Tehnike vizuelnog programiranja

VSER

Sadržaj kursa...

- Osnove .NET platforme
- C#
- Windows aplikacije
  - Događaji
  - Forme
  - Kontrole
  - MDI/SDI, Kontejneri
  - Meniji, Toolbar
  - GDI+, Niti
- ADO.NET
- Povezivanje sa podacima
- Asinhrono programiranje

Literatura...

- ***Materijali sa stranice predmeta.***

- *Tehnike vizuelnog programiranja* – VETŠ 2005. Beograd
- *Microsoft Visual C#.NET Step by step* – Microsoft Press (prevod CET) John Sharp, Jon Jagger
- *Microsoft.NET framework* – Microsoft Press (prevod CET) Jefrey Richter
- *Developing Windows-Based Applications with Microsoft Visual Basic.NET and Microsoft Visual C#.NET* - (prevod CET) Matthew A. Stoecker
- *C# Primer: A Practical Approach* – (prevod CET) Stanley B. Lippman
- *C# Tips and Techniques* - (Mikro knjiga) The McGraw-Hill Companies, Charles Wright

Ispit

- Kada je u celosti, radi se na računaru u terminu ispita
  - Polaže se gradivo predavanja + vežbe
  - Projekti se brane ili se radi zadatak.
- Parcijalno:
  - **Rad tokom cele godine (predavanja + vežbe)**
  - **Projekti – 2 kom**
  - Mogu se koristiti projekti koji su već rađeni ako pokrivaju dovoljno gradiva

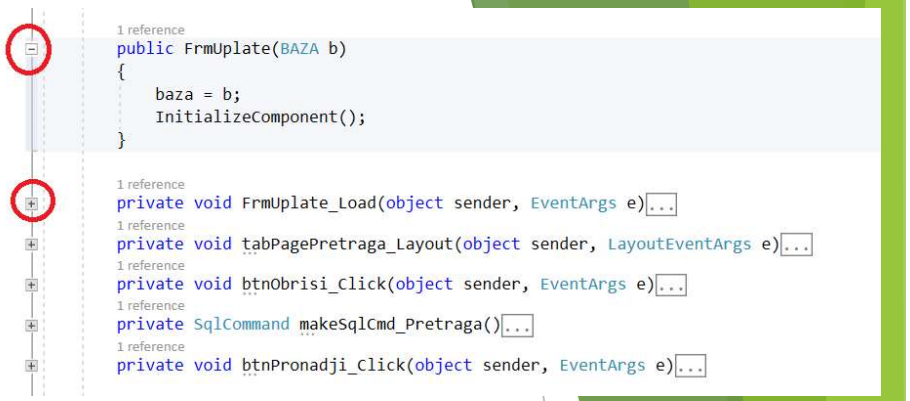
.NET (dot net) platforma...

- .NET (dot net) je proizvod softverske kompanije Microsoft. Veži za trenutnu i buduću orijentaciju u pogledu softvera iste kompanije, ali i mnogih drugih multinacionalnih kompanija. Platforma se razvija i u pravcu otvorenog koda, web aplikacija, komponent orijentisanih aplikacija.
- .NET je osnova za nove operativne sisteme. Koristi se u visoko pouzdanim sistemima za kontrolu leta. Omogućava razvoj univerzalnih Windows aplikacija. Omogućava razvoj i univerzalnih moblnih aplikacija (iOS, Android, WP). Otvorena specifikacija omogućila je realizaciju i na operativnim sistemima koji nisu Windows.
- .NET je se uveliko primenjuje u razvoju distribuiranih aplikacija i predstavlja jedno od najproduktivnijih rešenja u oslasti servisno orijentisanih arhitektura (SOA).
- Novi stil stvaranja kolekcija specifičnog koda na jednom mestu umesto dosadašnjeg gde se stvaraju redudantne kopije na mnogo mesta. Pruža programske celine raspoložive različitim uređajima.
- Pomera razvoj aplikacija od objektno orijentisanih ka servisnim (engl. *services privided)*. Jezička nezavisnos postignuta je postojanjem međujezika (IL ili MSIL).
- Podrška za WEB (XML) SERVISE. Jaka podrška za XML standard. Poboljšani pristup bazama podataka preko ADO.NET tehnike. Znatno unapređen rad sa dinamičkim Web stranicama baziranim na ASP.NET tehnologiji. Umesto dll-ova uvodi se koncept sklopova. Rad pod različitim platformama zasniva se na postojanju **.NET Framework-a**.
- .NET Framework sadrži **CLR** (Common Language Runtime) kao i kolekcije klasa ovog okruženja. Svi programski jezici obuhvaćeni Microsoft VisualStudio.NET imaju zajedničku

C# vs Java

- **Sličnosti**
  - Oba jezika imaju mehanizam za oslobađanje memorije tzv sakupljanje otpada (engl. "*garbage collection*" - GC). Pri razvoju ne moramo voditi računa o oslobađanju memorije, GC prepoznaje delove memorije koje se ne koriste i stavlja ih na raspolaganje aplikaciji po potrebi.
  - Kao i Java, C# ne podržava višestruko nasleđivanje.
  - Prevođenje se vrši do nivoa međujezika (*MSIL* odnosno Java bytecode).
  - Sintaksno su oba jezika C-olika. Veoma su slični na početnom nivou razvoja.
- **Razlike**
  - Za razliku od *Java bytecode*-a koji se najčešće interpretira, u .NET MSIL radi se prevođenje u pravo vreme JIT tako da se čuvaju performanse aplikacije.
  - C# ima više osnovnih tipova podataka. C# sadrži nabrojive tipove i preklapanje operatora za korisničke tipove.
  - C# uvodi novu vrstu objekata tzv. delegate (engl. *delegates*) reference na metode..
  - Potpuna podrška za *unicode*.

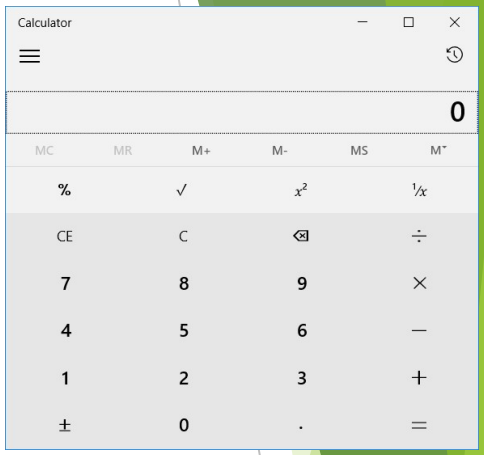
C# vs C++

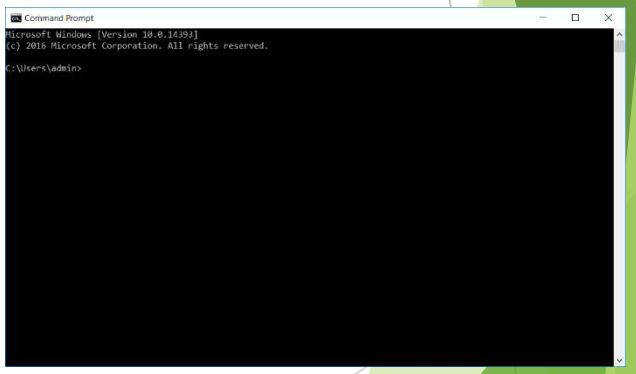


- Sličnosti:
  - Preklapanje operatora
  - Rada sa pointerima koristeći nebezbedni kod tzv. *unsafe*
- Razlike:
  - C# ne zahteva se fajl sa zaglavljima.
  - Preglednost je dobijena primenom editora sa sažimanjem tzv. *folding* editor.
  - C# koristi GC za oslobađanje memorije.
  - Pointeri se mogu koristiti ali samo u celinama označenim kao nebezbedni kod (engl. *unsafe*).
  - Potpuna podrška za *unicode*.
  - Kod C# sve je smešteno u klasama.
  - Kod C# sve je izvedeno iz klase *Object*.

