

Objekti u Toolbox-u



UVOD

Uvod

Pomoću *dot net* okvira .NET Framework, jezik Visual C# omogućuje širok opseg predefinisanih objekata koji se mogu koristiti da se kreira neka Windows aplikacija. Ovi objekti su deo Toolbox-a. U ovom predavanju ćemo analizirati tj. ilustrovati upotrebu objekata u Toolbox-u, naime predefinisane objekata raspoložive u Toolbox-u, koji se mogu iskoristiti da se složi neka Visual C# - aplikacija. Npr neki od sledećih objekata su raspoloživi u Toolbox-u: Button, Label, TextBox, PictureBox, CheckBox, RadioButton, GroupBox, ListBox, RichTextBox, ToolBar, itd. Dakle, ovaj niz predefinisanih objekata iz Toolbox-a se omogućuje preko **.NET Framework**, što je slično sa Java libraries. Ovi objekti se često nazivaju „*controls*“, tj. „kontrola“.

U prethodnim lekcijama videli smo da su objekti okarakterisani preko Properties, tj. karakteristika tj. osobina navedene u Properties Window, odnosno Properties Editor. A ove karakteristike definišu kako objekti izgledaju. Neka osobina, *property*, je nešto što objekat ima, ali potrebno je da analiziramo i metode Toolbox objekata, a metode objekata su nešto što opisuje šta objekti rade. Toolbox je prozor u Visual C# koji sadrži pojedine komponente neke grafičke aplikacije, npr. „dugme“, ili „tekstboks“, ili „nalepnica“, itd. U ovom predavanju će biti detaljno analizirani elementi u Toolbox-u, kao i njihove metode.

Ključna pitanja:

Koji objekti postoje u Toolbox-u?

Koje metode Toolbox objekata postoje?

Primer - toolbox

01

PRIMER - TOOLBOX

Evo primera gde se koristiti objekat Label u Toolbox-u, da se doda neki tekst na „formu“:

Dole na slici je prikazan kako izgleda **Toolbox**. I njene komponente, tj. objekti u **Toolbox**-u (tzv. „kontrolne“): Pointer, Label, LinkLabel, Button, TextBox, itd. Podsetimo se primera, gde se koristiti objekat **Label** u Toolbox-u, da se doda neki tekst na „formu“:

Korak 1 Klikne se na ikonu Label(tj. „Nalepnica“).

Korak 2 Pomoću *Mouse pointer*-a, Miš-pokazivača, i pritiskanja levog dugmeta Miša, može se na Form-u postaviti levi gornji ugao Nalepnice, tj. objekta Label. Držeći pritisnuto dugme, poveća se nalepnica po želji, i onda se otpusti dugme. Ako treba, objekat Label se može pomeriti, ili promeniti njena veličina. Nalepnica dobija automatski predefinisano ime, *default name*, „label1“.

Na ovaj način smo na „formu“ (C# -formu) postavili na željenom mestu tekst „label1“ željene veličine. Ali ovo je najjednostavniji slučaj korišćenja objekata u Toolbox-u. Medjutim, mogu se formirati mnogo složenije C# - forme od ove. Npr. dole je prikazana „forma“ sa dva **Textbox**-a (dva „tekstboks“) i jednim Button (jednim „dugmetom“). Kreiranje ove složenije forme je potpuno slično sa formiranjem prethodne jednostavne forme, s tim što se postavlja objekat po objekat na „formu“, npr. prvo „dugme“ pa onda prvi pa onda drugi „tekstboks“. Dole je takodje dat izgled Toolbox-a u okviru Visual 2010 Express.

FORMA SA DVA „TEKSTBOKSA“ I JEDNIM „DUGMETOM“

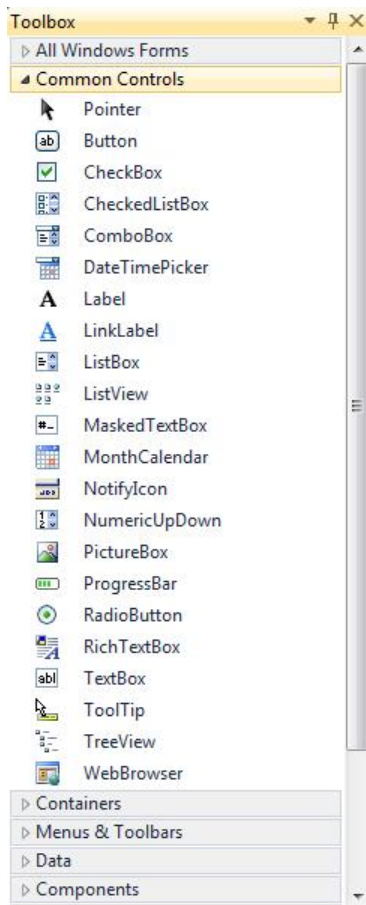
Na slici je forma sa imenom Form1 i sa dva „tekstboksa“ i jednim „dugmetom“.



Sl.1: forma

TOOLBOX WINDOW

Slika prikazuje Toolbox window.



Slika2: toolbox

PRIMER-PROPERTIES

Mogu se menjati ove karakteristike, properties, pomoću Properties Window. Karakteristike su npr. Font, Color, Size, Text, itd.

Property Editor tj. Properties Window (prozor Properties) prikazuje liste karakteristika, properties, za pojedine objekte u Toolbox-u, tj. za pojedine „kontrolne”, gde su objekti Label, Button, TextBox, itd, izlistani u Toolbox. Mogu se menjati ove karakteristike, properties, pomoću Properties Window. Karakteristike su npr. Font, Color, Size, Text, itd. Dole je na crtežu prikazan Properties Window. Upotreba Properties Window je brza i jednostavna, mada zahteva prvo upoznavanje sa mogućnostima koje pruža. Ali, posle ovog upoznavanja, jednostavno je koristiti Properties Window.

Npr za objekat Label nazvan „label1“, može se promeniti naslov (tj. tekst) umesto Label1 u MyLabel, tako što će se kliknuti Text u Properties Window, i onda ukucati: MyLabel. Takođe, ako se klikne na Font, može se promeniti veličina slova, npr. uvećati veličina slova. Dalje, za „formu“ Form1 sa dva Textbox-a i jednim Button, može se na potpuno sličan način promeniti naslov da umesto „button1“ bude npr. „Copy“.

PRIMER SA KOPIRANJEM TEKSTA

Pogledajmo primer, da se može npr. napraviti „forma“ koja učitava tekst pomoću textBox1 i kopira taj isti tekst u textBox2.

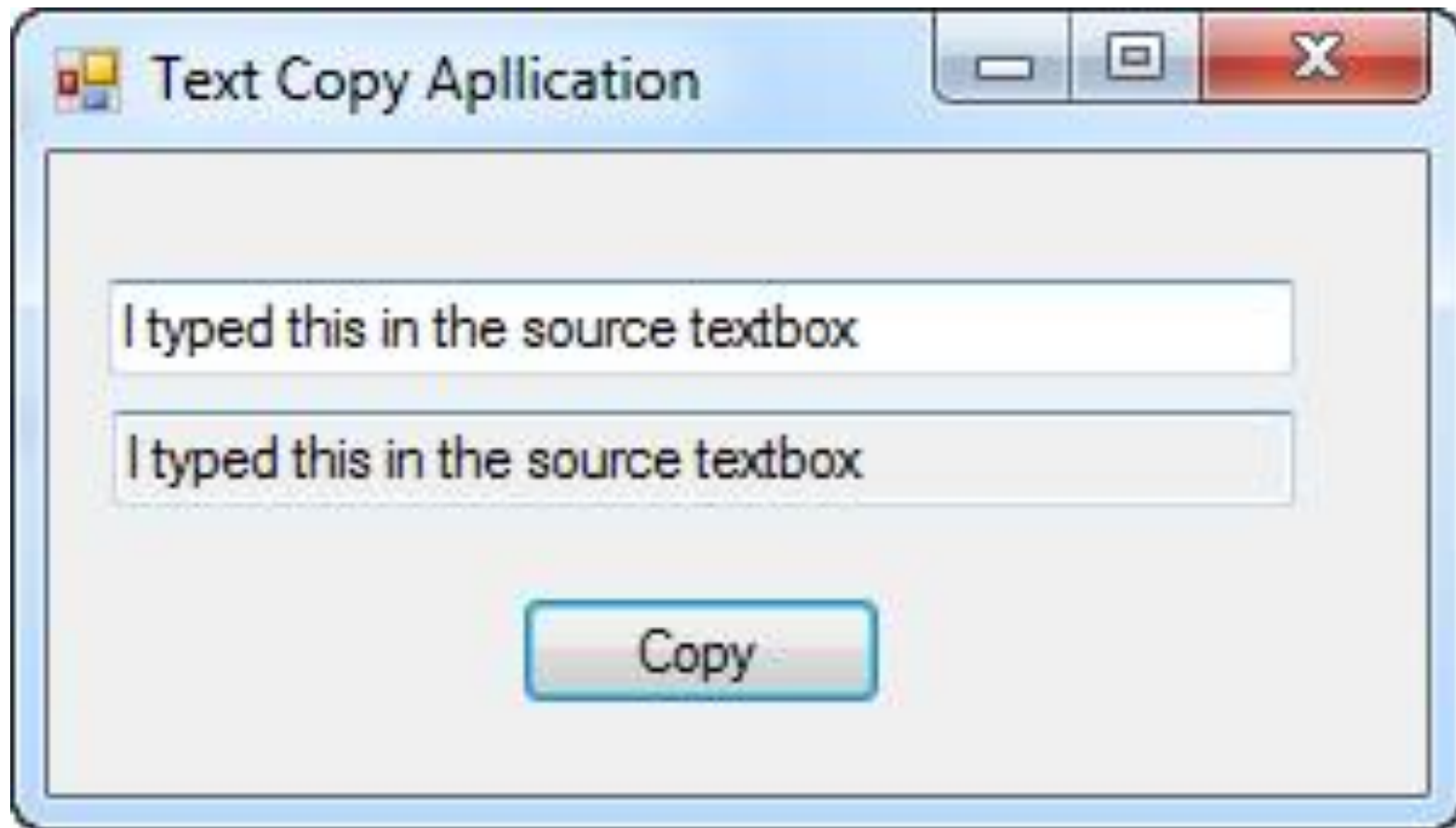
Pogledajmo primer, da se može npr. napraviti „forma“ koja učitava tekst pomoću textBox1 i kopira taj isti tekst u textBox2 po pritiskanju dugmeta *button1*. To je dole ilustrovano, gde se vidi forma nazvana **Text Copy Application**, i ova „forma“ je prikazana kako izgleda posle učitavanja teksta i pritiskanja dugmeta. Ali, prethodno je potrebno u *Code editor-u* dodati liniju koda:

```
textBox2.Text = textBox1.Text;
```

za *button1_Click()* metodu, što je dole prikazano na slici.

„FORMA“ NAZVANA TEXT COPY APPLICATION POSLE PRITISKANJA DUGMETA

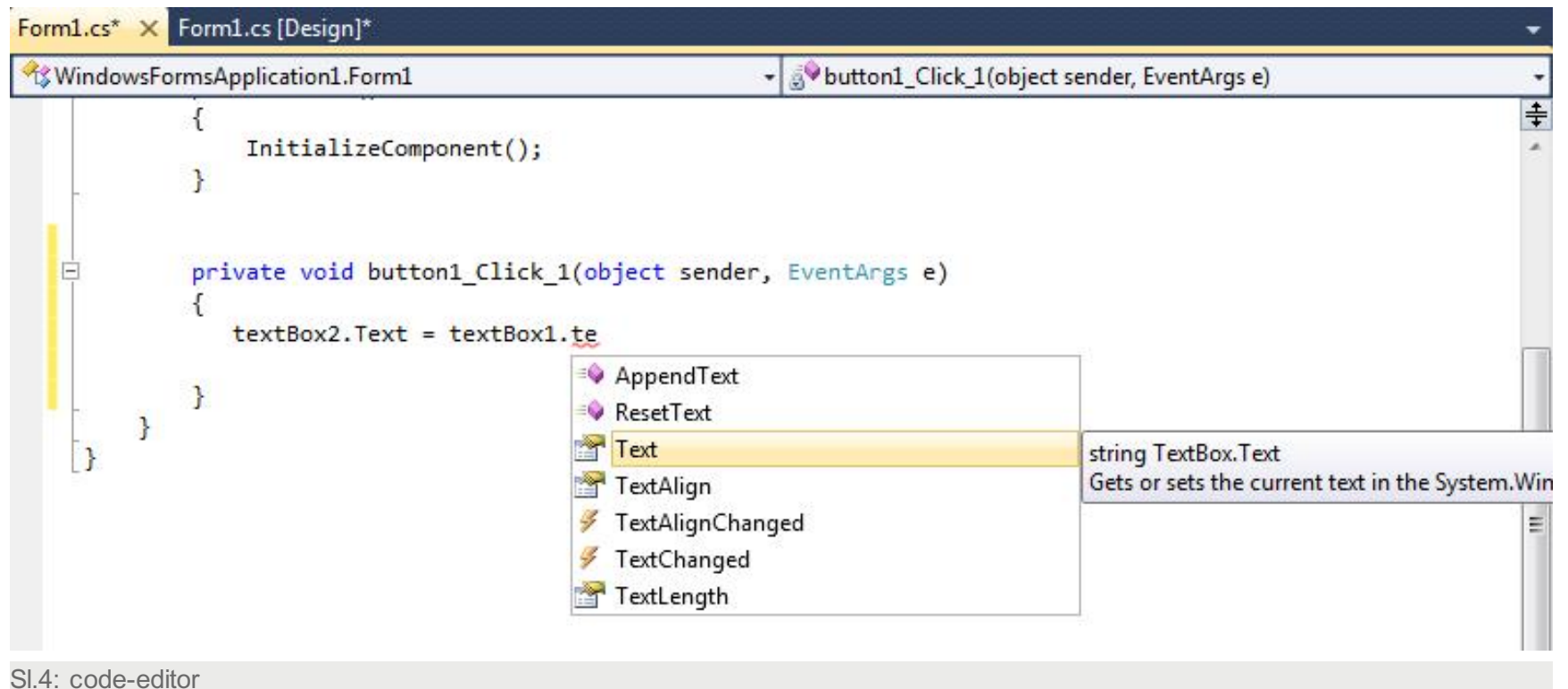
Slika ilustruje Text Copy Application posle pritiskanja dugmeta.



