

# **SKRIPTA IZ PROGRAMIRANJA**

## **ZA UČENIKE TREĆEG RAZREDA ELEKTROTEHNIČKIH ŠKOLA**

# SADRŽAJ

1. Objektni jezik i C jezik, šta je isto, a šta različito .....	1
1.1. C# programski jezik.....	1
1.2. Radno okruženje (konzolna aplikacija) .....	2
1.3. Osnove jezika C#.....	6
1.3.1. Tipovi podataka .....	6
1.3.2. Razlika između vrednosnih i referentnih tipova .....	6
1.3.3. Promenljive i konstante .....	9
1.3.4. Operatori .....	10
1.3.5. Naredbe.....	12
2. Osnovni koncept objektno orijentisanog programiranja .....	22
2.1. Klasa i objekat.....	23
2.1.1. Klasa. Sastavni elementi klase .....	23
2.1.2. Objekat .....	25
2.2. Modifikatori pristupa.....	26
2.3. Enkapsulacija .....	27
2.4. Nasleđivanje .....	29
2.4.1. Izvedena klasa .....	29
2.5. Polimorfizam .....	33
2.5.1. Virtual.....	33
2.5.2. Overriding.....	34
2.6. Objektni jezik C#.....	35
2.6.1. Definisanje nizovnih promenljivih i korišćenje sistemskih metoda za rad sa nizovima.....	36
2.6.2. Dvodimenzionalni i višedimenzionalni nizovi.....	37
2.6.3. Kolekcije .....	38
2.6.4. Definisanje nabrojivog i strukturnog tipa.....	41
2.6.4. Ključna reč <i>static</i> .....	43
2.6.5. Metode.....	44
3. Klase .....	46
3.1. Razlika između klase i strukture .....	46
3.2. Konstruktor i destruktur .....	46
3.2.1. Konstruktori.....	47
3.2.2. Destruktori .....	50
3.3. Ključna reč <i>this</i> .....	51

3.4. Enkapsulacija podataka .....	53
3.5. Propertiji (svojstva).....	53
3.6. Preklapanje metode u klasi .....	55
3.7. Pojam parcijalne klase.....	55
4. Rukovanje izuzecima .....	56
4.1. Rukovanje, prijavljivanje i prihvatanje (obrada) izuzetaka .....	56
4.2. Često korišćene klase izuzetaka .....	59
5. Izvedene klase .....	61
5.1. Izvedene klase .....	61
5.2. Apstraktna klasa .....	61
5.3. Interfejsi i nasleđivanje interfejsa .....	63
6. Biblioteka komponenata .....	64
6.1. Izrada projekta – Kreiranje WinFoms aplikacije u Visual Studio .....	64
6.2. Forma – Definisanje projekta kao WinForms .....	65
6.3. Izgled radne površine .....	66
6.4. Otvaranje prozora za pisanje koda .....	67
6.5. Događaji .....	68
6.5.1. Događaji miša.....	69
6.5.2. Događaji tastature.....	70
6.6. Dodavanje novih <i>item</i> -a u projekat.....	70
6.7. <i>Build</i> i <i>run</i> aplikacije .....	71
6.8. Metode.....	71
6.9. Windows kontrole .....	71
6.9.1. Kontrola Button .....	72
6.9.2. Kontrola RadioButton .....	80
6.9.3. Kontrola CheckBox .....	87
6.9.4. Kontrola Label .....	95
6.9.5. Kontrola TextBox .....	101
6.9.6. Kontrola RichTextBox.....	109
6.9.7. Kontrola ListBox.....	118
6.9.8. Kontrola ComboBox .....	126
6.9.9. Kontrola PictureBox.....	133
6.9.10. Kontrola GroupBox.....	140
6.9.11. Kontrola Panel .....	146
6.9.12. Kontrola Timer.....	153
LITERATURA.....	155

# 1. Objektni jezik i C jezik, šta je isto, a šta različito

Za realizaciju gotovo svih nastavnih sadržaja iz ovog predmeta koristiće se Microsoft Visual Studio. Svi primeri su rađeni u Microsoft Visual Studio 2019. Na slici 1 možete videti ikonicu Microsoft Visual Studio 2019.