KONCEPT grafičkih aplikacija

- Pogledati primer aplikacije u Win okruženju, recimo Kalkulator, naspram iste aplikacije u konzolnom prozoru.





- Koje su sličnosti i razlike?
  - Ulaz i izlaz
  - Početak i kraj rada aplikacije
  - Logika tj obrada
- Objektno orijentisano programiranje
  - Rad sa objektima, a pre svega
  - Rad sa grafičkim kontrolama
  - Rad sa događajima
  - Rad u grafičkom okruženju

Primer:
Zdravo svete

- U bilo kom editoru unesite sledeći kod:

zdravoSvete.cs

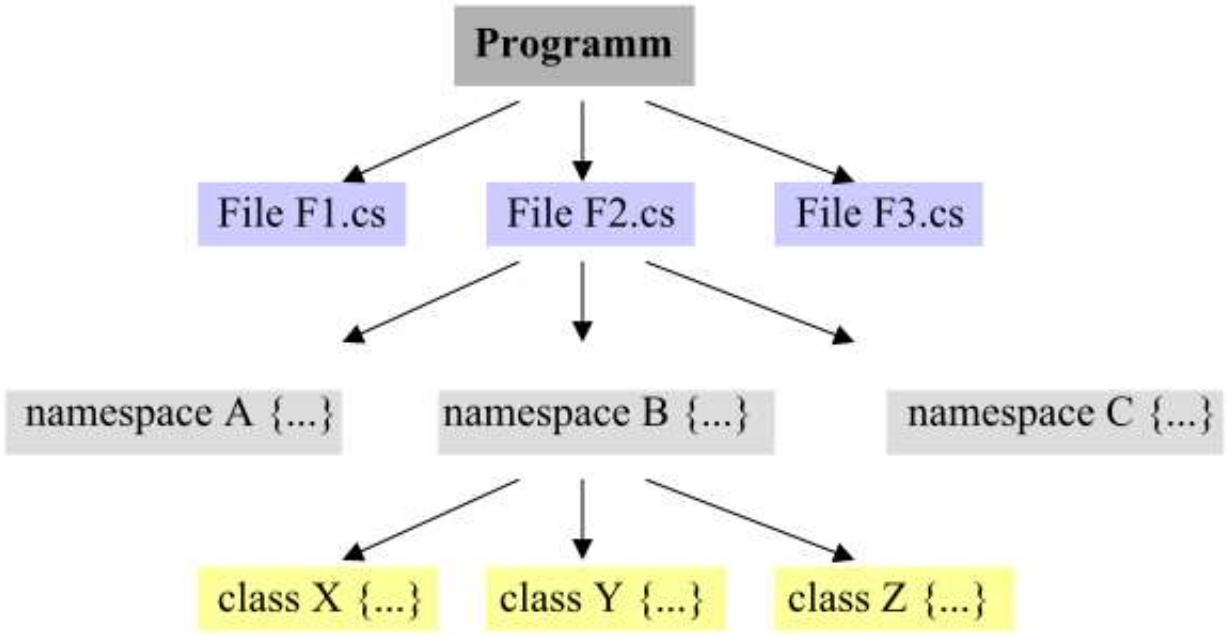
```
1  using System;
2
3  class Zdravo{
4      static void Main(){
5          Console.WriteLine("Zdravo...");
6          Console.ReadLine();
7      }
8  }
```

- System je obavezni imenski prostor koji se mora uključiti u projekat.
- Main – početna/ulazna tačka za program
- Naziv klase i fajla ne moraju biti ista

- Prevođenje:
  - csc zdravoSvete.cs
- Izvršavanje:
  - zdravoSvete

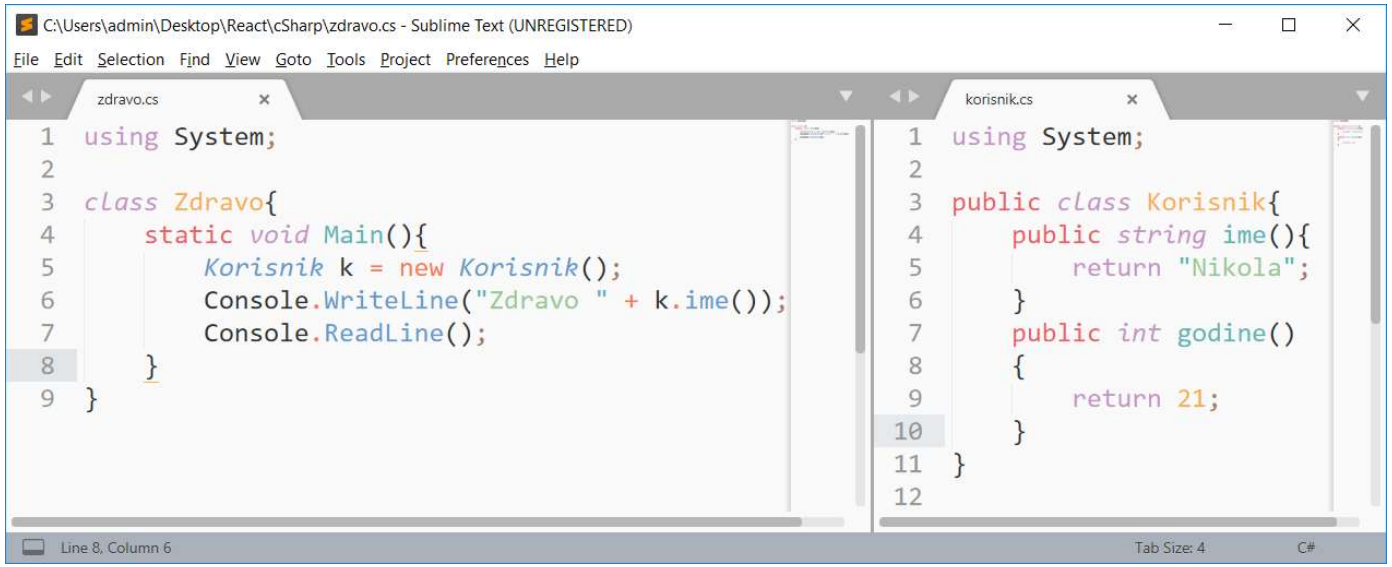
- **Napomena**: Potrebno je startovati konzolni prozor koji će obuhvatiti potrebne alate. Preporučujem jedan od konzolnih prozora koji se dobija iz grupe Visual Studio alata..