Sl.3: forma

CODE EDITOR SA DODATOM LINIJOM KODA: TEXTBOX2.TEXT = TEXTBOX1.TEXT;

Na slici vidimo Code editor sa dodatom linijom koda: textBox2.Text = textBox1.Text;



Metode Toolbox objekata

02

METODA TOOLBOX OBJEKTA

Objekti u Toolbox-u imaju čitav niz predefinisanih metoda, kje treba poznavati da bi se koristile.

Objekti su takodje okarakterisani ne samo pomoću **Properties**, već i preko:

Events,

Actions.

Šta je to metoda objekta, i koji je to koncept koji koristi metode objekata? Jednostavno se može objasniti koncept metoda objekata, naime, karakteristike objekata su nešto što objekti poseduju, a metode objekata su nešto što objekti rade tj. Izvršavaju. Posmatrajmo neki fizički oobjekt, npr. Auto, I taj auto ima osobine boje, starosti, maksimalne brzine, itd. A npr. Pokretanje i Zaustavljanje auta ili Trubljenje ili Kočenje su ono što auto radi, tj. akcije objekta koje se obavljaju u odredjenom periodu. I za auto se mogu znači definisati metoda Pokretanja, metoda Zaustavljanja, itd. koje opisuju akcije objekata.

Po pravilu, metode zahtevaju dodatne informacije, koje se zovu parametri metode, parametri metode se upisuju iza imena metode objekta, i to se upisuju u zagradama, i ovi parametri definišu ulazne podatke koji specificiraju kako će se akcija obavljati. Npr, auto može da se zaustavi ili vrlo brzo ili vrlo sporo. Parametri metode se upisuju dakle u zagradama iza imena metode, ali i ako nema potrebe za parametrima neke metode, tj. neke metode su bez parametara, ali ipak se stave dve prazne zagrade iza imena metode, da bi se videlo da se radi o metodi.

Objekti u **Toolbox**-u imaju čitav niz predefinisanih metoda, kje treba poznavati da bi se koristile. Npr. metoda **Hide()**.

PRIMER

Objekt Label, Nalepnica, ima metodu Hide().

Objekt **Label**, Nalepnica, ima metodu **Hide**, sakriti, koji omogućuje da se sakrije tj. napravi nevidljivom, kao i metodu **Show**, pokazati, koji omogućuje da se prikaže tj. čini je vidljivom. Da bi to ilustrovali, treba učiniti sledeće:

1. Postaviti Label i Button na C# - formu.
2. Za objekat Button i događaj Click, treba otkucati instrukciju:
label1.Hide();
3. Izvršiti program tj. aplikaciju, i kliknuti dugme

Na sličan način moglo bi se postaviti drugo dugme, *button2*, na C# - formu, dugme koje izvršava

```
label1.Show();
```

u cilju da bi se ponovo prikazala nalepnica, koja je prethodno sakrivena. Dole je na crtežu prikazan primer sakrivanja nalepnice.

U **Help** dugmetu u **Menu bar**, može se za svaki Toolbox objekat naći lista

- Properties,
- Events,
- i Methods.

C# - FORMA SA DVA DUGMETA I JEDNOM NALEPNICOM

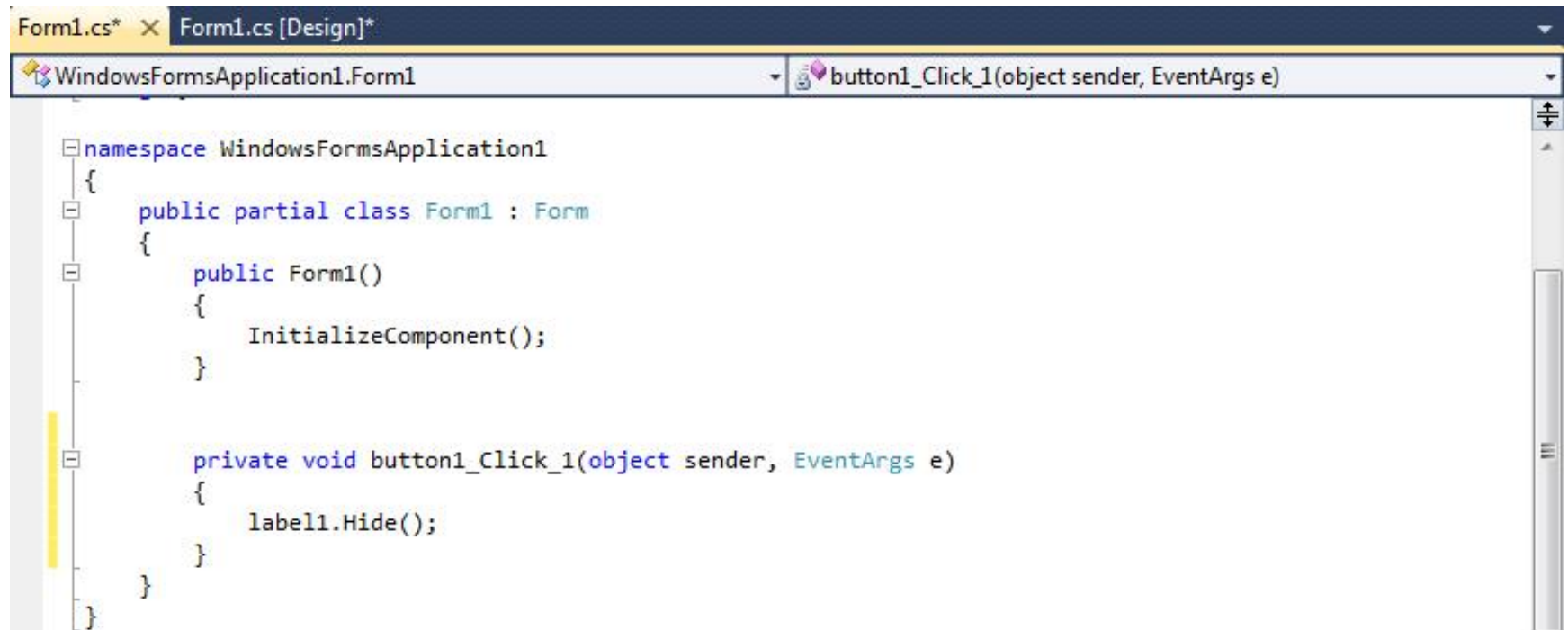
Na slici je C# - forma sa dva dugmeta i jednom nalepnicom.



Slika1: forma

EDITOVANJE PROGRAMSKOG KODA U CODE EDITOR-U

Slika ilustruje editovanje programskog koda u code editor-u.



```
Form1.cs* x Form1.cs [Design]*
WindowsFormsApplication1.Form1 button1_Click_1(object sender, EventArgs e)

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            label1.Hide();
        }
    }
}
```

Slika2: code-editor

Upotreba objekta *Button*

03

BUTTON-PRIMER

*Objekat **Button**, tj. Dugme, je koristan kad se želi na lak način da se startuje neka operacija, potvrdi ili izbaci neki izbor, ili potraži pomoć.*

Može se postaviti drugo dugme, *button2*, na C# - formu, dugme koje izvršava `label1.Show()`; u cilju da bi se ponovo prikazala nalepnica, koja je prethodno sakrivena. Dole je prikazan programski kod za primer „otkrivanja“ nalepnice „label1“.

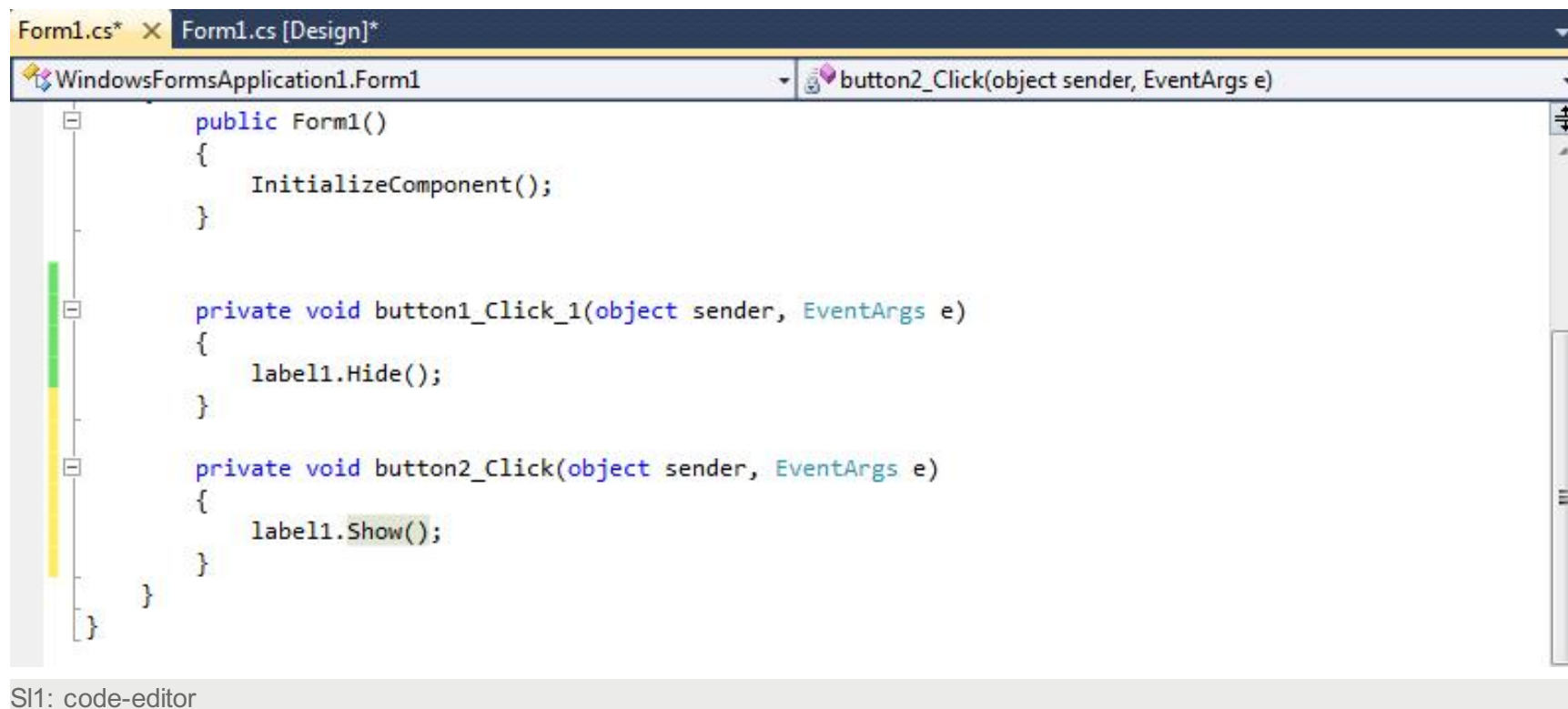
Objekat **Button**, tj. Dugme, je koristan kad se želi na lak način da se startuje neka operacija, potvrdi ili izbaci neki izbor, ili potraži pomoć. Po pravilu, upotreba **Button**-a se sastoji u zadavanju karakteristika, *properties*, i pisanja programskog koda za događaj `Kliknuti`,

Click.