*Slika 1. Ikonica Microsoft Visual Studio 2019*

## 1.1. C# programski jezik

C# je jednostavan objektnoorijentisan programski jezik opšte namene. Razvio ga je Microsoft tim, koji je vodio Andres Hejlsberg. Poslednja verzija C# je 4.5, koja je završena 15. avgusta 2012. godine. Prva verzija (C# 1.0) se pojavila 2001. godine, pa su se ubrzo pojavljivale nove verzije ovog programskog jezika. C# predstavlja naslednika C i C++ jezika, dobio je ime sharp koje je inspirisano muzičkom notacijom i znači da se napisana nota izvodi za pola koraka više. C# je naprednija verzija C++ (C++++). Fajlovi pisani u ovom jeziku imaju ekstenziju cs.

Bitno je napomenuti da C# programski jezik sintaksno nije složen (ima oko 80 rezervisanih reči), ali je vrlo izražajan u delu gde je potrebno rešiti bilo kakav problem softverskog development procesa. Podržava strukturirano, objektno orijentisano programiranje zasnovano na komponentama, u potpunosti podržava sve principe OOP i SOA, podržava interfejse nasleđivanja, strukture, delegate, komponentnoorijentisane elemente (svojstva, događaji, deklaracije).

Pored mogućnosti koje se tiču same implementacije programskih jezika, bitno je napomenuti da C# podržava i sledeće:

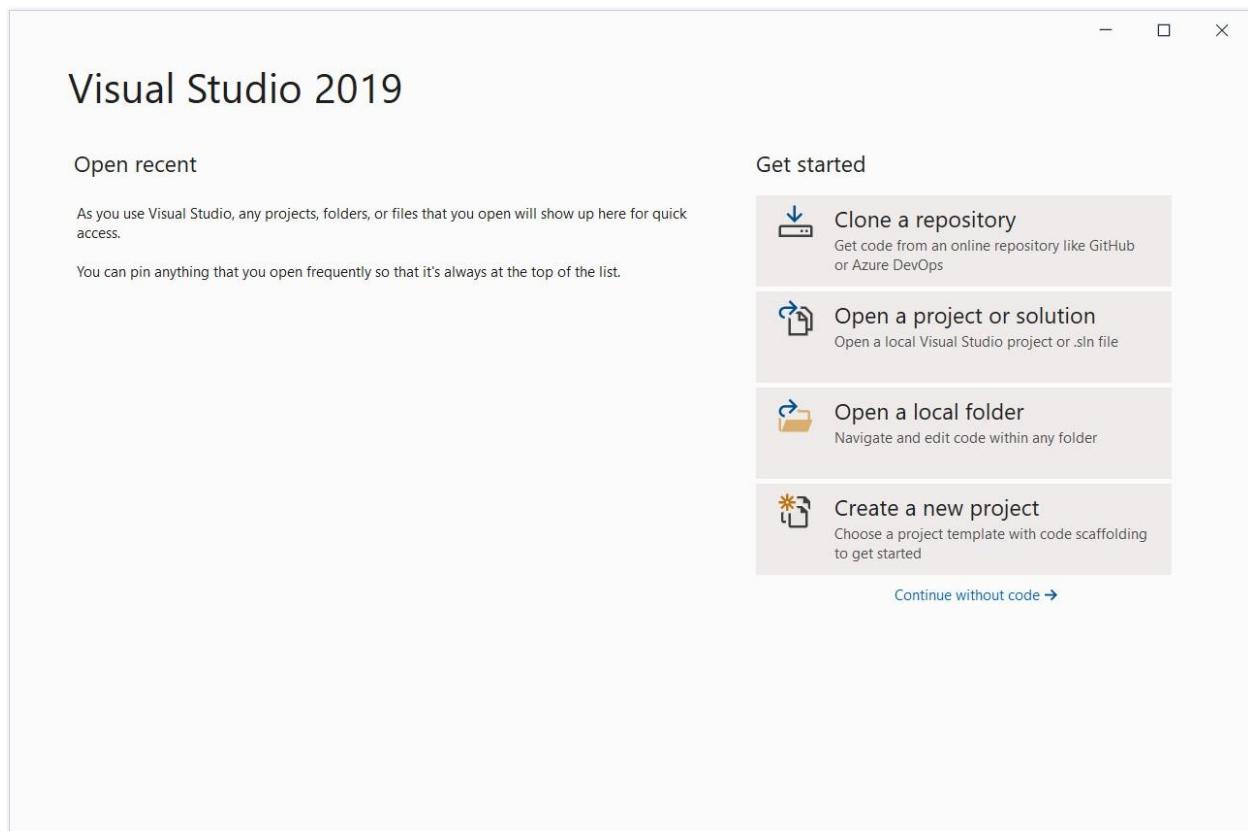
- Direktan pristup memoriji pomoću pokazivača u stilu C++ - a
- Rezervisane reči za izdvajanje nesigurnih operacija
- Upozoravanje skupljača smeća okruženja CLR da ne uništava objekte na koje pokazuju pokazivači dok se ti objekti ne oslobole, itd.