Struktura programa



Primer sa 2 fajla

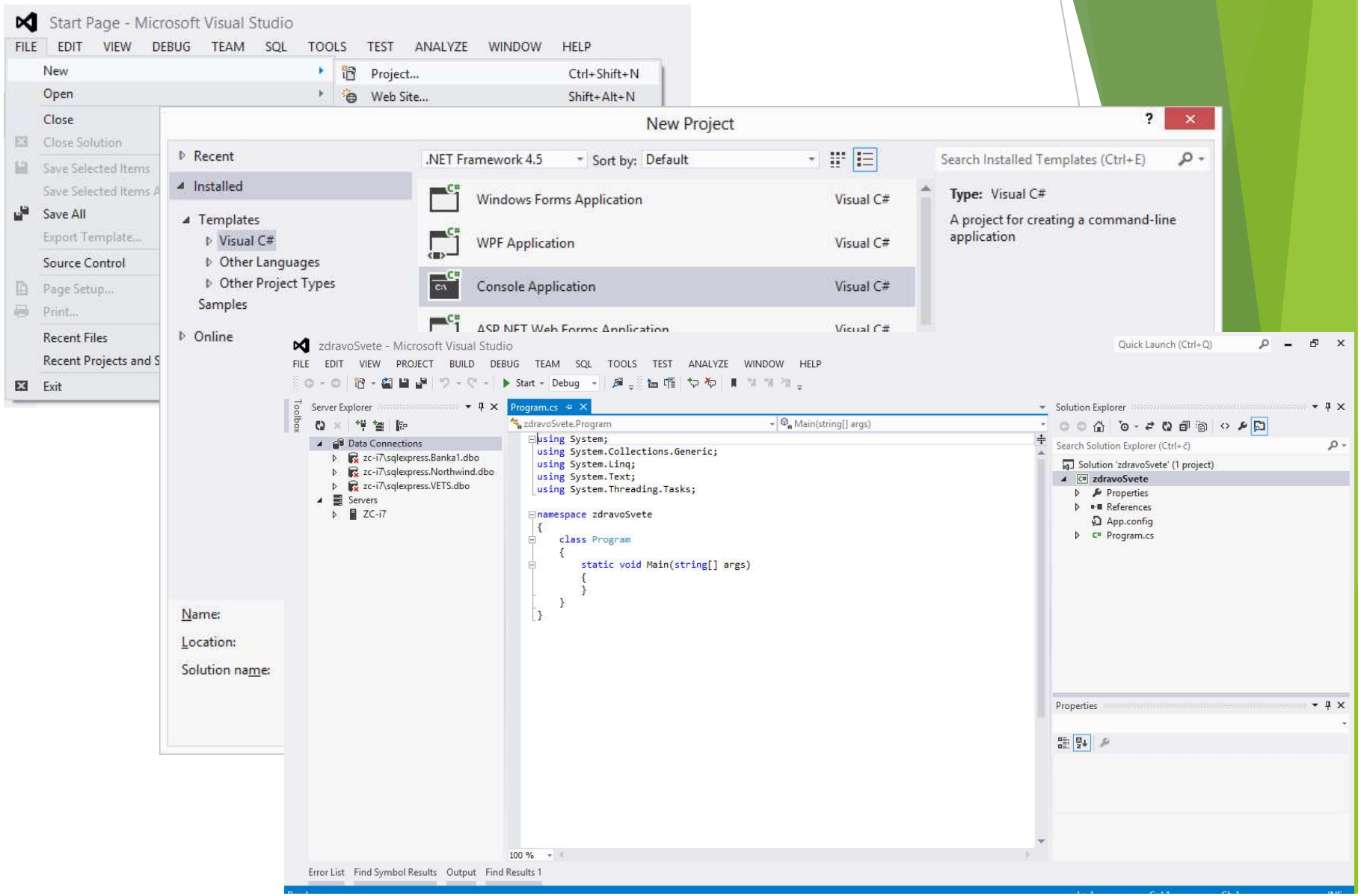
- Ako osim fajla koji obuhvata klasu sa statičkom metodom Main želimo da deo koda držimo u druom fajlu onda je pri prevođenju važno obuhvatiti oba fajla. Na primer:



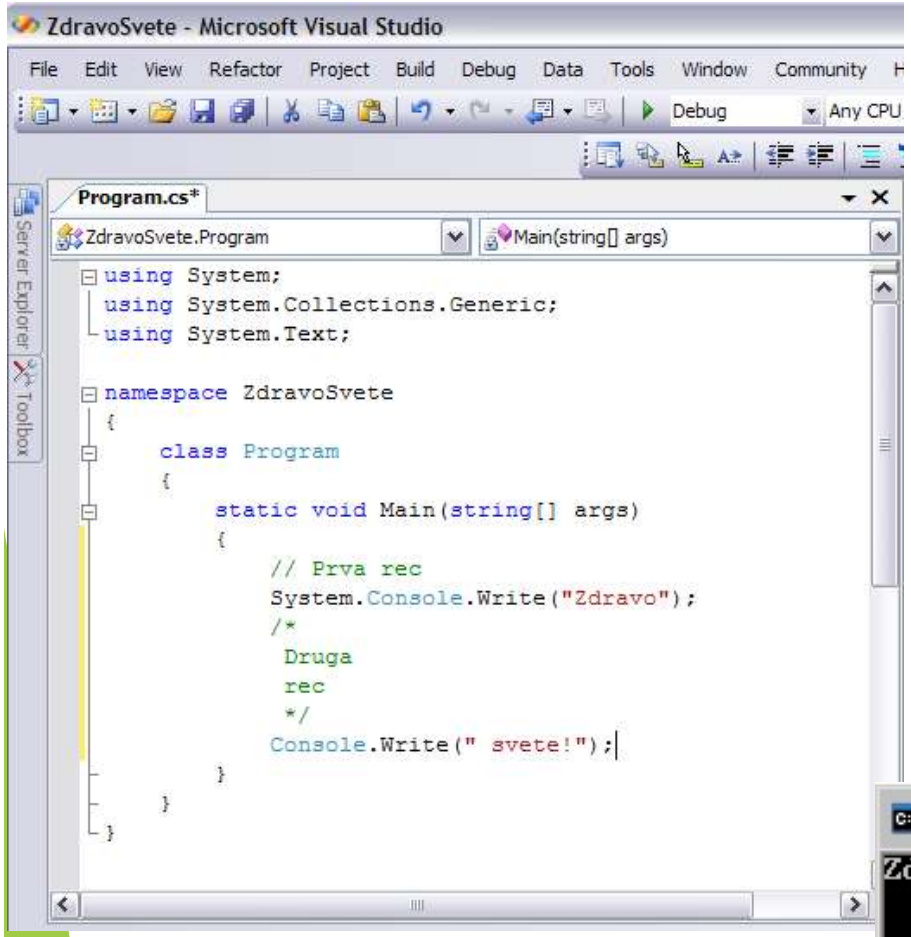
- Prevođenje: csc korisnik.cs zdravo.cs
- Pokretanje: zdravo

- Rad preko DLL biblioteke:
- Kreiranje dll-a: csc /target:library korisnik.cs
- Prevođenje i povezivanje sa dll-om: csc /reference:korisnik.dll zdravo.cs
- Pokretanje: zdravo

Primer – konzolna aplikacija kroz IDE
“Zdravo svete” - Korak 1.



