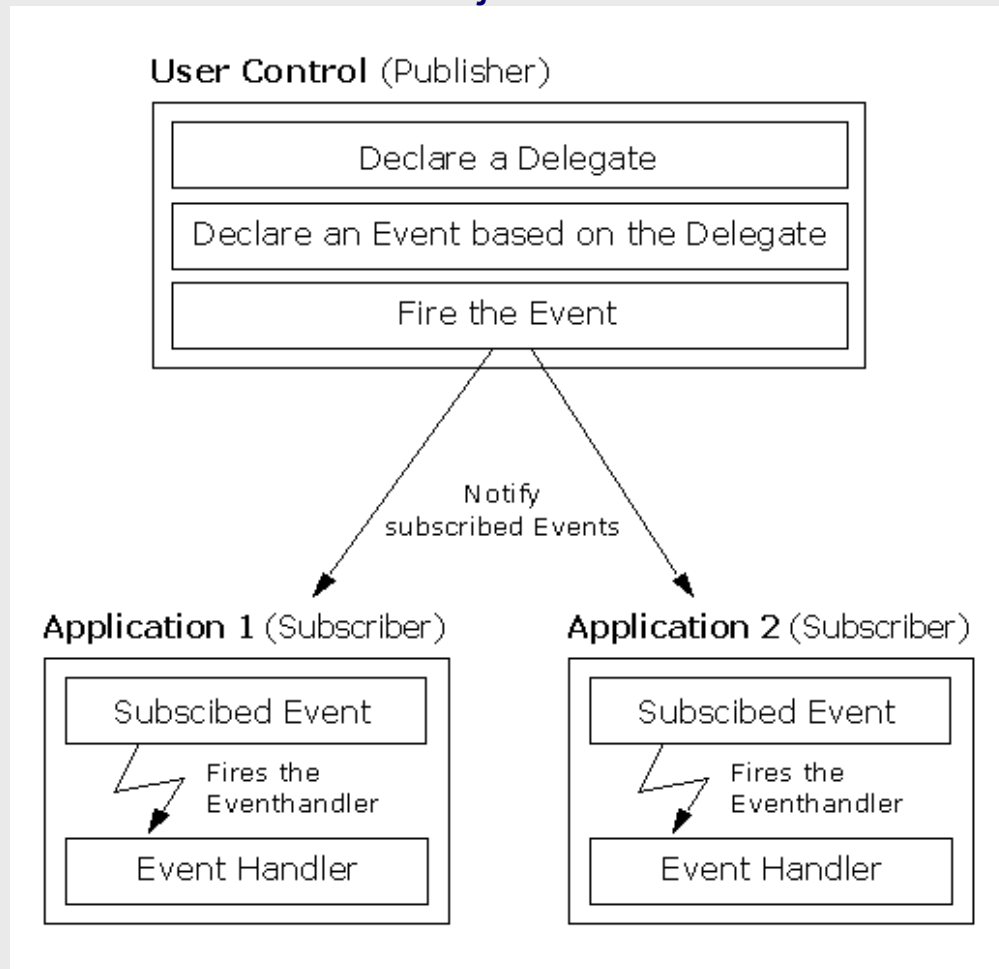
- Implementiramo svojstva vlastite kontrole kojima pristupamo iz aplikacije

```
public int DrzavaSelectedIndex
{
    get { return comboBoxDrzava.SelectedIndex; }
    set { comboBoxDrzava.SelectedIndex = value; }
}
```