Karakteristika, *property*, `Text` u **Property Editor**-u definiše koji tekst se pojavljuje na objektu **Button** tj. „Dugme“. A karakteristika **Font** omogućuje promenu tipa slova na naslovu objekta. Takođe, karakteristika `Image` omogućuje prikazivanje slika na dugmetu. Npr. pomoću *Text property*, jednostavno se može zadati tekst „**Click me**“ umesto teksta „button1“.

EDITOVANJE PROGRAMSKOG KODA U *CODE EDITOR-U*

Slika prikazuje editovanje programskog koda u code editor-u.



```
Form1.cs* x Form1.cs [Design]*
WindowsFormsApplication1.Form1 button2_Click(object sender, EventArgs e)

public Form1()
{
    InitializeComponent();
}

private void button1_Click_1(object sender, EventArgs e)
{
    label1.Hide();
}

private void button2_Click(object sender, EventArgs e)
{
    label1.Show();
}
}
```

SI1: code-editor

C# - FORMA SA OBJEKTOM „DUGME“ SA NASLOVOM “CLICK ME”

Nasluci je C# - forma sa objektom „dugme“ sa naslovom “Click me”.



SI1.2: forma

Toolbar

04

TOOLBAR

Toolbar, to je grupa dugmadi, buttons, grupisanih zajedno.

Toolbar, to je grupa dugmadi, buttons, grupisanih zajedno u vidu linije. **Primer:**

1 Postaviti Toolbar, RichTextBox i FontDialogBox na C# - formu.

2 Kliknuti Toolbar, i zatim u Properties-window otvoriti Button collection, i kliknuti **add** 6 puta

Upotreba objekta Label

05

OBJEKAT LABEL-PROPERTIES

*Može se uključiti slika na Nalepnici, Label, pomoću karakteristike **Image***

Objekat **Label** se koristi da prikaže predefinisani (prethodno-definisani) tekst koji nije potrebno editovati od strane korisnika aplikacije. Ali ipak omogućuje se i promena teksta objekta „nalepnica“ („Label“) pomoću karakteristike **Text**, **Text property** u **Property Editor-u**. Takođe, jednostavno se bira veličina i lokacija nalepnice na „formi“, što je opisano u poglavlju 2. Takođe, može se staviti nekoliko „nalepnica“ a ne samo jedna „nalepnica“, npr. „label1“ i „label2“.

Takođe, može se uključiti slika na Nalepnici, **Label**, pomoću karakteristike:

Image.

A pomoću karakteristike:

BackColor

može se menjati boja pozadine, što može da bude vrlo efektno pojavljivanje tog objekta.

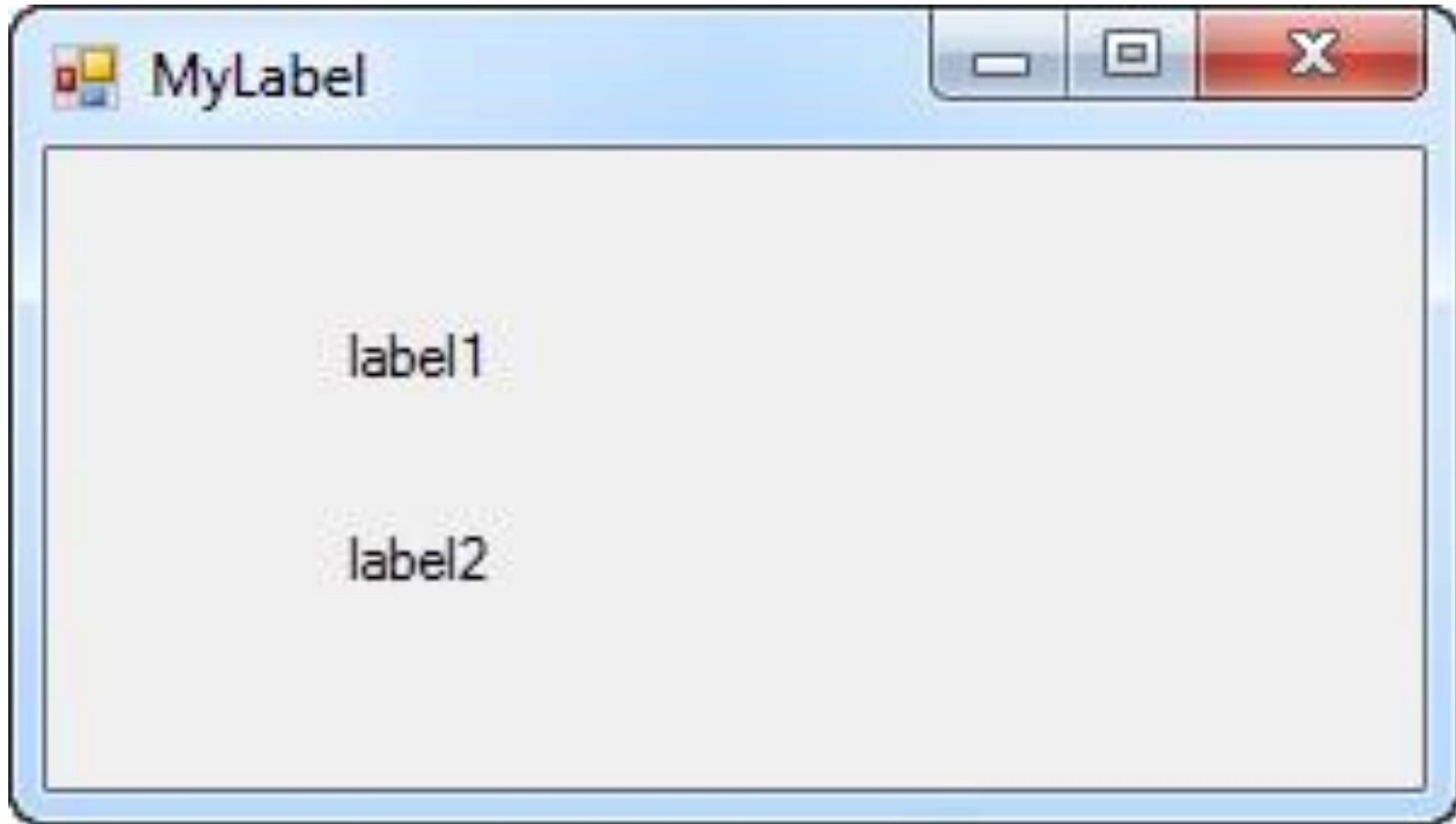
Konačno, karakteristika:

BorderStyle,

omogućuje zadavanje stila okvira Nalepnice.

C# - FORMA SA DVE „NALEPNICE“, „LABEL1“ I „LABEL2“

Slika prikazuje C# - formu sa dve „nalepnice“, „label1“ i „label2“.



Slika: forma

PRIMER SA BACKCOLOR

Evo primera sa „backcolor“.

Evo primera:

1. postavi 1 *label* objekt, 1 *textbox*, i 1 *button* na formu.
2. duplo kliknuti button, i uneti sledeće instrukcije:

```
string color = textBox1.Text.ToUpper();  
label1.Text = color;  
switch (color)  
{case "GREEN": label1.BackColor = Color.Green; break;  
case "YELLOW": label1.BackColor = Color.Yellow; break;  
//.....  
default: label1.Text = "Repeat";  
break;}
```

3. Izvršiti aplikaciju, ukucati tekst u *textbox*, i kliknuti button.

Pri tome, metoda `ToUpper()` prebacuje u velika slova.

INSTRUKCIJA SWITCH()

Instrukcija switch() je zgodnija od if() kada ima mnogo mogućih alternativa, ali nekad je zgodnija if() instrukcija jer je fleksibilnija od switch().

If() instrukcije su u mnogim slučajevima odlični za kreiranje grananja u programu, ali u nekim slučajevima, kada je mnogo alternativa tj. mnogo mogućih grananja, onda je elegantno primniti instrukciju **switch()**. Ova instrukcija *switch* omogućuje višestruko grananje u zavisnosti od vrednosti test varijable koja je tipa *int* ili *string* (dakle ne testira se vrednost logičkog izraza tj. rezultujuća logička varijabla, već se testira vrednost *int* ili *string* varijable). Umesto test varijable tipa *int* ili *string*, može biti funkcija koja vraća varijablu tipa *int* ili *string*.

Instrukcija switch() je zgodnija od if() kada ima mnogo mogućih alternativa, ali nekad je zgodnija if() instrukcija jer je fleksibilnija od switch(), npr.

```
if(stringColor=="BLUE"){.....}  
else if (int1==20){.....}  
//itd
```

Upotreba objekta Textbox

06

OBJEKAT TEXTBOX-PRIMER

Može se ukucavati tekst u objekat TextBox u toku izvršavanja programa.

Objekat **TextBox** je veoma važan deo većine Windows-aplikacija. Glavna razlika između objekta **Label** i objekta **TextBox** je da

- se može ukucavati tekst u objekat **TextBox** u toku izvršavanja programa.

Takodje, razlika je u tome da se u **TextBox** može ubaciti puno teksta. U primeru aplikacije **Adding Machine**, primenjeni su objekti **TextBox**, tj. „Tekstboks“. Objekat **RichTextBox** je samo moćnija verzija objekta već opisanog ranije, objekta **TextBox**. Naime, ovde se može prikazivati i editovati formatizovan tekst.

Primer:

Dole je dat primer aplikacije gde se broji koliko je znakova ukucano u **TextBox**.

1. Postaviti **TextBox** i **Button** na **C#** - formu.
2. Dodati sledeći programski kod za objekat **Button** i događaj **Click**:

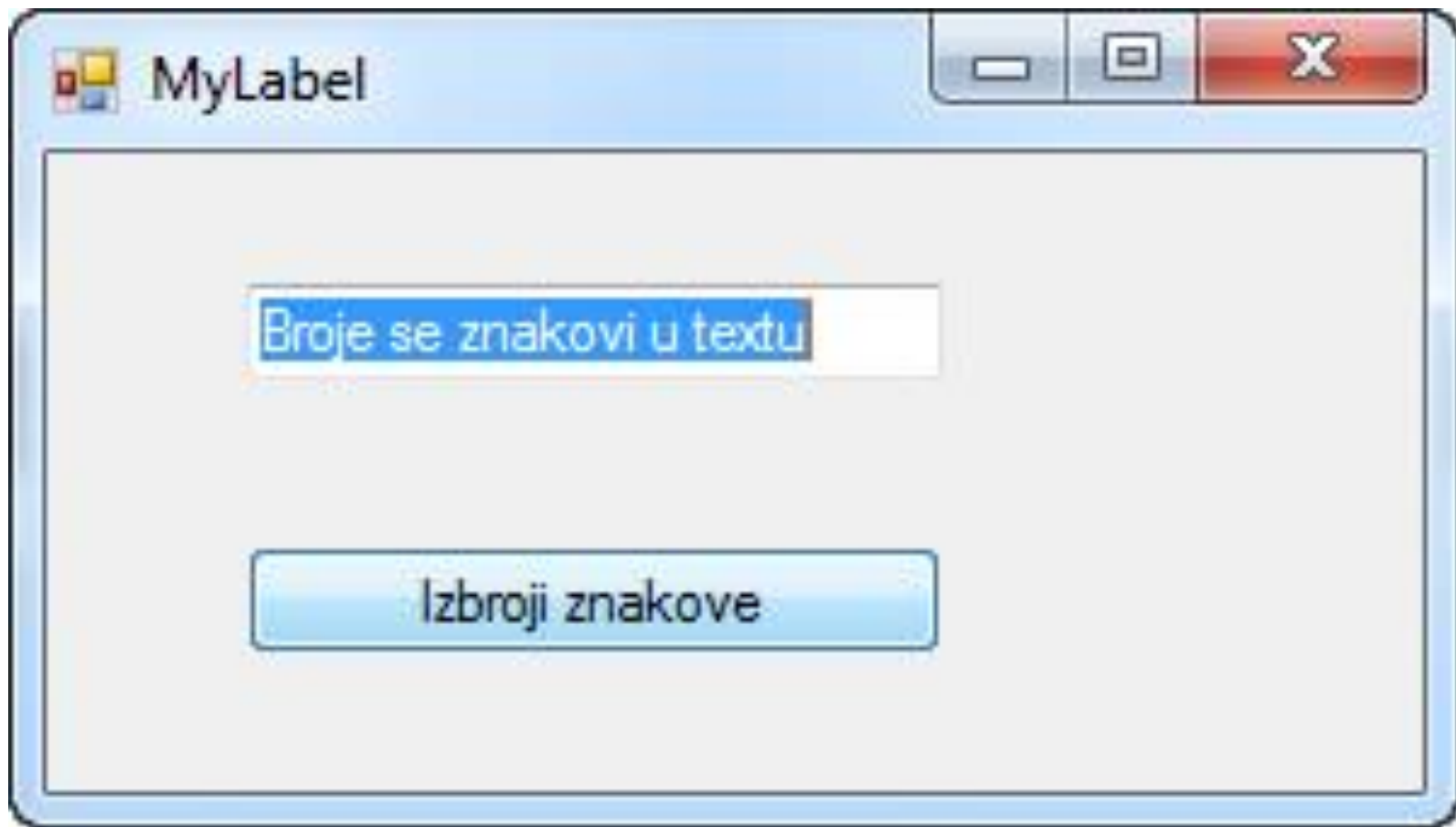
```
MessageBox.Show(“you typed: “ + textBox1.Text.Length.ToString() + “ characters.”);
```

3. Testiraj aplikaciju tako što se vrši egzekucija aplikacije, i pri tome ukucavanje teksta **“Some text to account“**, u **TextBox**, i klikanjem dugmeta.