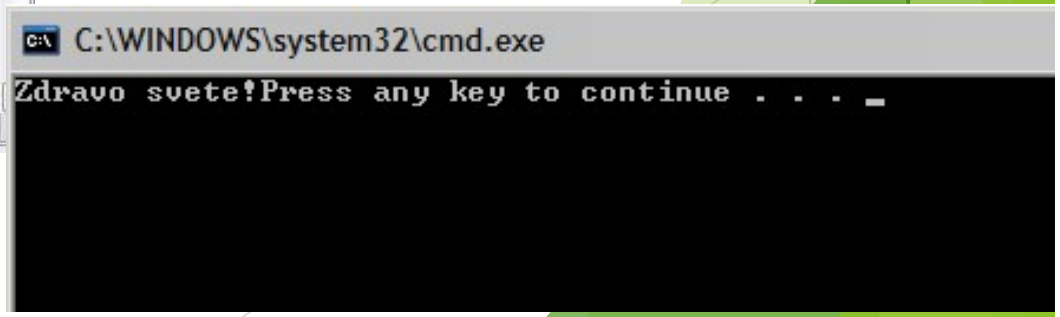
Korak 2.



Korak 3.

- Pokretanje zajedno sa prevođenjem

- **Napomena:** Obratite pažnju kako se jednostavno radi sa velikim broje fajova u projektu tj. lako se vrši održavanje projekta.



Deklaracija i inicijalizacija promenljive u C#

- Deklaracija: *int x;*
- Inicijalizacija: *x = 10;*
- Deklaracija sa inicijalizacijom: *int x = 10;*

Pojmovi "*polje*" odnosno "*lokalna promenljiva*"

- **Lokalna promenljiva** je definisana u okviru neke metode.
- **Polje** je promenljiva koja se definiše na nivou klase.

Inicijalizcija

- **Inicijalizacija je obavezna!** (za razliku od C++)
- **Implicitna i eksplicitna inicijalizacija**
- Promenljive koje se ne inicijalizuju, a predstavljaju polja neke klase ili strukture se implicitno inicijalizuju nakon stvaranja izjednačujući se sa nulom.
- Lokalne promenljive nekog metoda se moraju eksplicitno inicijalizovati.

Primer greške neinicijalizacije lok. promenljive

```
class Program
{
    static void Main(string[] args)
    {
        int y = 0;
        int x;
        if (y == 0)
            x = 4;

        Console.WriteLine(x);
    }
}
```

Use of unassigned local variable 'x'

Primer upotrebe lokalne promenljive istog imena kao i polje u klasi.

```
class Program
{
    static int y = 20; // polje
    static void Main(string[] args)
    {
        int y = 0; // lokalna promenljiva

        if (y == 0)
            Console.WriteLine("Vazi y koje je lokalna promenljiva");
        else
            Console.WriteLine("Vazi y koje je polje na nivou klase");

        Console.WriteLine("Staticka promeljiva/polje klase: {0}",Program.y);
    }
}
```

Formatiranje prikaza

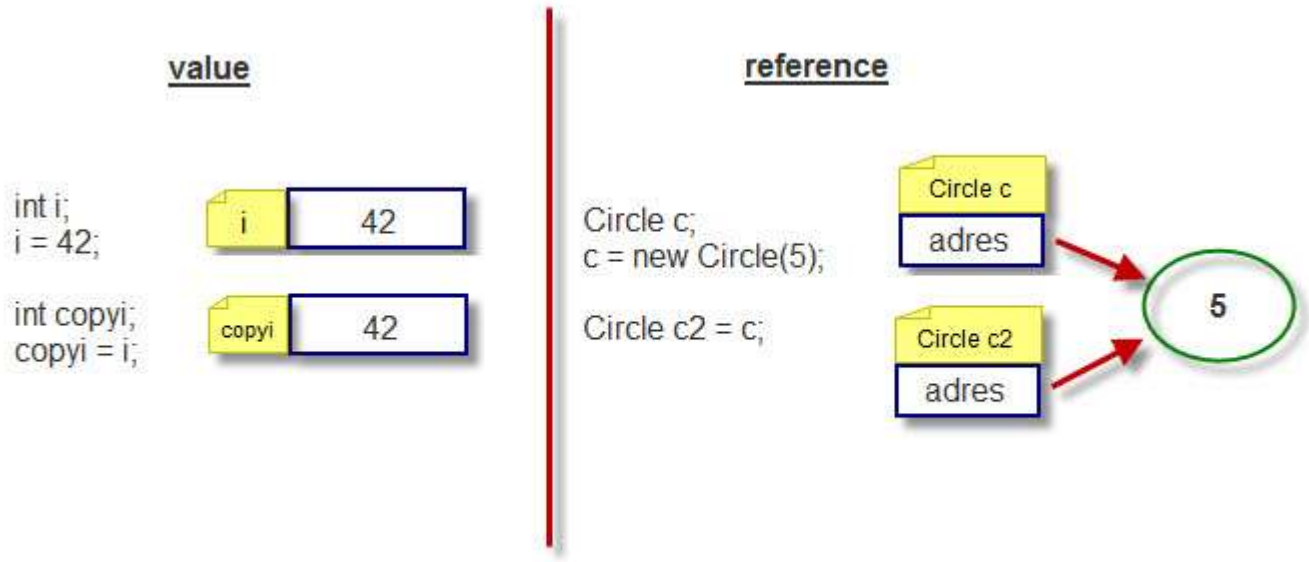
Console.WriteLine(“ a = {0:[FormatCharacter[Number]]}”,a);

- Opciono, može se specificirati karakter za formatiranje kao i broj koji ukazuje odgovarajuće svojstvo formata.

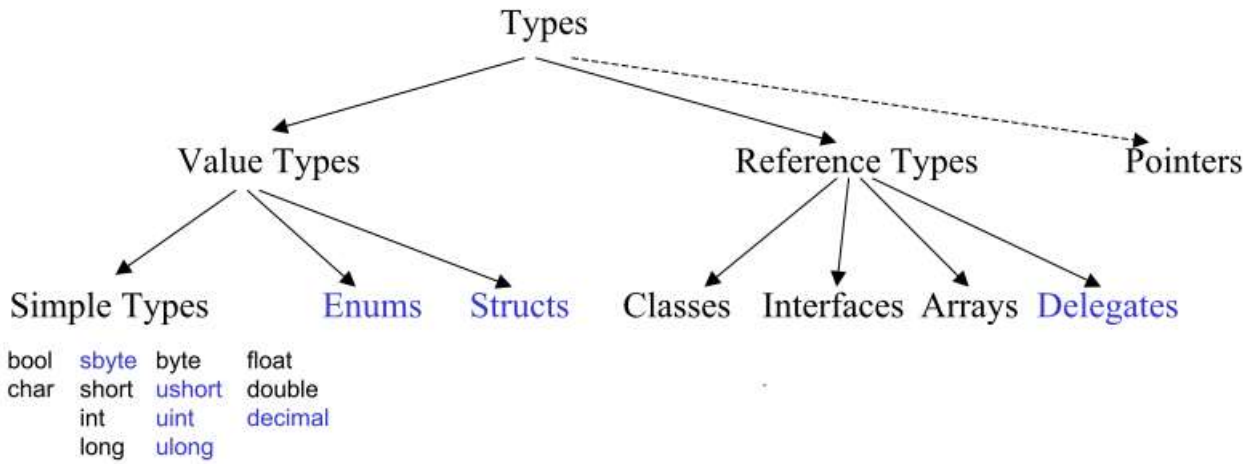
Oznaka	**Formatiranje**
C or **c**	Dodaje znak valute i vrši zaokruživanje na 2 decimale
D or **d**	Decimalni broj, D7 daje 7 mesta za broj
E or **e**	Eksponencijalni zapis.
F or **f**	Floating point numbers, F3 daje 3 decimalna mesta
N or **n**	Osnovni numerički format sa decimalnom tačkom.
X or **x**	Hexadecimalni format.