Događaji na vlastitoj kontroli

❑ Model izdavač-pretplatnik

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



Primjer vlastite kontrole

☐ Obrada događaja na vlastitoj kontroli

☐ Rukovatelj događajem

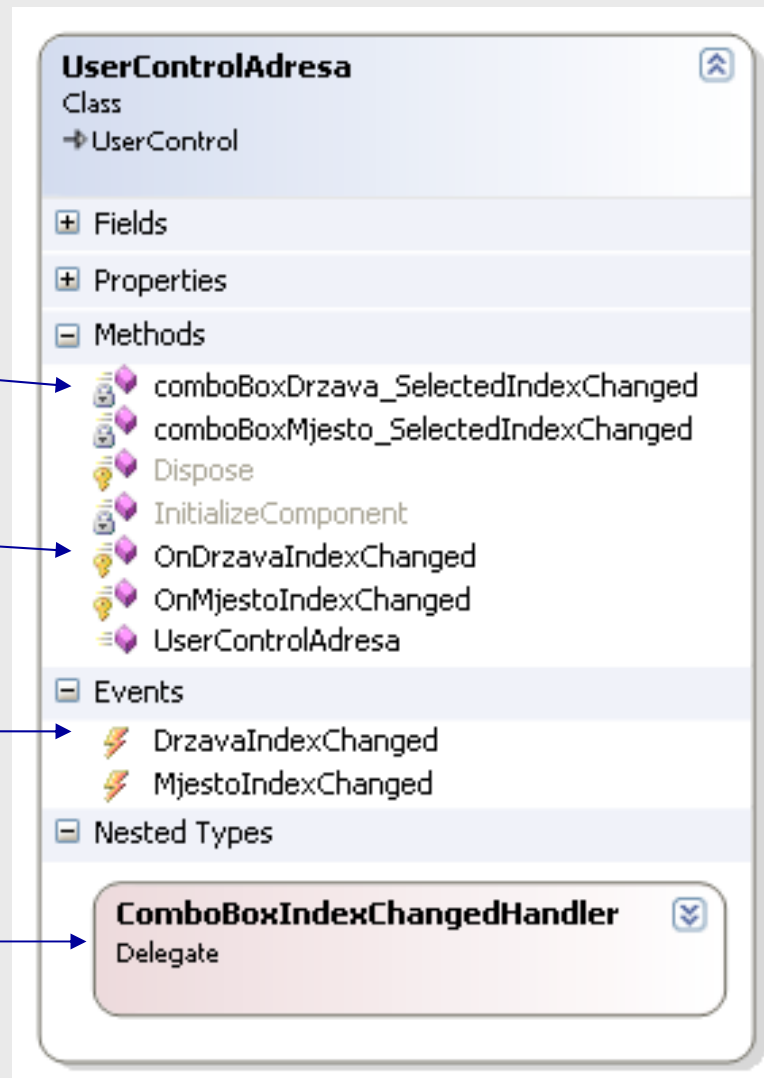
- lokalno na vlastitoj kontroli

☐ Postupak koji obrađuje događaj

☐ Događaj pridružen delegatu

- vidljiv aplikaciji koja koristi kontrolu

☐ Delegat



Primjer vlastite kontrole

❑ Delegat i događaj pridružen delegatu

- primjer, promjena indeksa u padajućoj listi

```
public delegate void ComboBoxIndexChangedHandler();  
public event ComboBoxIndexChangedHandler DrzavaIndexChanged;
```