Kao što vidimo, ovde smo koristili metodu **MessageBox.Show** za objekat **MessageBox**, i metodu **textBox1.Text.Length.ToString()** za objekat **TextBox**.

FORMA SA OBJEKTIMA TEXTBOX I BUTTON

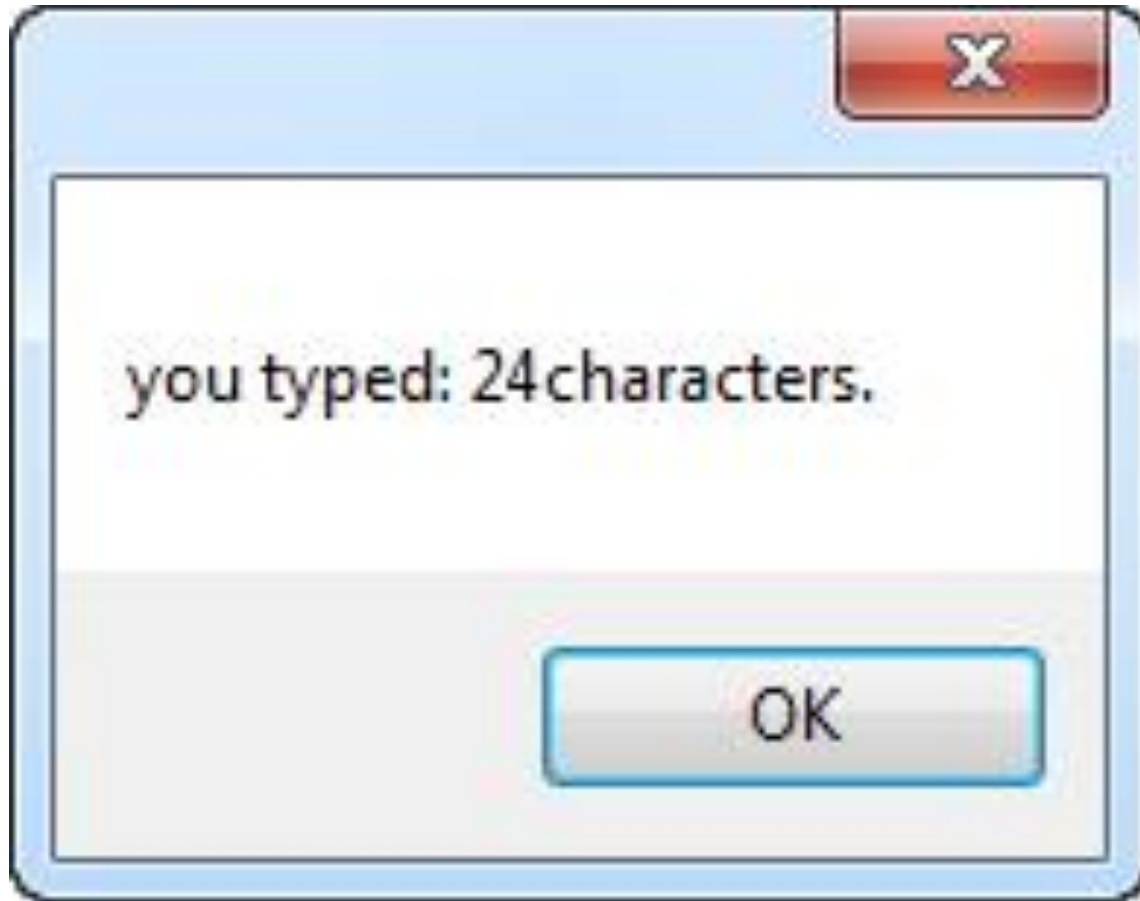
Na slici je forma sa objektima TextBox i Button.



Sl.1: forma

MESSAGEBOX

Slika prikazuje MessageBox.



Sl.1: message-box

Upotreba objekta Listbox

07

OBJEKAT LISTTBOX

Objekat ListtBox je jedan od najkorisnijih objekata u Visual C#. Ako se želi pristupiti nekom elementu u listi u ListBox, onda se koristi karakteristika ListIndex.

Objekat **ListtBox** je jedan od najkorisnijih objekata u Visual C#. Naime, ako se želi ponuditi tj prikazati izborne opcije korisniku, onda lista u ListBox može da sadrži od svega nekoliko do nekoliko hiljada, pa baze podataka kao npr adresari ili biznis-beleške često koriste ovakve liste da prikažu informacije. Ako se želi pristupiti nekom elementu u listi u **ListBox**, onda se koristi karakteristika **ListIndex** koja definiše koja linija u u listi se bira. Inače, događaj **Load** je izvanredan onda kada se postavljanje programskog koda koji se izvršava pre prikazivanja korisniku C# - forme.

Primer:

Dodavanje elemenata u listi u ListBox:

1. Postaviti jednu ListBox, jednu Label i jedno Button na C# - formu
2. Za objekt ListBox i osobinu Sorted zadati True, tako da će elementi u lčisti biti sortirani alfabetski.
3. Duplo kliknuti C# -formu, u cilju otvaranja događaja, *event*, Load. Zatim, koristiti metodu listBox.Items.Add da se dodaju elementi na listu u ListBox, npr

```
listBox1.Items.Add("C# programming");
```

ili

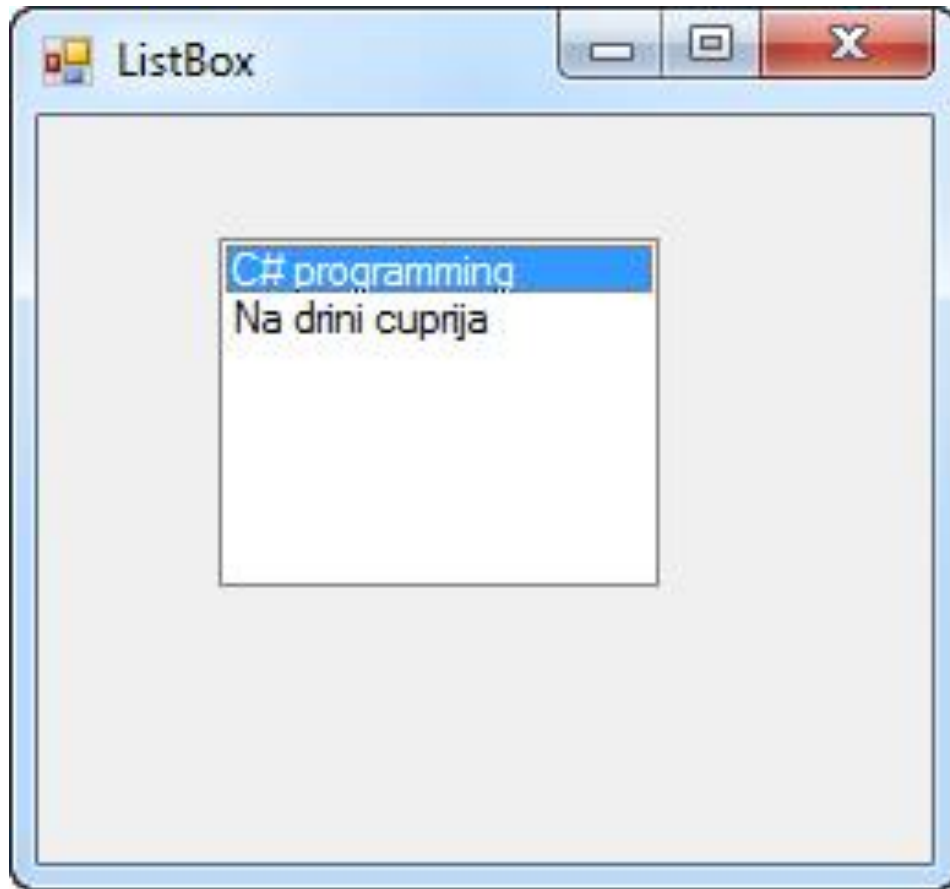
```
listBox1.Items.Add("Na drini cuprija");
```

itd.

4. Izvršiti aplikaciju, da bi se prikazala lista u ListBox.

C# -FORMA SA OBJEKTOM LISTBOX

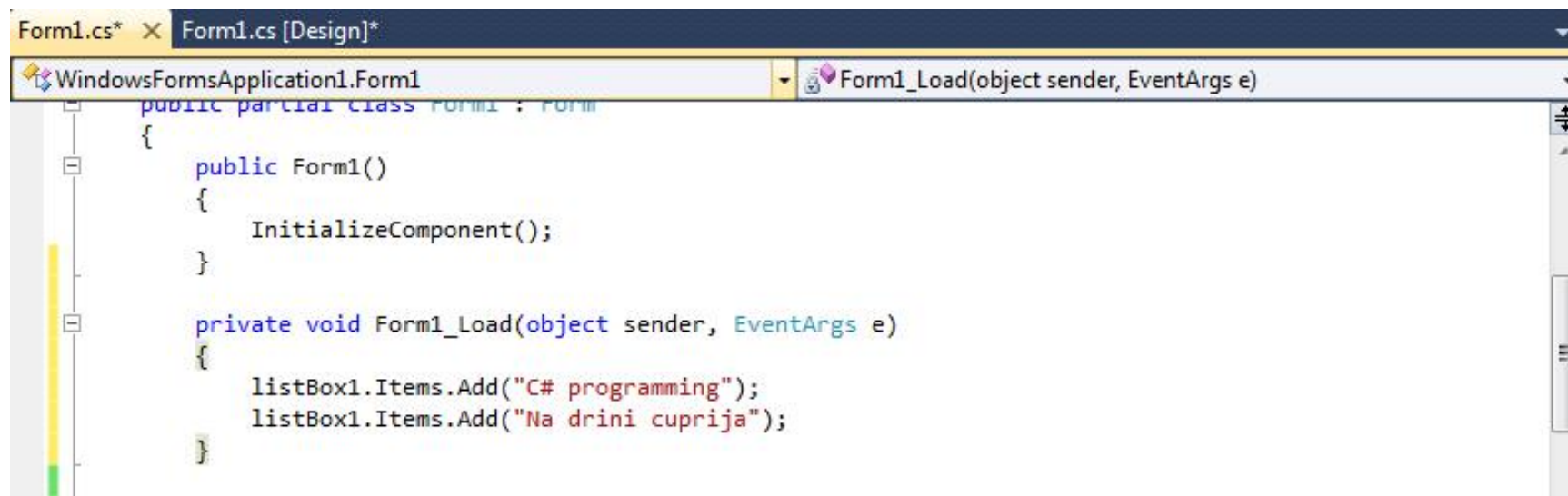
Na slici vidimo C# -formu sa objektom ListBox.



Slika 1: forma sa listboksom

KODIRANJE LISTBOX-A U CODE-EDITOR-U

Slika ilustruje kodiranje ListBox-a u code-editor-u.