Vrednosni I referentni tipovi podataka

- Vrednosni:
  - Direktno čuvaju svoje vrednosti.
  - Isto što u VB ili C++ predstavljaju jednostavni tipovi.
- Referentni:
  - Čuvaju samo reference na date vrednosti.
  - Podsećaju na pokazivače u C++.

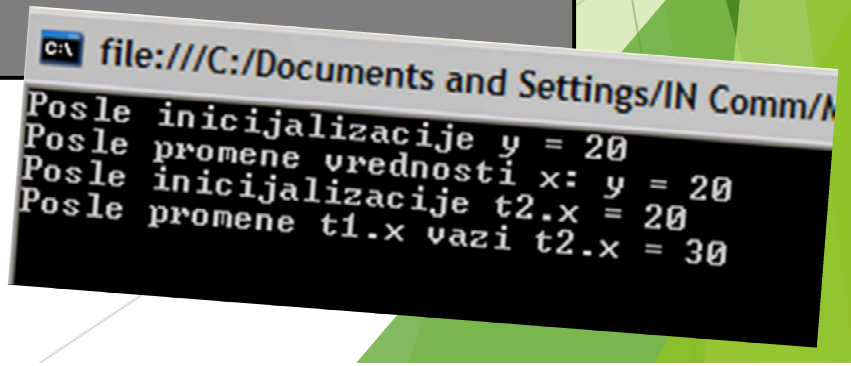


- Grafički prikaz tipova



Primer:

```
class test
{
    public int x;
}
class Program
{
    static void Main(string[] args)
    {
        int x = 20;
        int y = x; // y = 20;
        Console.WriteLine("Posle inicijalizacije y = " + y);
        x = 30;
        Console.WriteLine("Posle promene vrednosti x: y = " + y);

        test t1 = new test();
        t1.x = 20;
        test t2 = new test();
        t2 = t1;
        Console.WriteLine("Posle inicijalizacije t2.x = " + t2.x);
        t1.x = 30;
        Console.WriteLine("Posle promene t1.x vazi t2.x = " + t2.x);
    }
}
```



Predefinisani vrednosni tipovi

- **Celobrojni:**
  - *sbyte*, *short*, *int*, *long*, *byte*, *ushort*, *uint*, *ulong*
- Pokretni zarez:
  - *float* (*System.Single*) (32 bita)

    12.4F (naglašava se float)
  - *double* (*System.Double*) (64 bita)

    12.4 (podrazumeva se double)
- Decimalni tip – pokretni zarez za veće preciznosti, npr. finansijske kalkulacije:
  - *decimal* (System.Decimal) (128 bita) 12.4M
- *char* (znakovni tip)
  - Unicode (16 bitova)
  - Literali ovog tipa se navode između jednostrukih znakova navoda: ‘A’ ili ‘2’
  - Primer:
    - **char a = ‘\u0041’;** //unicode dodele vrednosti:
    - **char a = (char)65;** // integer
    - **char a = ‘\x0041’;** // hexadecimal
  - Specijalne sekvence:
    - \\, \’, \”, \0, \t, \n, \r,..
- bool
  - Nije moguća konverzija u celobrojni tip podataka!?
- Strukture (radićemo kasnije),
- Nabrojivi tipovi ili enumeratori (radićemo kasnije)

Pregled tipova

	Long Form	in Java	Range
sbyte	System.SByte	byte	-128 .. 127
byte	System.Byte	---	0 .. 255
short	System.Int16	short	-32768 .. 32767
ushort	System.UInt16	---	0 .. 65535
int	System.Int32	int	-2147483648 .. 2147483647
uint	System.UInt32	---	0 .. 4294967295
long	System.Int64	long	-2^{63} .. $2^{63}-1$
ulong	System.UInt64	---	0 .. $2^{64}-1$
float	System.Single	float	±1.5E-45 .. ±3.4E38 (32 Bit)
double	System.Double	double	±5E-324 .. ±1.7E308 (64 Bit)
decimal	System.Decimal	---	±1E-28 .. ±7.9E28 (128 Bit)
bool	System.Boolean	boolean	true, false
char	System.Char	char	Unicode character

Referentni tipovi:

- Sve klase koje korisnik definiše,
- *object (System.Object)*
- *string (System.String)*
- Interfejsi,
- Nizovi,
- Delegati.

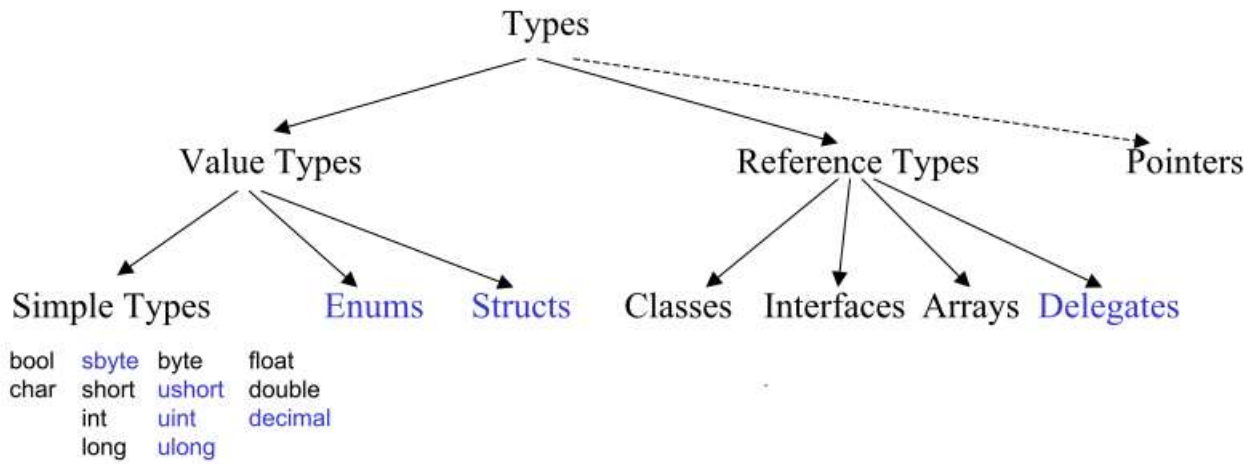
Operator poređenja (==)

- Dve promenljive vrednosnog tipa su iste ako su im vrednosti jednake.
- Dva stringa su ista ako imaju istu dužinu i iste karaktere na istim pozicijama, ili su oba null.
- Dve promenljive referencnog tipa su iste ako pokazuju na isti objekat ili su obe null.