❑ Postupak koji će obraditi događaj


- primjer, provjera postoji li pretplatnik na događaj i okidanje događaja

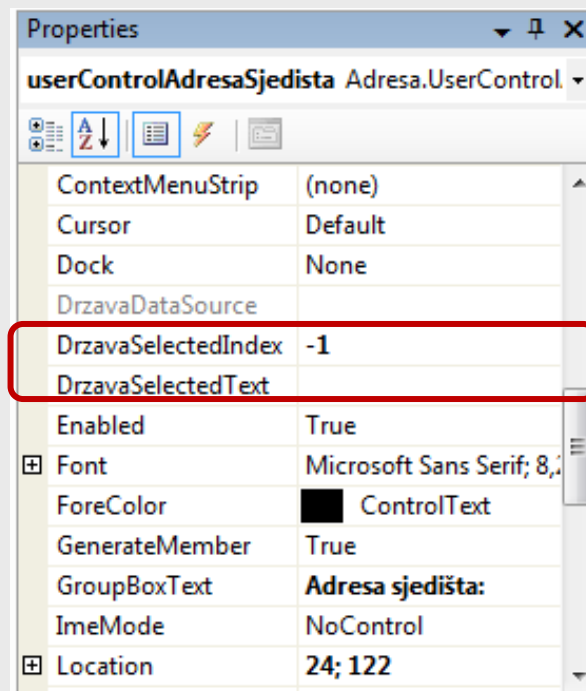
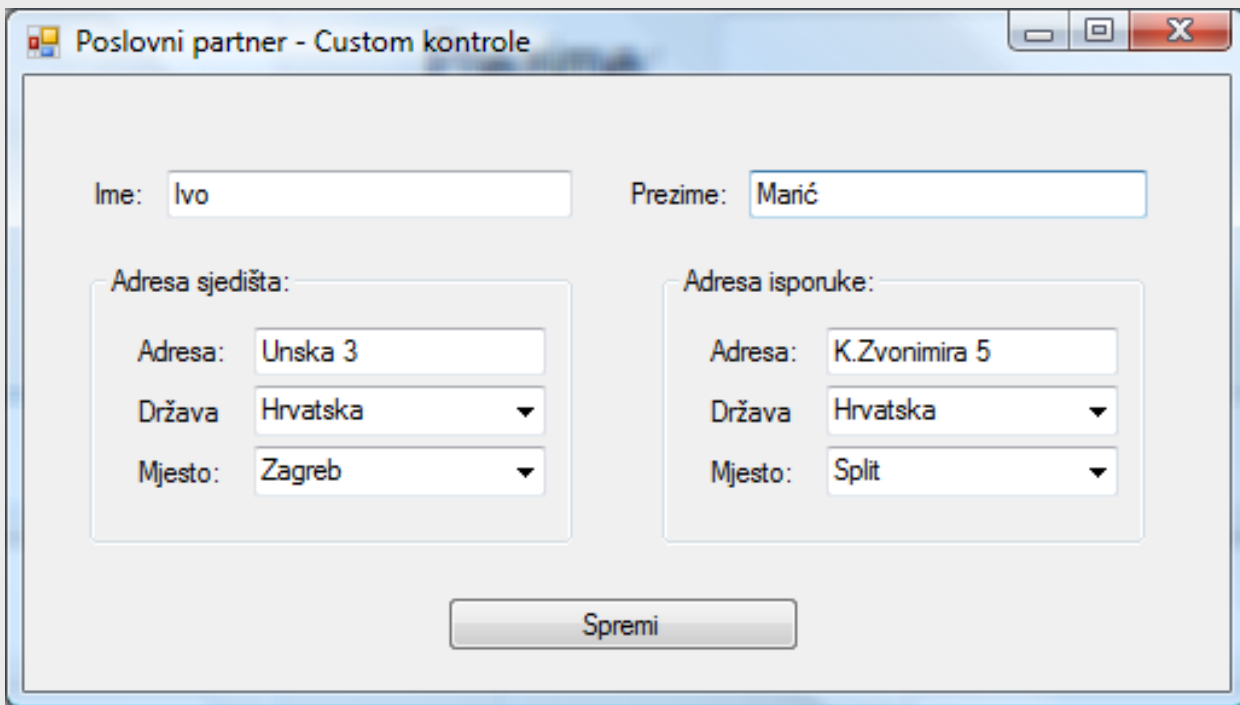
```
protected virtual void OnDrzavaIndexChanged()  
{  
    if (DrzavaIndexChanged != null) DrzavaIndexChanged();  
}
```

❑ Rukovatelj događaja za promjenu indeksa u padajućoj listi

```
private void comboBoxDrzava_SelectedIndexChanged(  
    object sender, EventArgs e)  
{  
    OnDrzavaIndexChanged();  
}
```

Upotreba vlastite kontrole

- ❑ Dodavanje vlastite kontrole (Adresa.dll) u projekt,  GUI\Custom
 - Desni klik na *Toolbox* -> *Choose Items* -> *Browse...*
 - Ili, desni klik na projekt – *Add reference* - *Project* - *Adresa*
- ❑ Dodamo kontrolu i postavljamo joj svojstva
 - Na popisu su i svojstva koja smo sami definirali