Pored svega gore navedenog, C# podržava još mnogo elemenata programiranja, ali to u ovom delu nećemo razmatrati, jer nam nije cilj pobrojavanje elemenata koje C# može da podrži nego shvatanje njegove moći.

## 1.2. Radno okruženje (konzolna aplikacija)

Po tradiciji izučavanja svih programskih jezika, prilikom realizacije prve aplikacije napisaćemo program za ispis pozdravne poruke „ZDRAVO SVETE“.

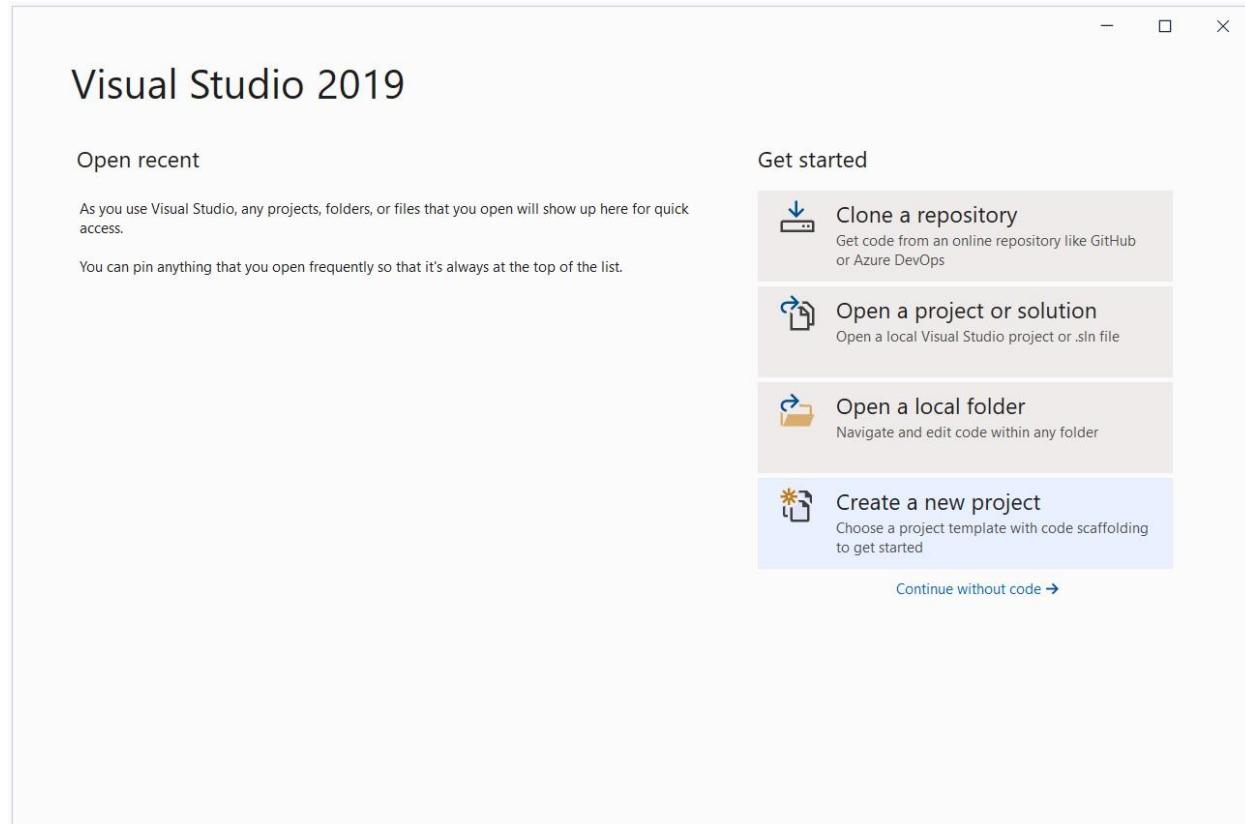
Prvo je potrebno pokrenuti Microsoft Visual Studio 2019, slika 2.