- Napomena: if(a = b) // greska, operator = je operator dodele vrednosti, a ne poređenja.

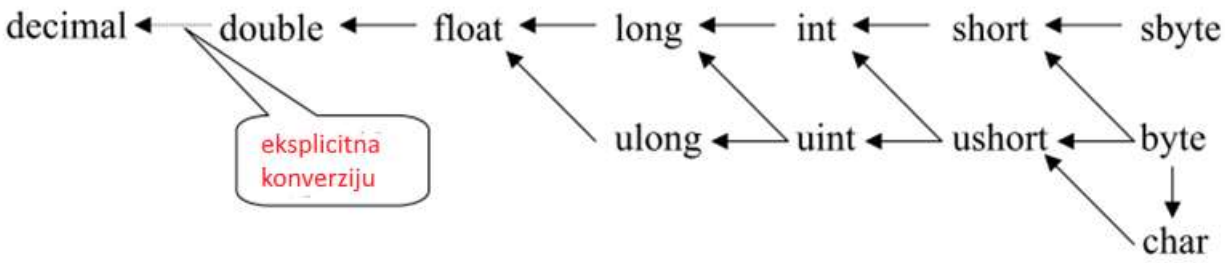
Primer

```
static void Main(string[] args)
{
    string s = "Test";
    string t = string.Copy(s);
    if (s == t)
        Console.WriteLine("t == s");
    if( (object)s != (object)t )
        Console.WriteLine("(object)t != (object)s");
}
```


Types
Value Types
Reference Types
Pointers
Simple Types
Enums
Structs
Classes
Interfaces
Arrays
Delegates
bool
char
sbyte
short
int
long
byte
ushort
uint
ulong
float
double
decimal

Konverzije podataka

- **Implicitna** konverzija – bez gubitaka u tačnosti
  - *int x = 123;*
  - *long y = x;*
- **Explicitna** konverzija
  - *long max = Int64.MaxValue;*
  - *int x = **(int)**max;*



Unificirani tipovi podataka

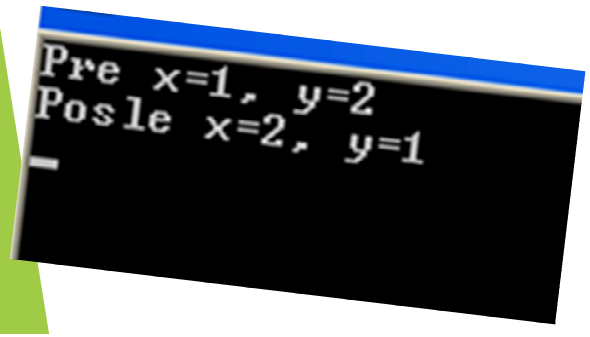
- Svi tipovi (i vrednosni) su izvedeni iz klase **object,** na pr:
  - *Console.WriteLine(**3.ToString()**);*

Prenos parametara metodama

- **Vrednosni:**
  - Vrednost se kopira. Promenljive unutar metode su nove i potpuno nezavisne od onih koje su eventualno korišćene za prenos vrednosti.
  - Isključivo prenost vrednosti KA metodi. Na primer:
    - *public int add(int a, int b);*
- Referencni:
  - Prenosi se samo referenca na promenljivu.
  - Ne kreira se nova promenljiva.
  - Promene u metodi su vidljive i van nje.
  - Ključne reči **ref, out.**

???Primer:

```
class Program
{
    static void Swap(ref int a, ref int b)
    {
        int t = a;
        a = b;
        b = t;
    }
    static void Main(string[] args)
    {
        int x = 1;
        int y = 2;
        Console.WriteLine("Pre x={0}, y={1}", x, y);
        Swap(ref x, ref y);
        Console.WriteLine("Posle x={0}, y={1}", x, y);
    }
}
```



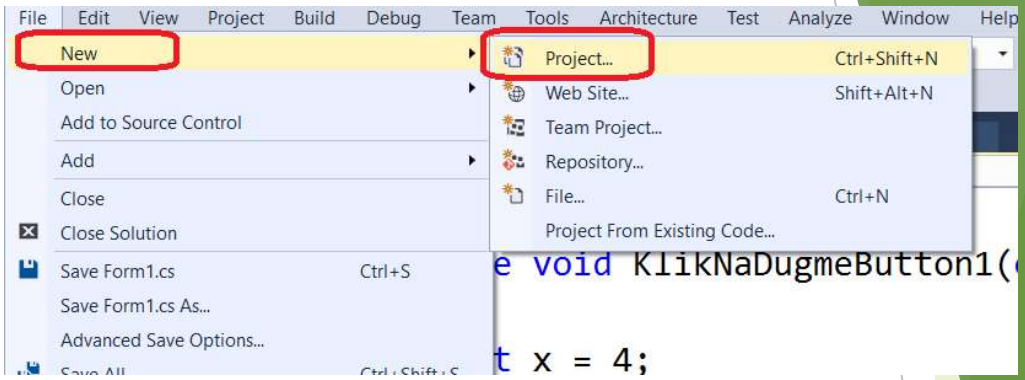
Izlazne promenljive - ključna reč out

- Slične su kao reference.
- Ne zahteva se inicijalizacija.
- Koriste se za izlazne vrednosti.
- Ključna reč **out.**

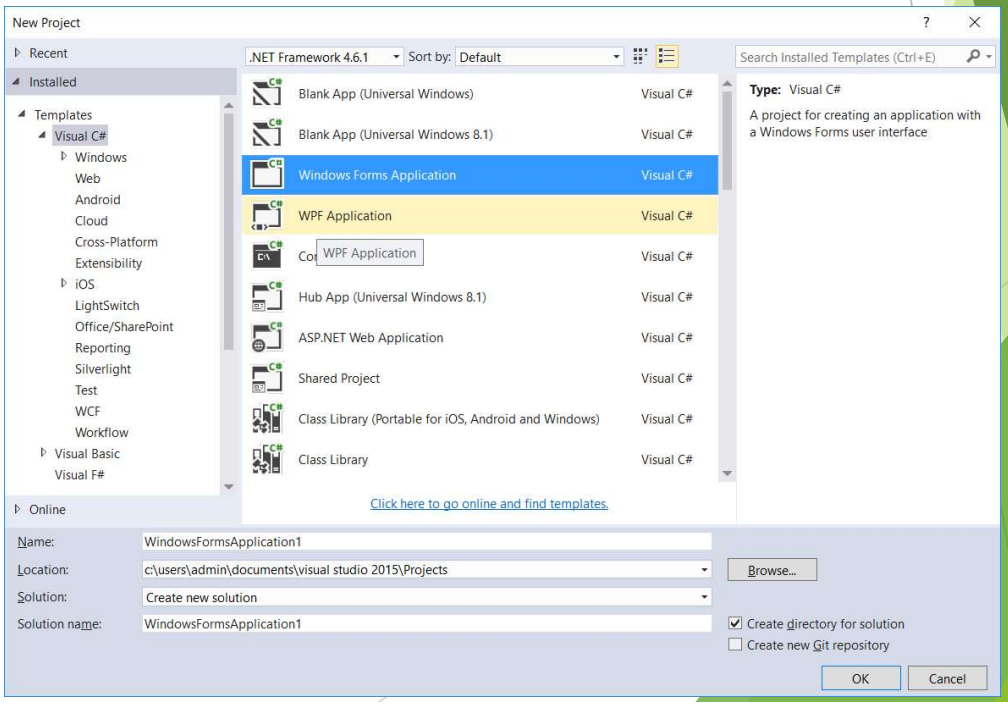
```
class Program
{
    static void Devide(int a, int b, out int res, out int remainder)
    {
        res = a / b;
        remainder = a % b;
    }
    static void Main(string[] args)
    {
        int x = 23;
        int y = 4;
        int res, rem;
        Devide(x, y, out res, out rem);
        Console.WriteLine("Posle res={0}, rem={1}", res, rem);
    }
}
```