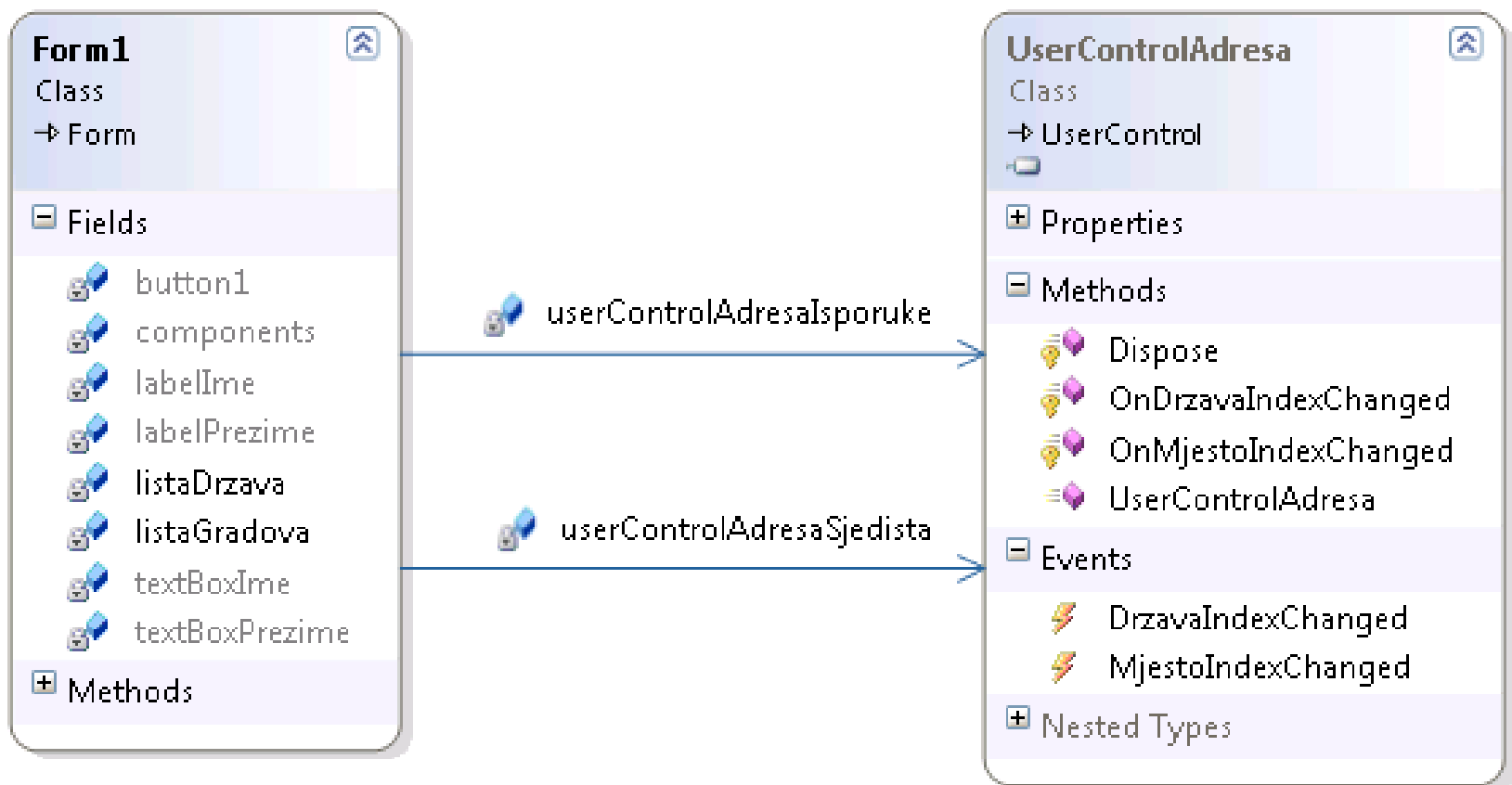


Properties	
userControlAdresaSjedista Adresa.UserControl	
ContextMenuStrip	(none)
Cursor	Default
Dock	None
DrzavaDataSource	
DrzavaSelectedIndex	-1
DrzavaSelectedText	
Enabled	True
Font	Microsoft Sans Serif; 8,;
ForeColor	ControlText
GenerateMember	True
GroupBoxText	Adresa sjedišta:
ImeMode	NoControl
Location	24; 122