Slika 2. Izgled Microsoft Visual Studio 2019

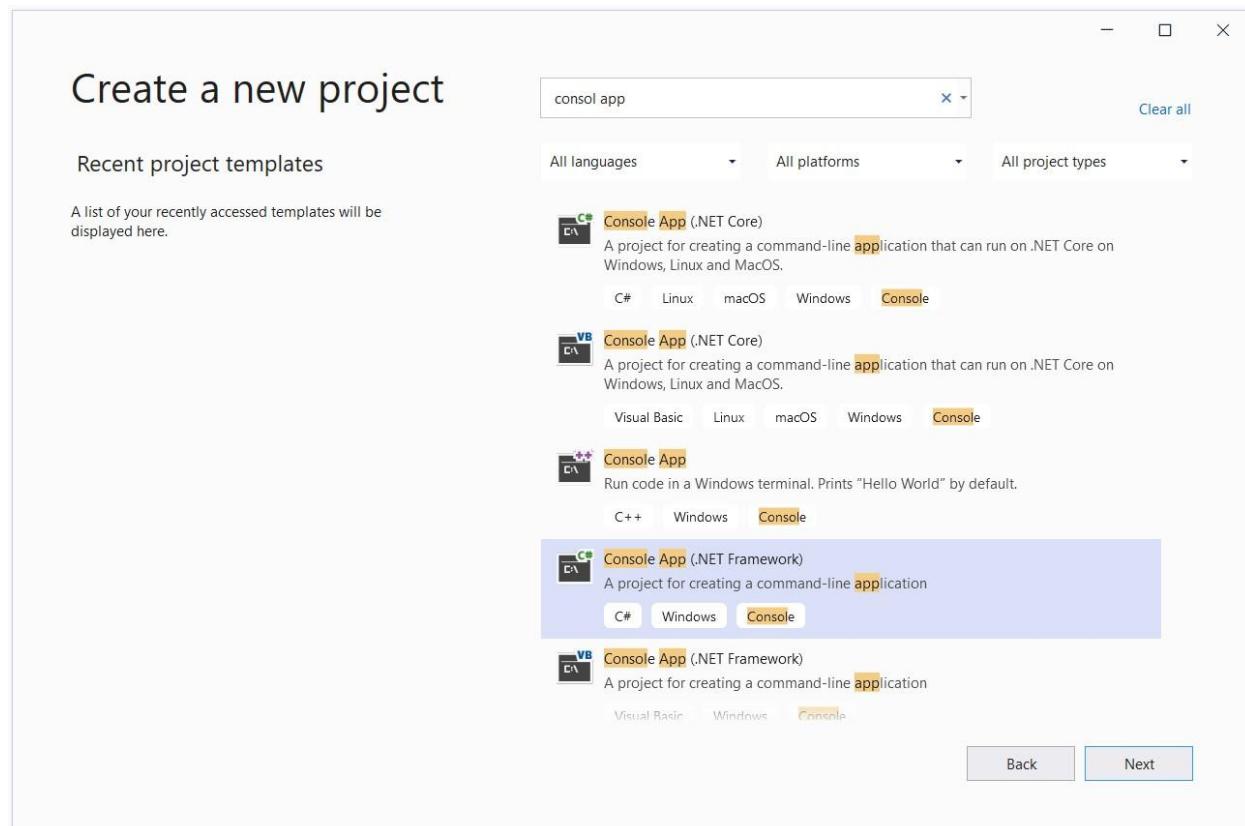
Da biste kreirali aplikaciju ZDRAVO SVETE, potrebno je da kreirate konzolnu aplikaciju. Rad sa konzolnim aplikacijama biće obrađen kasnije, ali u ovom delu će biti prikazan način kreiranja ZDRAVO SVETE aplikacije.