Dodavanje projekta:

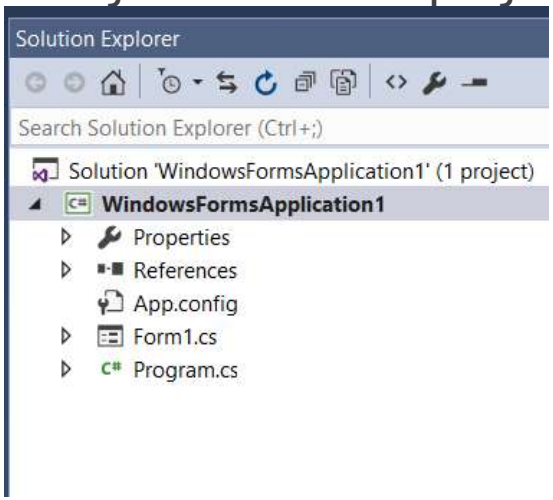
- Moguće je postojećem projektu dodati novi i tako imati u jednom radnom okruženju u okviru jednog rešenja više projekata.



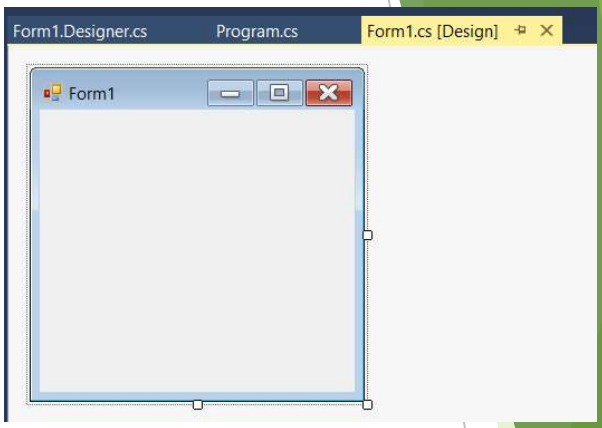
- Izbor tipa projekta.



- Dobijena struktura projekta:



- Izgled početne forme Form1 iz „Dizajnera“:



- Pokretanje aplikacije i startna forma Form1:
- Može postojati samo jedna ulazna tačka za program (ali i više Main metoda po klasama=.

```
static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```

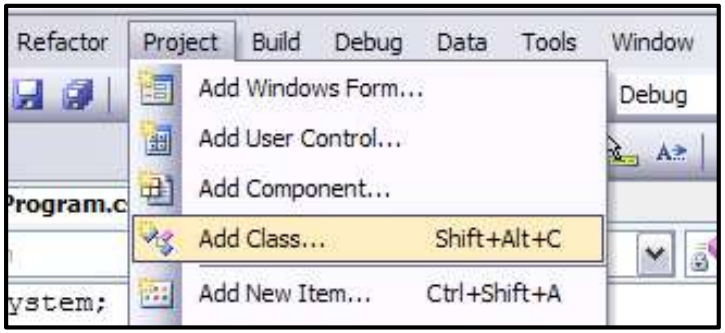
- Početni kod pripremljen za dalje programiranje:

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

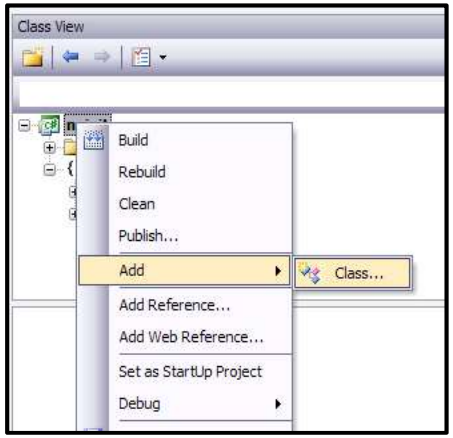
Metoda koja sadži kod „Dizajnera“ u drugom fajlu

Dodavanje nove klase projektu...

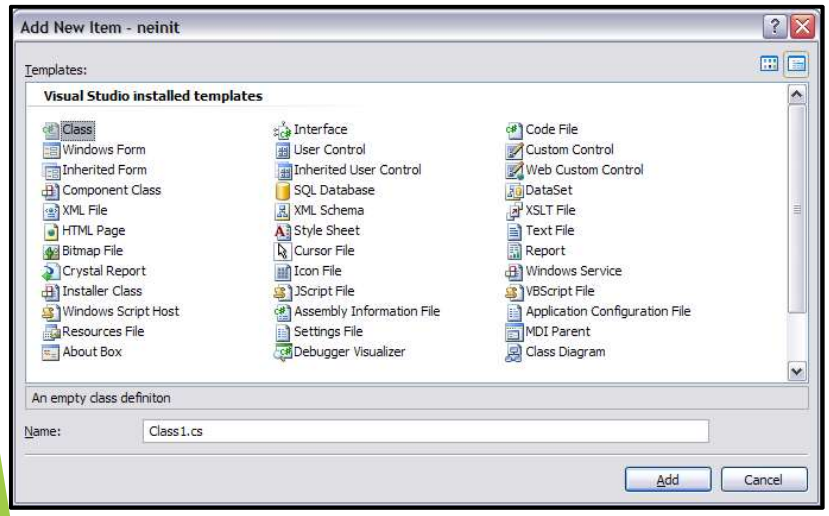
- 1. Iz glavnog menija:



- 2. Iz padajućeg menija Class View:



- 3. Preko stavke u meniju *Add New Item*:



- Podrazumevano pri dodavanju nove klase važi:
  - Nova klasa pripada istom prostoru imena koji je tekući!
  - Nova klasa se smešta u novi fajl sa istim imenom kao što je ime klase.

Operatori i prioritet

Primary	(x) x.y f(x) a[x] x++ x-- new typeof sizeof checked unchecked
Unary	+ - ~ ! ++x --x (T)x
Multiplicative	* / %
Additive	+ -
Shift	<< >>
Relational	< > <= >= is as
Equality	== !=
Logical AND	&
Logical XOR	^
Logical OR	|
Conditional AND	&&
Conditional OR	||
Conditional	c?x:y
Assignment	= += -= *= /= %= <<= >>= &= ^= |=

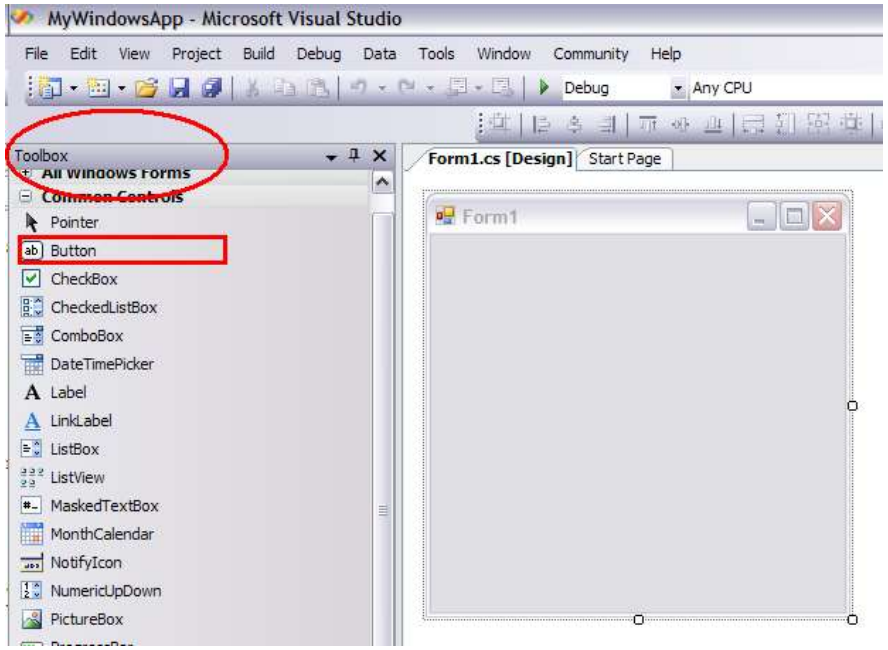
Operatori na istom nivou se izvršavaju s leva na desno

Značenje operatora checked

- checked ključna reč se koristi da eksplicitno omogući proveru prekoračenja opsega za aritmetičke.
- Po podrazumevanoj vrednosti, izraz koji sadrži samo konstantne vrednosti izaziva grešku kompajlera ako izraz daje vrednost izvan opsega tipa odredišta. Ako izraz sadrži jednu ili više nekonstantnih vrednosti, kompajler ne detektuje prekoračenje. Vrednovanje izraza dodeljenog i2 u sledećem primeru ne izaziva grešku kompajlera.