Upotreba vlastite kontrole

□ Dijagram razreda za primjer GUI\Custom

- Pretplatnik Form1 i izdavač UserControlAdresa
- Svojstva `userControlAdresaIsporuke` i `userControlAdresaSjedista` prikazana su kao asocijacija



Upotreba vlastite kontrole

❑ Primjer obrade događaja vlastite kontrole

```
private void userControlAdresaSjedista_DrzavaIndexChanged()  
{  
    userControlAdresaSjedista.MjestoDataSource =  
        listaGradova[userControlAdresaSjedista.DrzavaSelectedIndex];  
}
```

❑ 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

```
userControlAdresaSjedista.DrzavaBindingContext = new BindingContext();  
userControlAdresaIsporuke.DrzavaBindingContext = new BindingContext();
```



Zadaci

❑ Domaća zadaća 2.1

- Svaki član oblikuje jednu zaslonsku masku za unos podataka iz vlastitog problemskog područja.
 - Ugraditi svojstva, postupke i obradu događaja, uključujući provjeru valjanosti podataka (validaciju).
 - Po potrebi napraviti vlastitu kontrolu (User control).

❑ Zadatak projekta

- Uspostava izbornika aplikacije
- Ugraditi pozive pojedinačnih zaslonskih maski


Zadaci za vježbu

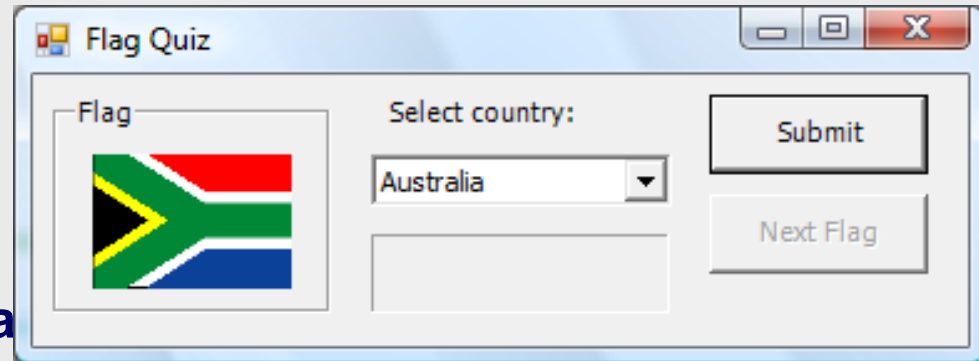
❑ Napraviti formu za uređivanje slike

- 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”