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

    private void Form1_Load(object sender, EventArgs e)
    {
        listBox1.Items.Add("C# programming");
        listBox1.Items.Add("Na drini cuprija");
    }
}
```

Sl.2: kodiranje listboksa

Objekat OpenFileDialog

08

OBJEKTI OPENFILEDIALOG I SAVEFILEDIALOG

Objekti OpenFileDialog i SaveFileDialog se koriste na slični način kao i drugi objekti tj. „kontrola“ u Toolbox-u.

U mnogim Windows-aplikacijama, potrebno je Preuzeti tj. učitati, *load*, ili memorisati, *save*, fajlove na disk. Umesto da pišete programski kod za to, Visual C# koristi jednostavan Windows dialog. Objekti

OpenFileDialog i SaveFileDialog

se koriste na slični način kao i drugi objekti tj. „kontrola“ u Toolbox-u, naime, bira se ikonica u Toolbox-u, i postavlja se na C# - formu. Međutim, razlika je u tome, što se oni ne vide kod izvršavanja aplikacije, i što se oni pojavljuju u prostoru ispod C# - forme u Form-dizajneru. Podsetimo se, šta je to form-dizajner: to je radna površina označenom sa dugmetom **Form1cs[Design]***, gde naime treba kliknuti da bi se došlo u Form-dizajner.

Npr. dole je prikazana C# - forma koja sadrži **PictureBox**, i **Button**. Objekat **OpenFileDialog** se ne vidi na „formi“, ali se pominje u kodu aplikacije, npr. osobina **Filter**:

Primer: **Creating a Picture Viewer:**

1. postaviti **PictureBox**, **button**, i **OpenFileDialog** na formu.

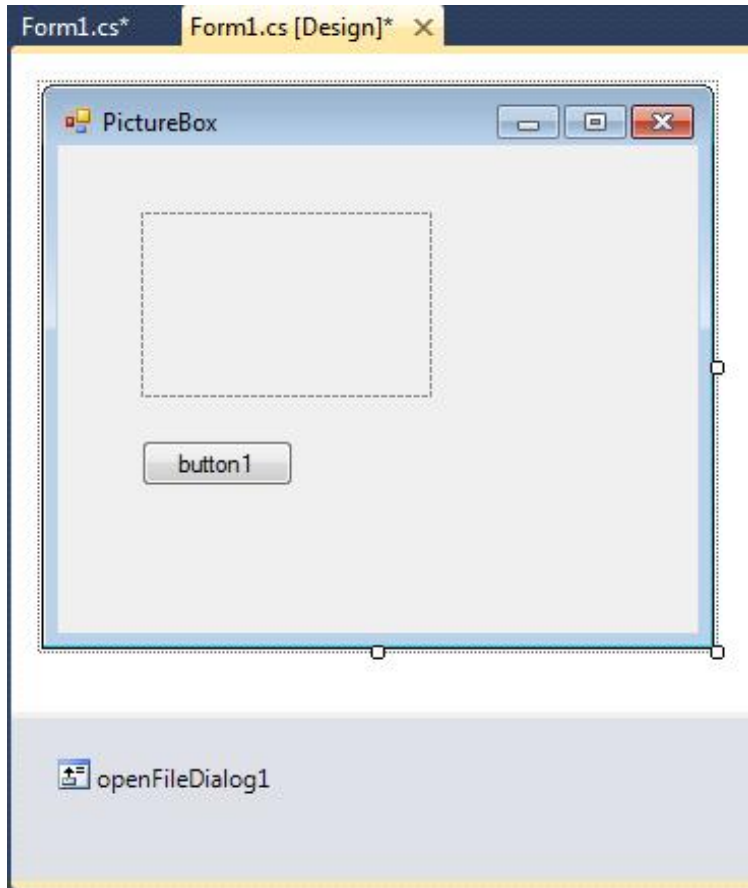
2. otvoriti **Load** event i dodati sledeću liniju koda:

```
this.openFileDialog1.Filter = "Pictures|*.bmp;*.ico;*.jpg;*.wmf";
```

3. Zatim se može testirati aplikacija ako se ona izvrši, klikne „dugme“ na formi, i nadje neki bitmap-fajl u **PictureBox-u**.

C# -FORMA SA OBJEKTIMA PICTUREBOX I BUTTON

Na slici je C# -forma sa objektima PictureBox i Button



Slika 1: fajl dijalog

Meni-editor

09

MENU EDITOR:

Ovde ćemo opisati još dva objekta iz *Toolbox*-a: **MainMenu**, i **ContextMenu**.

Kao što je poznato, niz objekata u *Toolbox*-u sadrži: *Label*, *Button*, *TextBox*, *ListBox*, itd. Ovde ćemo opisati još dva objekta iz *Toolbox*-a: **MainMenu**, i **ContextMenu**. Naime, u većini Windows-aplikacija, koristi se meni, *menu*, da omogući korisniku aplikacije potpuniju kontrolu aplikacije. *Visual Studio* omogućuje da se lako doda meni-linija, *menu bar*, u nekom projektu u Visual C#. Meni (tj. „meni-linija“, *menu bar*) se sastoji od nekoliko „dugmadi“, i kad kliknete neko dugme na meniju, onda se pojavljuju skup opcija za to dugme. A *pop-up* meni (tj. „pojavljujući“ meni) je meni koji se pojavljuje tek kad kliknete desnim dugmetom miša na neki objekat na formi (a pre toga se *pop-up* meni ne vidi), npr. kada kliknete na sliku na formi pojavljuje se *pop-up* meni.

Toolbox u Visual Studio-u:

Toolbox:

TextBox

MainMenu

CheckBox

.....

ContextMenu

ToolBar

.....

KREIRANJE MENIJA (MENI-LINIJE) POMOĆU *MAINMENU CONTROL*:

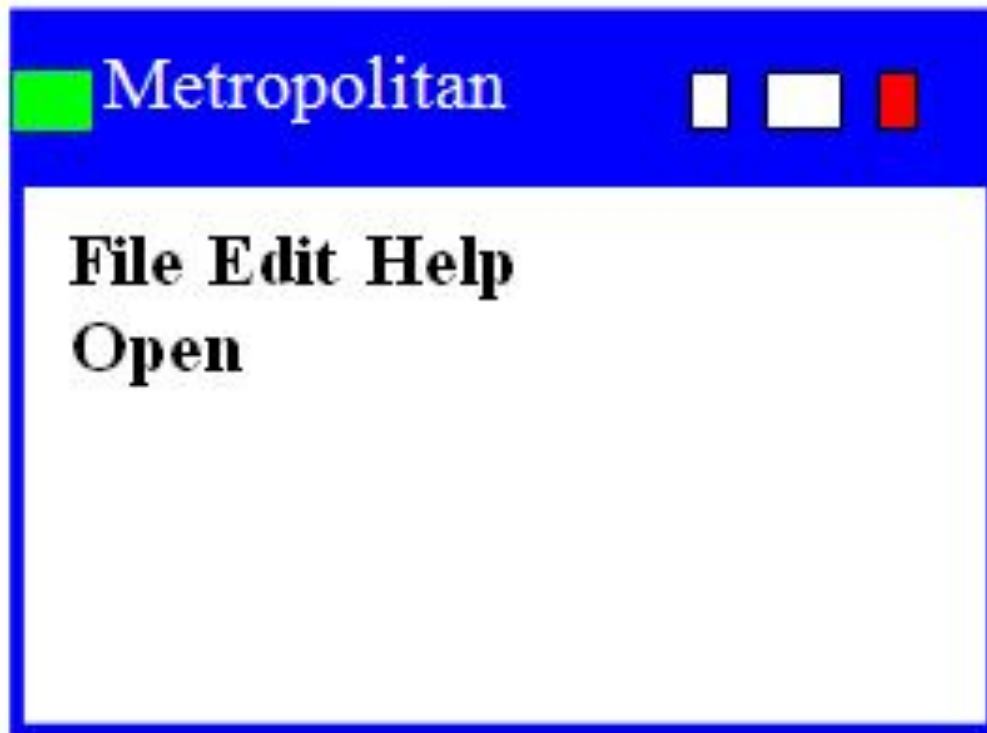
Na sledeći način možemo kreirati jedan meni (meni-liniju od nekoliko elemenata).

I to npr. na sledeći način možemo kreirati jedan meni (meni-liniju od nekoliko elemenata tj. dugmadi, gde kad pritisnete dugme pojavljuju se razne opcije) na nekoj „formi“, koji ima elemente **File**, **Edit**, **Help**, i **Open**, kao što je dole ilustrovano na crtežu. Evo primera kreiranja menija:

1. Startovati novi projekt, i iz *Toolbox*-a dodati „kontrolu“ *MainMenu* na „formu“. Zatim, pomoću miša selektovati „formu“, i onda u *Properties-window*, pronaći *Menu property* zadato po „defaultu“ da je jednako *mainMenu1*. Kliknuti meni-liniju na „formi“ na vrhu levo, i otkucati *File*.
2. Dodati meniju *Edit* i *Help* (pored *File*), a ispod dodati *Open*.
3. Pomoću *Property editor*-a, zadati imena elementima menija (*name property*), npr. *mnuFile*, *mnuFileOpen*, itd.
4. Izvršiti aplikaciju, i proveriti da li je kreirani meni aktivan, i ako još uvek opcije u meniju nemaju nekog efekta.
5. Zatim, može se meni-linija prilagoditi korisniku, *customized*, i ovo se radi pomoću *Properties-window*. Npr. meni-element tj. dugme menija *Open* može da sadrži opcije (*submenu*) *Open picture* i *Open text*, itd., tako da kod kliktanja dugmeta menija pojavljuju se te opcije.

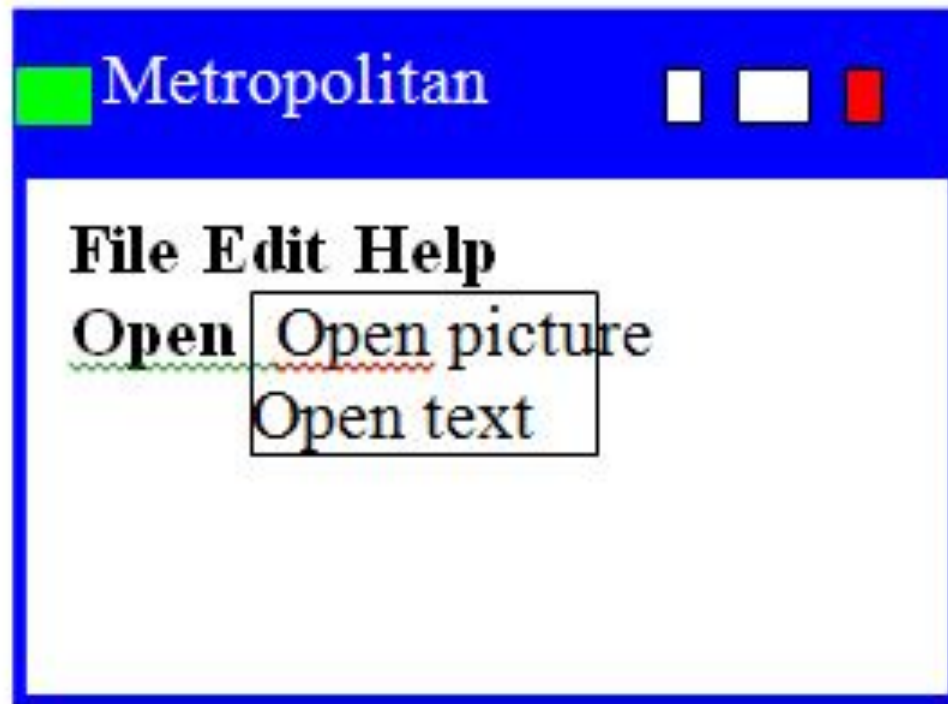
„FORMA“ SA MENIJEM

Na slici je „Forma“ sa menijem.



„FORMA“ SA MENIJEM I SA AKTIVIRANIM DUGMETOM OPEN

Na slici je „Forma“ sa menijem i sa aktiviranim dugmetom Open.



sl.2: forma

KREIRANJE *POP-UP* MENIJA POMOĆU *CONTEXTMENU CONTROL*:

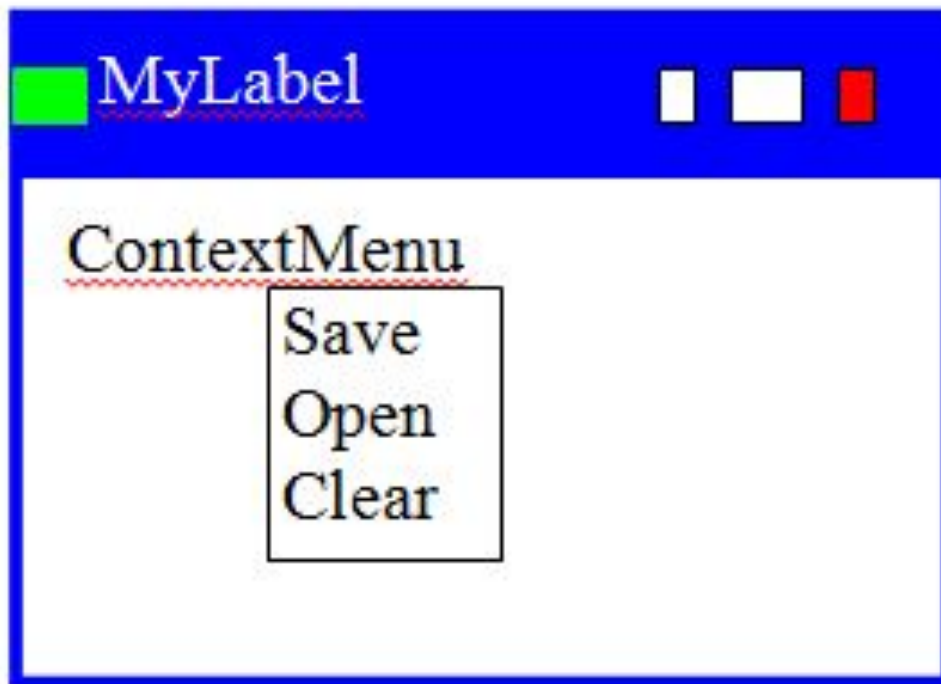
Evo primera kreiranja pop-up menija.

U Windows-aplikacijama često se koristi tzv *pop-up menu*, „pojavljujući“ meni, koji se pojavljuje kao rezultat klikanja desnog dugmeta na mišu sa strelicom miša na nekom objektu na formi (a pre toga je nevidljiv). *Pop-up* meni se može npr. napraviti na sledeći način:

1. Postaviti „kontrolu“ *ContextMenu* na „formu“. U stvari, pojavljuje se jedna ikona ispod forme, i kada se ona selektuje mišem, pojavljuje se *Context Menu* na „formi“. Kliknuti na *ContextMenu*, i ukucati ispod (na formi) pojedine stavke, npr. *Save*, *Open*, i *Clear*.
2. Postaviti „kontrolu“ *PictureBox* na „formu“. Za ovu „kontrolu“ zadati *ContextMenu property* u *Property editor*-u da je ime *contextMenu1*.
3. Izvršiti aplikaciju, i kliknuti desni „miš“ preko slike. Meni koji smo napravili, se otvara kao *pop-up* meni.

„FORMA“ SA POP-UP MENIJEM

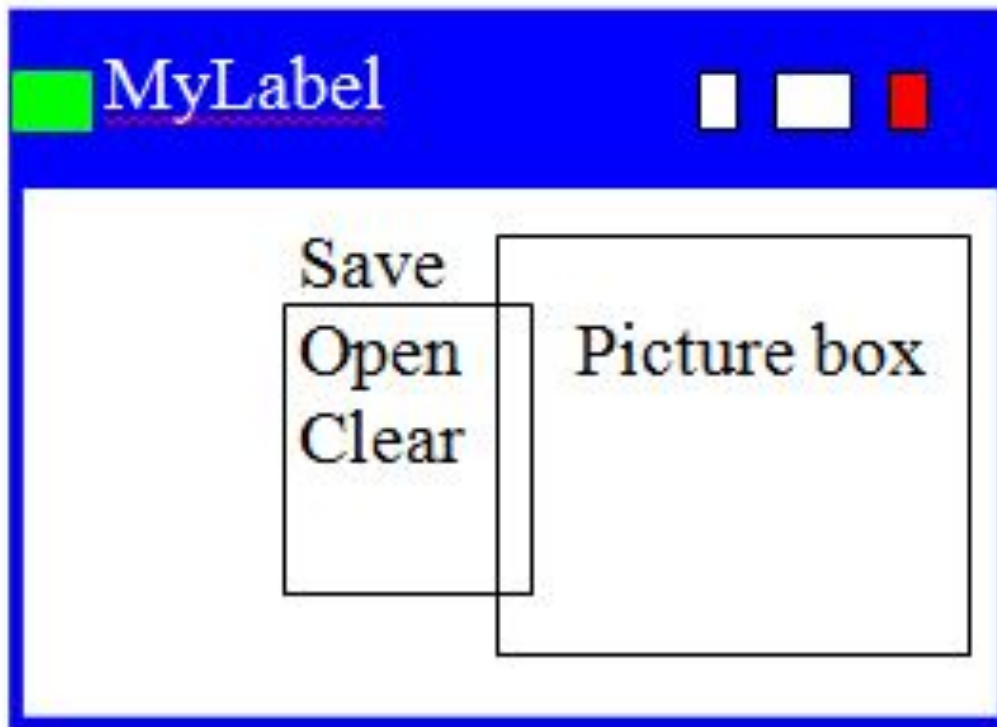
Na slici se vidi „Forma“ sa pop-up menijem.



sl.3: forma

„FORMA“ SA KLIKNUTIM *POP-UP* MENIJEM PORED *PICTURE BOX*-A

Na slici se vidi „Forma“ sa kliknutim pop-up menijem pored Picture box-a.



sl.4: forma

FUNKCIONALNOST MENIJA:

Ovde ćemo pokazati kako da se napiše programski kod koji će se izvršiti, kad se neka opcija menija izabere.

Meniji koje smo prethodno napravili lepo izgledaju, ali još uvek nemaju neku funkcionalnost, naime još uvek nemaju nikakvog efekta. Ovde ćemo pokazati kako da se napiše programski kod koji će se izvršiti, kad se neka opcija menija izabere:

1. Izabrati ikonu *Context menu* koja se nalazi ispod „forme“ u form-dizajneru, tako da je *Context Menu* vidljiv na „formi“.
2. Izabrati pojedine elemente menija i zadati odgovarajuće *name property*, npr. *mnuSavePicture*, *mnuOpenPicture*, itd.
3. Izabrati neku od opcija na meniju, i duplo kliknuti. Otvara se *code-editor* za *Click event handler* za konkretnu opciju menija:

```
private void mnuSavePicture_Click(object sender, System.EventArgs e)
```

```
{.....  
}
```

4. Ukucati programski kod koji memoriše sliku, i pri tome koristiti metodu *Save* u klasi *Image*.

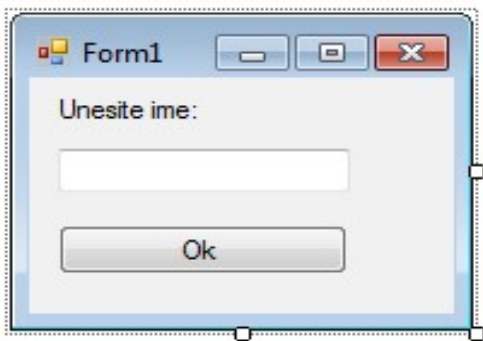
Vežba 3

10

BUTTON, TEXTBOX, LABEL, GROUPBOX I PANEL, PICTUREBOX, OPENFILE DIALOG

U ovom delu nam je osnovno da uradimo iščitavanje unetog teksta iz tekst boksa. Recimo, možemo ga ispisati na novi panel.

- Kreirajte formu kao na slici:



Slika-1: Forma za unos imena

Da bismo postigli ovakvo ponašanje forme, potrebno je da uradimo sledeće:

Nakon kreiranja forme, odredimo akciju kojom ćemo realizovati našu zamisao. Kako je to akcija dugmeta click, uradimo dupli levi klik na dugme OK.

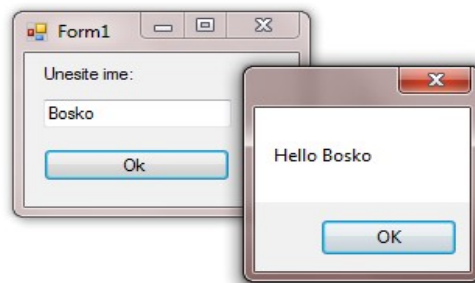
Odmah nakon toga će se otvoriti prikaz koda u kome ćemo moći da programskim kodom rešimo naš zadatak.

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

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello " + textBox1.Text);
        }
    }
}
```

U okviru akcije dugmeta button1_Click unesite kod sa listinga. Time ćemo rešiti da nam se prikaže novi MessageBox sa prikazanim pozdravom i imenom koje smo uneli u tekst boks.



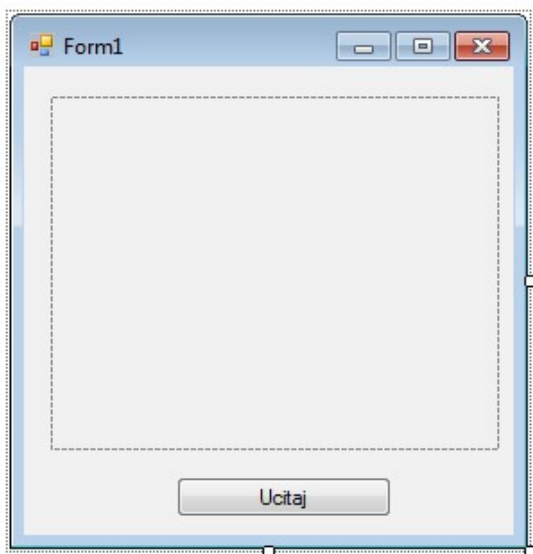
Slika-2: MessageBox

PICTUREBOX, OPENFILEDIALOG

Pretpostavimo da želimo da odaberemo neku sliku iz našeg računara, da je učitamo i prikažemo u nekom prozoru.

Pretpostavimo da želimo da odaberemo neku sliku iz našeg računara, da je učitamo i prikažemo u nekom prozoru. U tom slučaju biće nam potrebno nekoliko windows kontrola i naravno, nekoliko linija koda koje će to da omogućće.

Kreirajte formu kao na slici:



Slika-3: Forma

Vodite računa da je prostor koji je na slici u veži pictureBox. Nakon što ste ga kreirali, dodajte akciju dugmeta.

Za akciju dugmeta, posebno regulišite učitavanje fajla. Pregled koda je dat ispod.

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

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

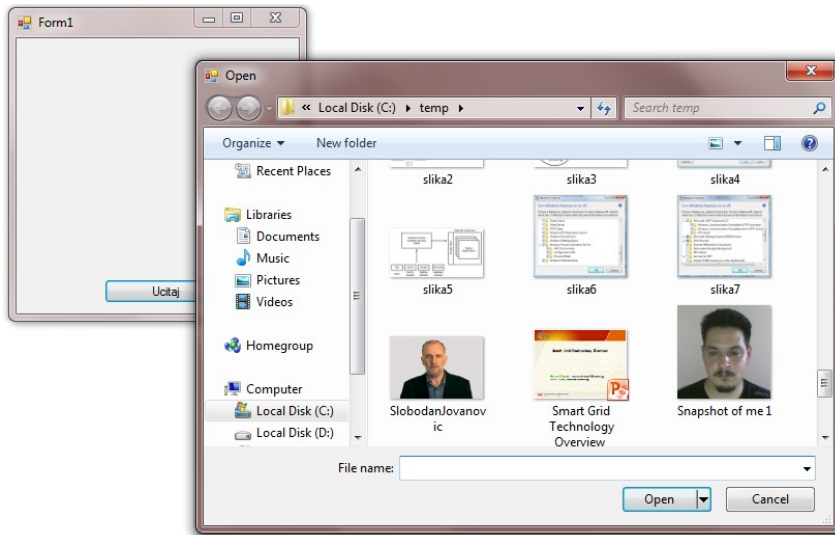
        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog of = new OpenFileDialog();
            DialogResult rez = of.ShowDialog();
            if (rez == DialogResult.OK)
            {
                Bitmap slika = (Bitmap)Bitmap.FromFile(of.FileName);
                pictureBox1.Image = slika;
            }
        }
    }
}
```

PICTUREBOX, OPENFILEDIALOG DRUGI DEO

Pretpostavimo da želimo da odaberemo neku sliku iz našeg računara, da je učitamo i prikažemo u nekom prozoru.

Realizacijom ovog koda dobijamo učitavanje neke slike sa hard diska.

Slika



Slika-4: Biranje slike

Odaberemo konkretnu sliku i ona će biti učitana u pictureBox.



Slika-5: Prikaz odabrane slike

RAZNI PRIMERI

Pretpostavimo da želimo da odaberemo neku sliku iz našeg računara, da je učitamo i prikažemo u nekom prozoru.

U cilju izbegavanja prevelikog rasplinjavanja u prikazu realizacije, u nastavku je dat samo potreban kod sa zadatkom. Vaš je zadatak da sve ove vežbe proradite i uverite se u način realizacije. Takvom realizacijom ćete i shvatiti upotrebu konkretne kontrole koja se obrađuje.

Text Box, button

Na osnovu unetog rednog broja dana u nedelji u jednom tekst boksu, ispisati naziv dana u drugom tekst boksu.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace dani
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            tBRedniBrojDana.Text = "unesi";
        }
    }
}
```

Timer, radio box, check box

Prikaz kontrola Timer, radio box I CheckBox kroz implementaciju brojanja.

```
using System;
using System.Drawing;
using System.Collections; using System.ComponentModel; using
System.Windows.Forms; using System.Data;
namespace Brojanje
{
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button btNapred; private
System.Windows.Forms.Button btNazad; private
System.Windows.Forms.Button btZaustavi; private
System.Windows.Forms.Button btPonisti; private
System.Windows.Forms.Timer timer1;
        private System.ComponentModel.IContainer components;
        int korak;
        private System.Windows.Forms.TextBox tbBroj;
        public Form1()
        { InitializeComponent(); }
        static void Main()
        { Application.Run(new Form1()); }
        private void btNapred_Click(object sender, System.EventArgs e)
        { timer1.Enabled = true; korak = 1; }
        private void btZaustavi_Click(object sender, System.EventArgs e)
        { timer1.Enabled = false; }
        private void btNazad_Click(object sender, System.EventArgs e)
        { timer1.Enabled = true; korak = -1; }
        private void btPonisti_Click(object sender, System.EventArgs e)
        { tbBroj.Text = "0"; timer1.Enabled = false; }
        private void timer1_Tick(object sender, System.EventArgs e)
        {
            int broj = Convert.ToInt32(tbBroj.Text);
```

RAZNI PRIMERI DRUGI DEO

Pretpostavimo da želimo da odaberemo neku sliku iz našeg računara, da je učitamo i prikažemo u nekom prozoru.