```
static void Main(string[] args)
{
    int x = 100;

    // nema provere
    Console.WriteLine("2147483647 + x");
    Console.WriteLine(2147483647 + x);

    // provera
    Console.WriteLine("checked(2147483647 + x)");
    Console.WriteLine(checked(2147483647 + x));

    // provera u bloku
    checked
    {
        int r = 2147483647 + x;
        Console.WriteLine(r);
    }
}
```

Dodavanje vizuelnih kontrola iz IDE okruženja



Podrazumevani događaj neke kontrole...

- ... se generiše dvosktrukim klikom na tu kontrolu. Za dugme to je klik mišem na dugme.
- *Ovo je dovoljno dok ne budemo radili događaje posebno*

Generisani kod!

Form1.cs | Form1.cs [Design] | Start Page

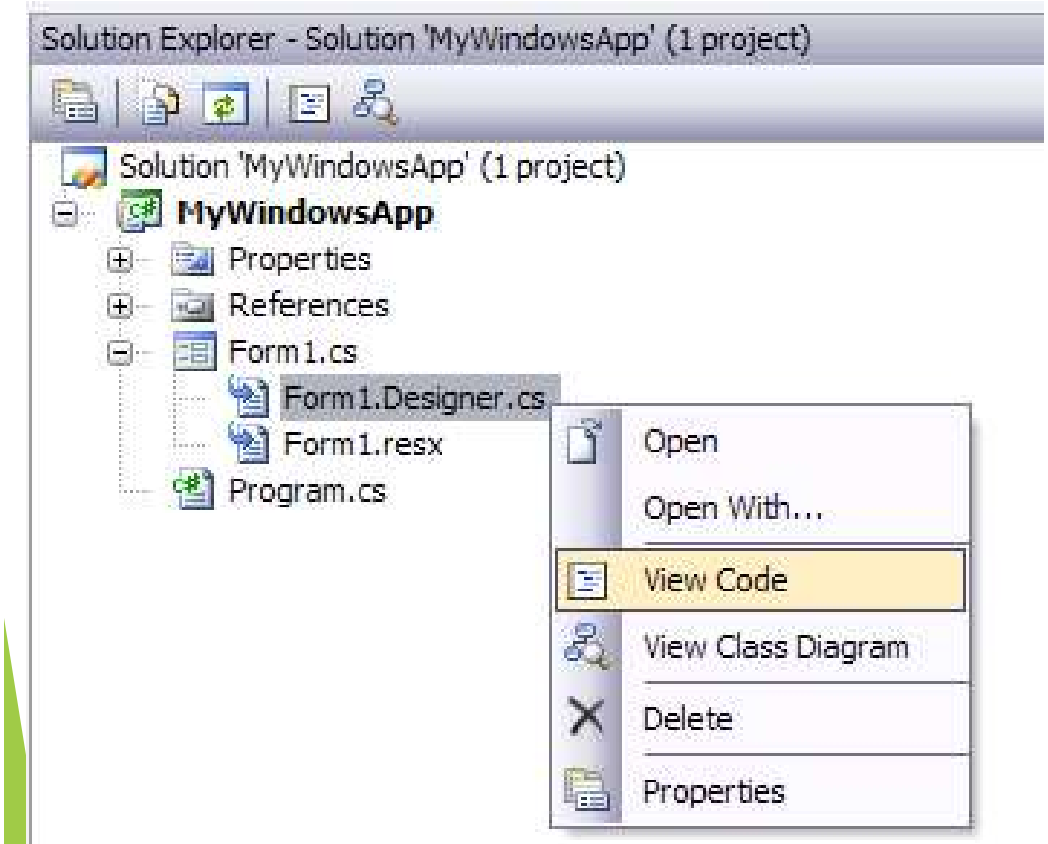
MyWindowsApp.Form1 | button1_Click(object sende

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

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

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovo je odziv na klik misa");
        }
    }
}
```

Gde je dugme?

- Sve što uradite koristeći pomoće alate IDE okruženja rezultuje kodom.
- Neki delovi koda su sakriveni, ali ne i nedostupni.
- Namenjeni su prvenstveno za rad iz **dizajnera.**

Pogled na prozor “Solution Explorer”



Klasa u dva fajla!!

```
        base.Dispose(disposing);
    }

    Windows Form Designer generated code
}
#region Windows Form Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(110, 84);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 0;
    this.button1.Text = "button1";
    this.button1.UseVisualStyleBackColor = true;
    this.button1.Click += new System.EventHandler(this.button1_Click);
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
...
```