Automatizirano povlačenje i ispuštanje

❑ 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

❑ 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

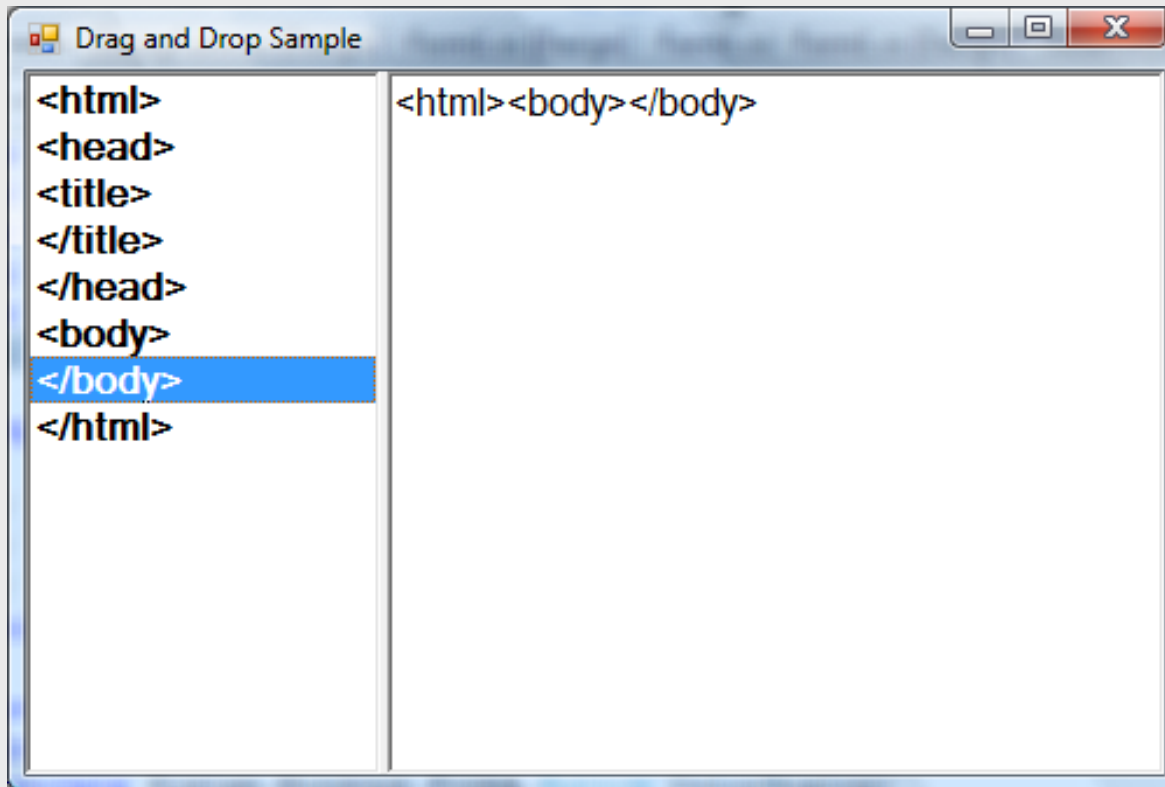
❑ Članovi razreda DragEventArgs

- 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

Zadaci za vježbu

❑ 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



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)

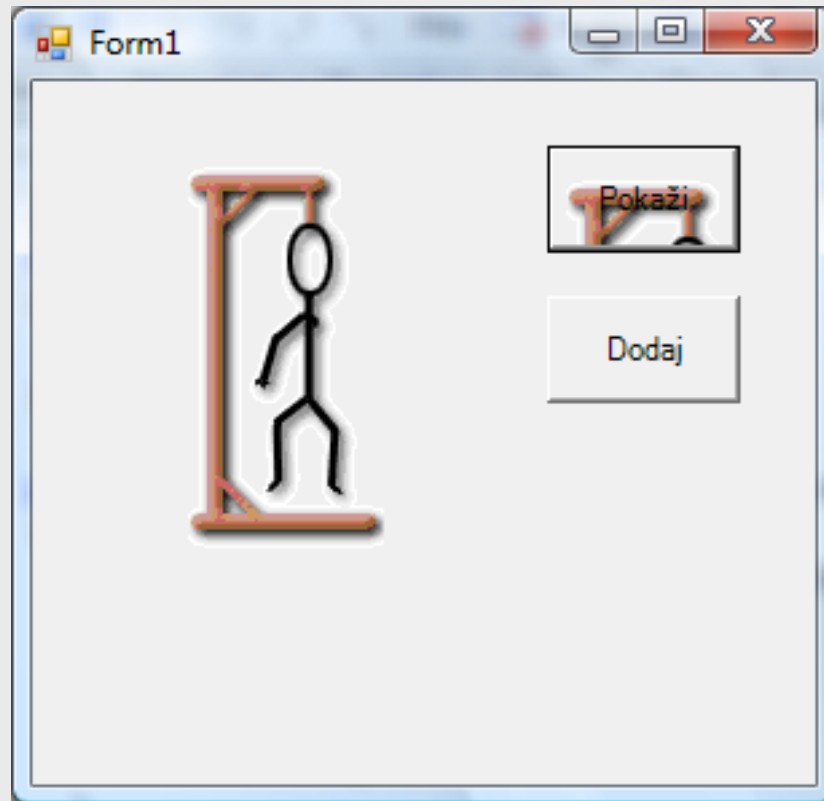
❑ Dodavanje bitmape programski:

- `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



Reference

Resursi

- csharp_ebook

Dodatna literatura:

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