Postupak je sledeći: Na slici 3 je prikazana startna strana, na kojoj se mogu odabrati ponuđene mogućnosti za formiranje novog projekta izborom opcije *Create a new project*.



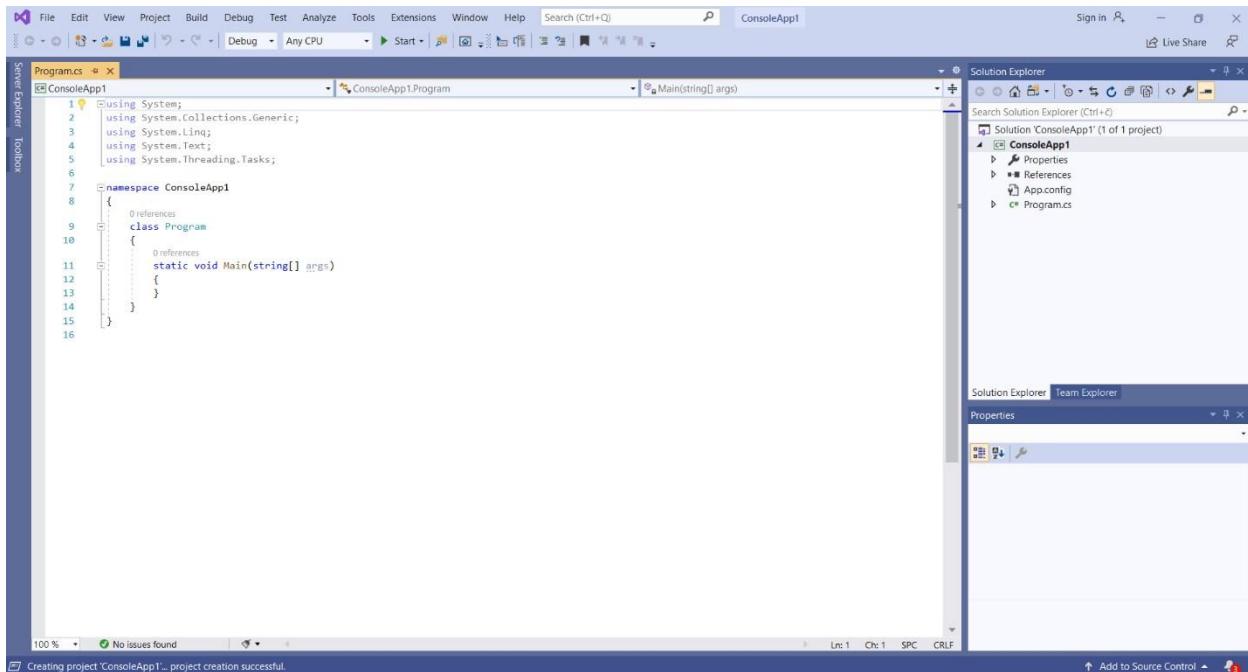
*Slika 3. Kreiranje projekta*

Zatim se u okviru panela Templates izabere Console App (.NET Framework), slika 4.



*Slika 4. Izbor Console App (.NET Framework)*

Nakon uspešnog kreiranja konzolne aplikacije, izgled Microsoft Visual Studio 2019 izgleda kao na slici 5.



*Slika 5. Izgled konzolne aplikacije “ConsoleApp1”*

Nakon unosa imena, lokacije i solution-a, okruženje vam automatski kreira kod koji je dovoljan za pokretanje jedne konzolne aplikacije (listing koda vam je dat ispod).