CheckBox, label

Jednostavan primer za promenu fonta labela u zavisnosti od stanja check-box-a.

```
using System;
using System.Drawing;
using System.Collections; using System.ComponentModel;
using System.Windows.Forms; using System.Data;
namespace primer3a
{
public class CheckBoxTest : System.Windows.Forms.Form
{
private System.Windows.Forms.CheckBox BoldCheckBox;
private System.Windows.Forms.CheckBox ItalicCheckBox;
private System.Windows.Forms.Label outputLabel;
private System.ComponentModel.Container components = null;
public CheckBoxTest()
{
InitializeComponent();
}
}
#region Windows Form Designer generated code
#endregion

[STAThread]
static void Main()
{
Application.Run(new CheckBoxTest());
}

private void BoldCheckBox_CheckedChanged(object sender,
System.EventArgs e)
```

MessageBox, radio button

Jednostavan primer prikaza različitih MessageBox kontrola.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace MessageBox1
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
MessageBox.Show("Zdravo!", "pozdrav", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
}
private void button2_Click(object sender, EventArgs e)
{
DialogResult rez;
rez = MessageBox.Show("Da li zelite", "Brisanje",
MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (rez
== DialogResult.Yes)
{
MessageBox.Show("Brisem!");
}
}
}
```

RAZNI PRIMERI DRUGI DEO

Pretpostavimo da želimo da odaberemo neku sliku iz našeg računara, da je učitamo i prikažemo u nekom prozoru.

Radio Button, listBox

Jednostavan prikaz forme koja rukuje temperaturama pomoću radio button kontrole i list box kontrole.