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class
Program
{
    static void Main(string[] args)
    {
    }
}
```

Da bismo razumeli šta nam je okruženje generisalo u kodu, daćemo i kratak opis koda: *namespace* - predstavlja .NET način da se kod i njegov sadržaj jedinstveno identifikuju. Takođe se koristi za kategorizaciju elemenata u .NET Framework. Namespace je deklarisan za kod aplikacije koja se u ovom slučaju naziva ConsoleApp1.

`using System;` – predstavlja korišćenje System namespace (odnosi se na biblioteku klasa .NET Framework, čiju biblioteku koristi C#). Za veće projekte može se kreirati sopstveni namespaces. Analogno sa ovim objašnjnjem su i objašnjnenja za Collections.Generic, Linq i Text. `class Program {` – definicija klase Primer. Početak klase počinje sa otvorenom vitičastom zagradom {}, a završava se sa zatvorenom vitičastom zagradom }. `static void Main()` – počinje glavni metod, gde se izvršava program. Ovaj metod počinje sa static, što znači da je statički (ne može da se instancira, odnosno ne može od njega da se napravi objekat) i void - ne vraća vrednost.

Kako bismo napravili naš prvi program, potrebno je da uradimo malo editovanje koda i da naš program nateramo da izvrši ispis na ekran. To ćemo uraditi sledećom naredbom:

`Console.WriteLine("ZDRAVO SVET.");` – objašnjnenje: WriteLine() je funkcija koji kao rezultat vraća string koji mu je prosleđen. `Console` - podržava konzolni I/O. U kombinaciji se kompjajleru govori da je `WriteLine()` članica `Console` klase. Izrazi u C# se završavaju sa tačka sa zapetom (;).

Kada bismo pokrenuli naš program, verovatno bismo videli da se podigla konzola, a zatim odmah i zatvorila. Razlog ovoga je to što smo rekli da se izvrši ispis na ekran, što je računar i uradio i odmah nakon toga je nastavio dalje. Kako nema drugih instrukcija, a radi se o `main` metodi, vraća `int` vrednost (u default slučaju 0) i zatvara konzolu. Da bismo predupredili ovakvo ponašanje, potrebno je da zahtevamo da se desi neka akcija od strane korisnika. Najjednostavnije je da tražimo unos nekog karaktera. To ćemo postići na sledeći način:

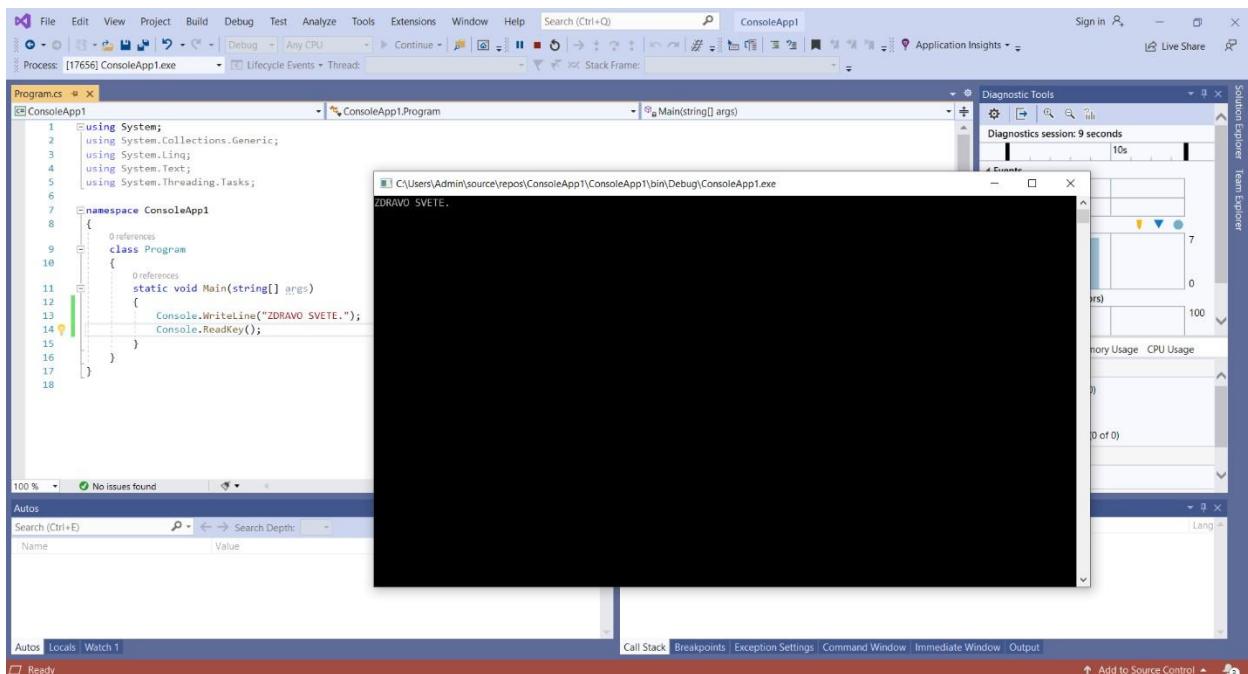
`Console.ReadKey();` – objašnjnenje: funkcija za unos nekog znaka sa tastature.

Sa ovakvim izmenama, naš kod izgleda ovako:

```
using
System;
using System.Collections.Generic;
using System.Linq; using
System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class
Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("ZDRAVO SVETE.");
            Console.ReadKey();
        }
    }
}
```

Da bismo realizovali u potpunosti naš program, potrebno je da ga izgradimo i kompjajliramo. Pokretanje programa je na dugme START. Na slici 6 možete videti rezultat kompjajliranja.



*Slika 6. Rezultat kompjajliranja*

## 1.3. Osnove jezika C#

### 1.3.1. Tipovi podataka

Osnovna podela tipova je na:

- vrednosne (*value*) tipove i
- referentne (*reference*) tipove.

Vrednosni tipovi su:

- jednostavni tipovi (kao što su npr. *byte*, *int*, *long*, *float*, *double*),
- nabrojivi tipovi,
- strukture.

Referentni tipovi su:

- klase,
- interfejsi,
- nizovi,
- delegati.

Svi tipovi (uključujući jednostavne kao što je *int*) su podtipovi *System.Object*

### 1.3.2. Razlika između vrednosnih i referentnih tipova

Postoji nekoliko bitnih razlika u osnovnim karakteristikama između vrednosnih i referentnih tipova, i to:

1. Razlike u alokaciji memorije:

- vrednosni tipovi se čuvaju na steku, odnosno u okviru objekta, a ako su članovi referentnih tipovana, čuvaju se na hipu,
- referentni tipovi se čuvaju na hipu.

2. Razlike u sadržaju:

- vrednosni tipovi sadrže podatak,
- referentni tipovi sadrže pokazivač (referencu) na podatak.

3. Razlike u uništenju:

- vrednosni tipovi se uništavaju odmah pošto se napusti oblast definisanosti,
- referentni tipovi se uništavaju pomoću sakupljača đubreta.

**Tabela 1.** Tipovi podataka u C# programskom jeziku

Tip podatka	Opis	Veličina (bit)	Rang	Primer
<i>int</i>	celi brojevi	32	-2 <sup>31</sup> do 2 <sup>31</sup> -1	<code>int x = 25;</code>
<i>long</i>	celi brojevi (veći rang)	64	-2 <sup>63</sup> do 2 <sup>63</sup> -1	<code>long x = 25L;</code>
<i>float</i>	brojevi u pokretnom zarezu	32	±1.5 x 10 <sup>45</sup> do ±3.4 x 10 <sup>38</sup>	<code>float x = 0.25F;</code>
<i>double</i>	dupla preciznost brojeva u pokretnom zarezu	64	±5.0 x 10 <sup>-324</sup> do ±1.7 x 10 <sup>308</sup>	<code>double x = 0.25;</code>
<i>decimal</i>	novčane vrednosti	128	28 značajnih figura	<code>decimal x = 0.25M;</code>
<i>string</i>	niz karaktera	16 bita po karakteru	nije primenljiv	<code>string x = "pas";</code>
<i>char</i>	jedan karakter	16	0 do 2 <sup>16</sup> -1	<code>char x = 'h';</code>
<i>bool</i>	boolean	8	true ili false	<code>bool x = false;</code>

PRIMERI:

- Ako se kod *double* zaboravi F, kompjajler javlja grešku.