```
using System;
using System.Drawing;
using System.Collections; using System.ComponentModel; using
System.Windows.Forms; using System.Data;
namespace temperature
{
public class temperature : System.Windows.Forms.Form
{
int min, max, s = 0;
private System.Windows.Forms.Label lTemp;
private System.Windows.Forms.TextBox tBTemperatura;
private System.Windows.Forms.Button btDodaj;
private System.Windows.Forms.ListBox lBTemperatura;
private System.Windows.Forms.GroupBox gBIzbor; private
System.Windows.Forms.RadioButton rBRaspon; private
System.Windows.Forms.RadioButton rBMin; private
System.Windows.Forms.RadioButton rBMax; private
System.Windows.Forms.RadioButton rBProsek; private
System.Windows.Forms.Label lIspis;
public temperature()
{ InitializeComponent(); }
static void Main()
{ Application.Run(new temperature()); }
private void btDodaj_Click(object sender, System.EventArgs e)
{
lBTemperatura.Items.Add(tBTemperatura.Text);
if (lBTemperatura.Items.Count == 1)//postavljanje min i max na prvu unetu
min = max = Convert.ToInt32(tBTemperatura.Text);
s = s + Convert.ToInt32(tBTemperatura.Text); // dodavanje temperature
if (Convert.ToInt32(tBTemperatura.Text) > max) // korekcija min i max
max = Convert.ToInt32(tBTemperatura.Text);
}
```

RAD SA MENIJIMA

Svaka Windows aplikacija ima meni kao davno ustanovljeni vid interakcije sa korisnikom.

Svaka Windows aplikacija ima meni kao davno ustanovljeni vid interakcije sa korisnikom. Prednosti menija su mogućnost hijerarhijske organizacije koje odgovaraju vašoj aplikaciji uz minimalno zauzeće prostora.

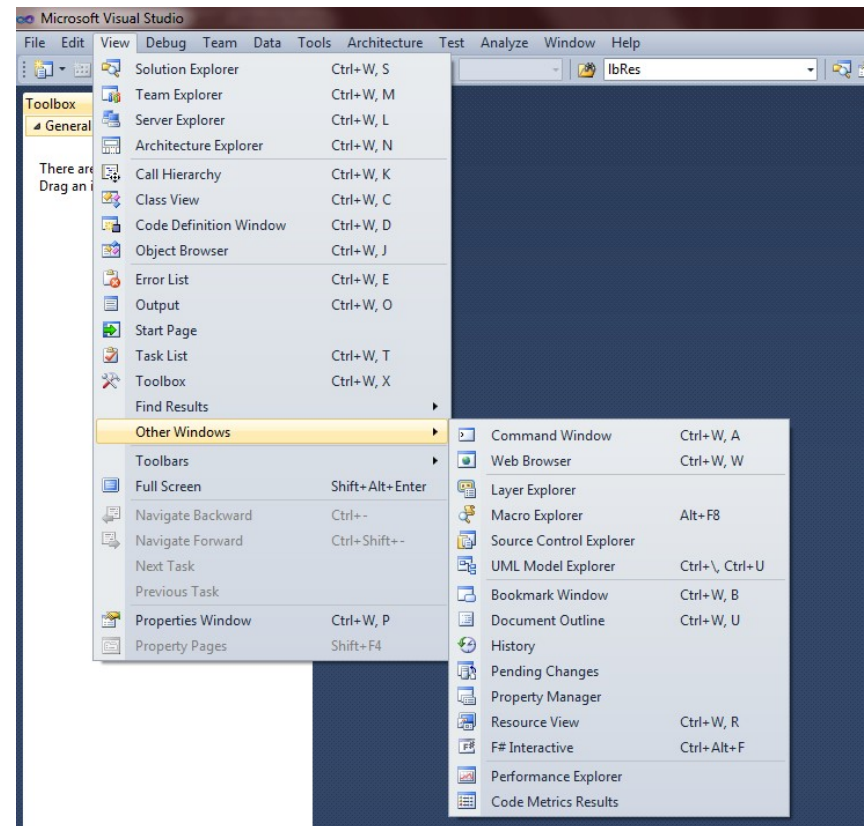
Meni se obično nalazi pri vrhu forme, ali to ne mora biti slučaj. Možete ga postaviti na dno ekrana, pa čak i po vertikali ako vam to odgovara.

Dodatnu snagu menija predstavlja mogućnost dinamičkog dodavanja ili sakrivanja stavki čime meni prati korisnika zavisno od akcija koje se događaju u aplikaciji.

Prilikom kreiranja menija razlikujemo horizontalnu liniju i vertikalne delove menija koji se prikazuju kada korisnik klikne na stavku u horizontalnoj liniji. Vertikalna linija može imati stavki na više nivoa hijerarhije. Na sledećoj slici vidite View meni iz radnog okruženja Visual C#.

Osim teksta stavka menija može imati sličicu (eng. Icon) i definisanu skraćenicu sa tastature (eng. Shortcut).

Slika 6:



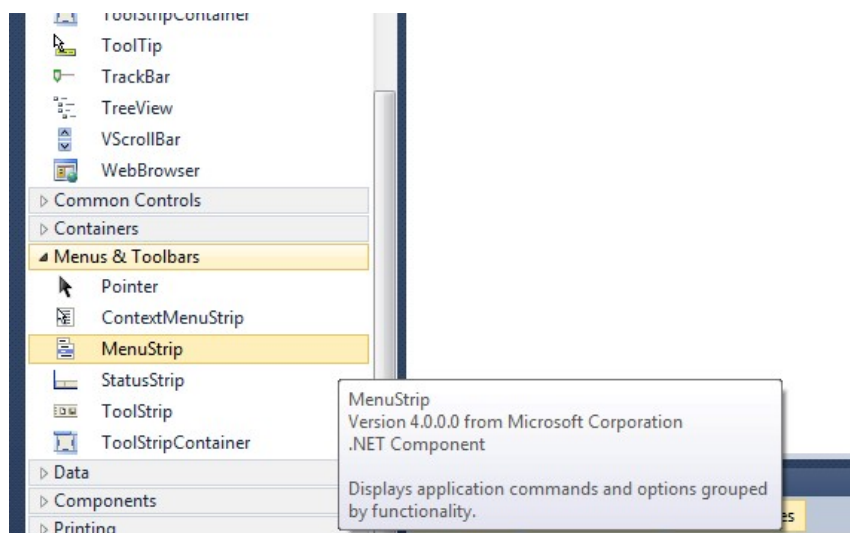
Slika-6: Meni

KREIRANJE MENIJA

Pokrenite Visual C# Express razvojno okruženje i kreirajte novu Windows aplikaciju.

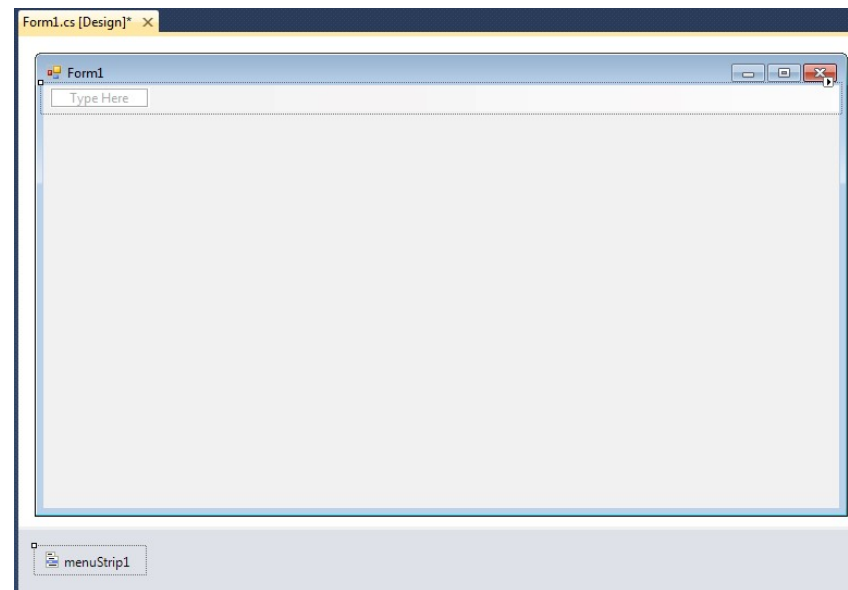
Pokrenite Visual C# Express razvojno okruženje i kreirajte novu Windows aplikaciju. Meni ćemo kreirati na inicijalnoj formi (Form1).

U paleti sa alatkama (eng. Toolbox) pronađite sekciju „Menus & Toolbars“. U okviru nje pronađite kontrolu „MenuStrip“ koja daje potrebne funkcionalnosti menija na formi.



Slika-7: MenuStrip

Prevucite je i pustite na formu - nije važno gde, jer se meni automatski pozicionira na vrh forme, a sama kontrola u specijalnom prozoru ispod forme kao što je prikazano na sledećoj slici.



Slika-8: Meni na formi

KREIRANJE MENIJA DRUGI DEO

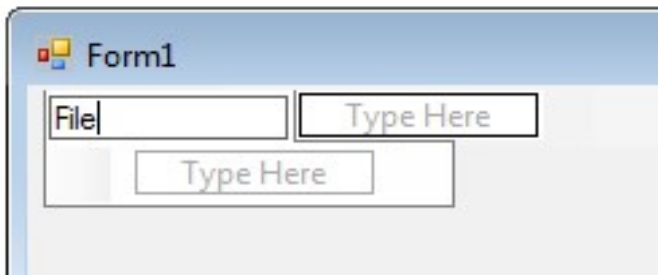
Pokrenite Visual C# Express razvojno okruženje i kreirajte novu Windows aplikaciju.

Ako želite da kreirate meni, prvo morate da kliknete na kontrolu na dnu forme koja je dobila inicijalno ime „menuStrip1“. Tada se u dizajneru prikazuje meni na vrhu forme.

Zapazite tekst „Type Here“ koji dovoljno govori sam za sebe. Napravićemo deo menija koji je manje-više standardan za većinu aplikacija.

Tamo gde piše „Type Here“ otkucajte „Datoteka“ (bez znakova navoda) ili „File“ ako želite meni na engleskom jeziku.

Čim počnete da kucate odmah će se ispod i desno od ove stavke otvoriti mogućnost za dalji unos stavki menija, horizontalno i vertikalno:



Slika-9: Dodavanje opcije

Verovatno ste primetili da svaki meni ima pridružen takozvani „hot key“. Slovo ili broj koji uz pritisnut „Alt“ specijalni taster predstavlja skraćenicu za izbor stavke u horizontalnom delu menija. Ova funkcionalnost se jednostavno definiše – potrebno je ispred karaktera koji treba da bude „hot key“ ukuca karakter &. Ako želimo da „hot key“ za meni „File“ bude kombinacija Alt+D jednostavno ukucajte „&File“. Karakter & se ne prikazuje, već je ispisano „File“. Podvučeno slovo označava skraćenicu sa tastature. Ovo se i vidi čim pritisnete Enter taster čime završavate unos teksta u stavci menija. Nastavljamo sa menijem „File“ tako što ćemo ispod njega napraviti nekoliko stavki. Ukucajte jedno ispod drugog, redom:

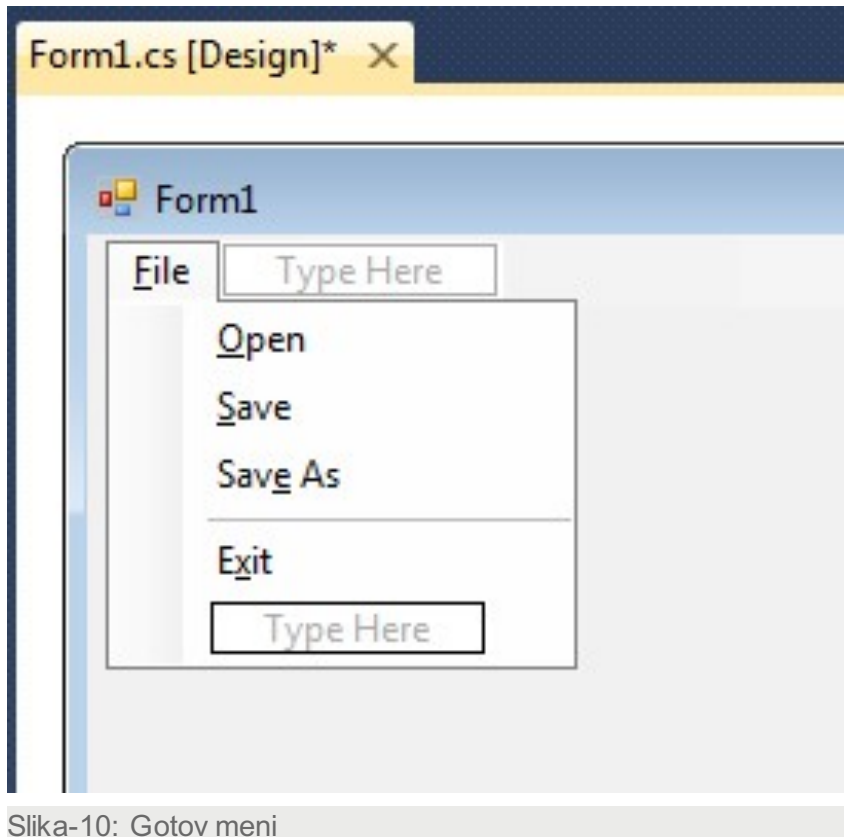
- &Open
- &Save
- Save &as
- &Exit

Ako za tekst stavke unesete samo znak – (minus), on ima specijalno značenje: u meniju se prikazuje horizontalni separator. Korisno kada treba da vizuelno razdvojite različite funkcionalne grupe u okviru vertikalnog menija.

KREIRANJE MENIJA TREĆI DEO

Pokrenite Visual C# Express razvojno okruženje i kreirajte novu Windows aplikaciju.

Na kraju naš meni treba da izgleda kao na sledećoj slici:



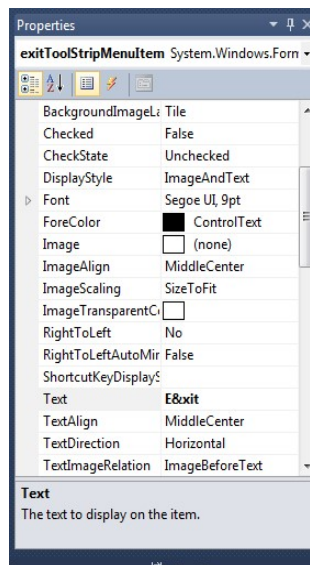
Slika-10: Gotov meni

PROGRAMIRANJE MENIJA

Pokrenite Visual C# Express razvojno okruženje i kreirajte novu Windows aplikaciju.

Svaku stavku menija je najbolje tretirati kao komadno dugme (eng. Button). Programiranje se vrši u Click događaju stavke menija.

U primeru koji smo započeli kliknite samo jednom na stavku menija „Kraj rada“. Svaka stavka menija ima svoj set svojstava koja su vidljiva u standardnom prozoru za prikaz svojstava objekta - „Properties“ prozoru:



Slika-11: Properties

Kao i kod svake druge kontrole ključno svojstvo je „Name“ koje je automatski generisano – u našem slučaju „krajRadaToolStripMenuItem“. Svojstvo „Text“ definiše tekst stavke menija i njega smo podešavali iz dizajnera. Sada uradite dupli klik na stavku menija „Kraj rada“, čime je otvoren Click događaj u prozoru za pisanje koda. Kada korisnik klikne na „Kraj rada“ želimo da zaustavimo izvršavanje programa. Upišite sledeću liniju koda:

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Application je referenca na aplikaciju koju kreiramo. Posедуje metod Exit koji izvršava bezuslovni prekid rada aplikacije i oslobađa sve resurse koje je aplikacija zauzela (memoriju, datoteke, baze podataka i tako dalje...)

Pokrenite aplikaciju i mišem kliknite na „Kraj rada“. Pokušajte takođe i pomoću skraćenica sa tastature:

prvo Alt+D kako bi otvorili meni „Datoteka“, a potom samo taster K kako bi aktivirali stavku „Kraj rada“.

SKRAĆENICE

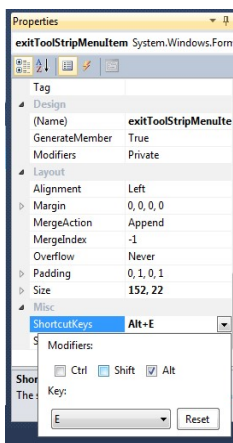
Skraćenice se mogu definisati kombinacijom bilo kog specijalnog tastera (Control, Alt ili Shift) i običnog karaktera.

Skraćenice se mogu definisati kombinacijom bilo kog specijalnog tastera (Control, Alt ili Shift) i običnog karaktera. Takođe možete iskoristiti funkcijske i druge specijalne karaktere sa tastature.

Zaustavite program ako je pokrenut i ponovo jednim klikom miša izaberite stavku

„Kraj rada“.

Potom u prozoru svojstava (eng. Properties) pronađite svojstvo „ShortcutKeys“. Otvorite padajuću listu pored ovog svojstva i trebalo bi odmah sve da bude jasno:



Slika-12: Shortcut keys

Markirajte Alt i iz padajuće liste izaberite slovo E kao što je prikazano na gornjoj slici. Sada kombinacija Alt+E završava našu aplikaciju. Pokrenite aplikaciju i probajte.

Primitite da se uz stavku menija prikazuje i njena skraćenica. Ovo se podešava svojstvom „ShowShortcutKeys“ koja može imati vrednosti True (da) ili False (ne).

SLIKE

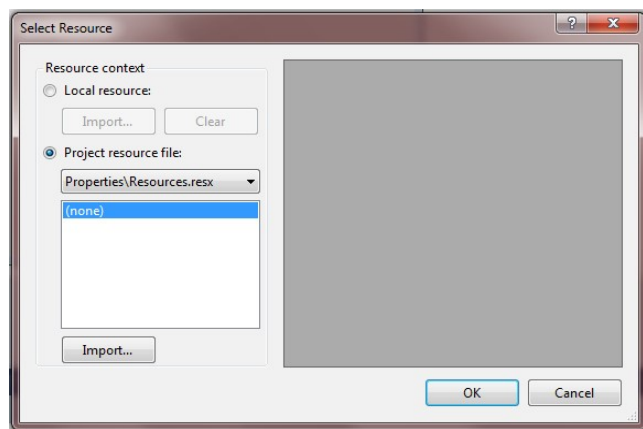
Svaka stavka menija (uključujući i stavke u horizontalnoj liniji) može imati i sličicu (eng. Ikon) koja doprinosi lepšem i razumljivijem korisničkom interfejsu.

Svaka stavka menija (uključujući i stavke u horizontalnoj liniji) može imati i sličicu (eng. Ikon) koja doprinosi lepšem i razumljivijem korisničkom interfejsu.

Sličice koje se koriste za ovu namenu su najčešće bmp ili jpeg formata, dimenzija 16x16 piksela.

Pomoću svojstva „Image“ možete da izaberete sliku sa vašeg računara i dodelite je stavci vašeg menija.

Kliknite na malo dugme pored ovog svojstva i dobićete sledeći dijalog:



Slika-13: Biranje slike

U ovom dijalogu se dodaju resursi aplikaciji. Oni osim slika mogu biti i zvučne datoteke, animacije kao i druge binarne datoteke koje koristite u aplikaciji.

Kliknite na dugme „Import“ i sa diska vašeg računara izaberite sliku koju ste pripremili. Na kraju kliknite na dugme „OK“. Ako je slika veća od 16x16 piksela, biće automatski skalirana na tu veličinu.

Korisna je i mogućnost da pored svake stavke menija imate znak za overu (eng. Check).

Svojstvo Checked koje može imati vrednosti True ili False obezbeđuje upravo ovo.

Izaberite stavku „Save“ u svojstvima pronađite Checked i postavite ga na True. Sa leve strane ove stavke se pojavljuje znak za overu.

U praksi se znak za overu prikazuje ili nezavisno od logike programskog toka aplikacije i akcija korisnika. Na primer, linija koda:

SLIKE DRUGI DEO

Svaka stavka menija (uključujući i stavke u horizontalnoj liniji) može imati i sličicu (eng. Ikon) koja doprinosi lepšem i razumljivijem korisničkom interfejsu.

će za vreme izvršavanja programa postaviti znak za overu pored stavke menija.

Takođe, u praksi se često koristi dinamičko prikazivanje ili skrivanje stavke menija ili cele grupe, opet zavisno od logike aplikacije. Za ovo imamo dve mogućnosti, odnosno svojstva:

1. Svojstvo Enabled koje kada je postavljeno na False, prikazuje stavku menija u sivoj boji i korisnik ne može da klikne na nju. Korisno ako želite da korisnik zna da stavka menija postoji, ali trenutno nije dostupna.

2. Restriktivnije svojstvo Visible (koje imaju sve kontrole) postavljeno na False

čini da se stavka menija uopšte ne vidi.

ZADACI ZA INDIVIDUALNU VEŽBU

Zadatak 1.

Napraviti aplikaciju za konverziju slika iz JPEG u PNG koristeći PictureBox i OpenFileDialog.

Zadatak 2.

Napraviti aplikaciju za okretanje String-a naopačke. Korisnik treba u TextBox da unese ime a potom klikom na dugme treba da mu se okrene string i prikaže u Labelu.

Zadatak 3.

Napraviti aplikaciju za pogađanje brojeva. Aplikacija treba da generiše random broj od 0 do 10 a korisnik treba da pogađa broj preko TextBox-a.

Zaključak

ZAKLJUČAK

Zaključak

-U ovom predavanju ćemo analizirati tj. ilustrovati upotrebu objekata u Visual C# Toolbox-u, predefinisane objekata raspoložive u Toolbox-u, koji se mogu iskoristiti da se složi neka vizuelna C# - aplikacija npr neki od sledećih objekata, Button, Label, TextBox, PictureBox, CheckBox,GroupBox, ListBox, OpenFileDialog Control,Tab Control, RichTextBox, ToolBar,ItD

-Posmatrajmo neki fizički oobjekt, npr. Auto, I taj auto ima osobine boje, starosti, maksimalne brzine, itd., A npr. Pokretanje i Zaustavljanje auta ili Trubljenje ili Kočenje su ono što auto radi, tj. akcije objekta koje se obavljaju u odredjenom periodu.I za auto se mogu znači definisati metoda Pokretanja, metoda Zaustavljanja, itd.

-Po pravilu, metode zahtevaju dodatne informacije, koje se zovu parametri metode, parametri metode se upisuju iza imena metode objekta, i to se upisuju u zagradama, i ovi parametri opisuju kako će se akcija obavljati. Npr, auto može da se pokrene vrlo brzo ili vrlo sporo.

-Parametri metode se upisuju dakle u zagradama iza imena metode, ali i ako nema potrebe za parametrima neke metode, tj. neke metode su bez parametara, ali ipak se stave dve prazne zagrade iza imena metode.