```
double x = 5;    // greška
double x = 5F;  // OK
```

- Sve promenljive se moraju definisati pre upotrebe inače je greška u kompjajliranju.

```
int x, y, z = 6;  
z = x/y;
```

Use of unassigned local variable 'x'

Use of unassigned local variable 'y'

- Tip *var* – govori kompjleru da će pri inicijalizaciji biti odlučeno koji je tačno tip, mora se inicijalizovati odmah pri deklaraciji promenljive.

```
var x; //Error- compile cannot infer type
```

## Konverzija – *Convert*

Sve što se u programskom jeziku C# unosi sa tastature, i sve što se ispisuje na standardnom izlazu je tipa string. Da bi vršili manipulaciju sa različitim tipovima podataka, potrebno je izvršiti konverziju u odgovarajući tip podatka. Za konvertovanje iz jednog u drugi tip se koriste metode predviđene za to. Postoje sledeće konverzije:

1. u *int* – napomena:

- int i Int32 su sinonimi pomoću `int x = Convert.ToInt32(...);`
- pomoću System.Int32.Parse npr: `int x = System.Int32.Parse("50");` ili `string a; int x = int.Parse(a);`

2. u *double* – `int x = Convert.ToDouble(...);`

- pomoću System.Double.Parse npr: `double x = System.Double.Parse("50.0");` ili `string a; double x = double.Parse(a);`

3. u *string* – `string x = Convert.ToString(...);`

4. u *char* – `char x = Convert.ToChar('0'+ 3); //štampa broj 3`

5. ne postoji automatska konverzija `int` u `bool`.

### 1.3.3. Promenljive i konstante

#### 1.3.3.1. Promenljive

Promenljive su osnovni objekti koji se koriste u programima. Promenljiva u svakom trenutku svog postojanja ima vrednost kojoj se može pristupiti i koja se može pročitati i koristiti, ali i koja se (ukoliko nije traženo drugačije) može menjati.

Imena promenljivih su određena identifikatorima. Za identifikator možemo postaviti proizvoljno ime, ali ne sme počinjati brojem, a takođe ne sme biti ni ključna reč.

Posebno treba da voditi računa da je Case Sensitive, odnosno osetljivo na velika i mala slova.

Primer inicijalizacije i dodele vrednosti promenljivoj:

```
int mojaIntPromenljiva = 5;
```

#### 1.3.3.2. Konstante

Konstanta je promenljiva čija se vrednost ne može menjati. Konstanta se dobija stavljanjem ključne reči **const** ispred promenljive prilikom njene deklaracije i inicijalizacije.

```
const int broj = 100;
```

Ograničenja: Kao konstanta se mogu navesti samo lokalna promenljiva ili polje klase. Konstante moraju biti inicijalizovane prilikom deklaracije, i kada im se jednom dodeli vrednost, ona se ne može promeniti. Takođe, konstanta se ne može inicijalizovati vrednošću neke promenljive, odnosno vrednost kojom se inicijalizuje konstanta mora biti dostupna u vreme kompajliranja.

Konstante su uvek implicitno statičke, s tim što nije dozvoljeno navođenje ključne reči *static* u deklaraciji konstante.

Prednosti upotrebe konstanti se ogledaju u tome što se kod lakše razume i modifikuje (promena se vrši samo na jednom mestu prilikom prepravljanja koda).

### 1.3.4. Operatori

U programskom jeziku C# postoji više operatora i to pokazuje upravo njegovu snagu. Operator je funkcija koju primenjujemo nad vrednostima ili promenljivima da bismo dobili željeni rezultat.

#### 1.3.4.1. Operatori dodele vrednosti

Najjednostavniji i najviše korišćeni operator je takozvani operator pridruživanja ili operator dodele vrednosti. Ako imate bilo kakav primer izraza, onda je poenta da se rezultat svih operacija u izrazu dodeljuje ili čuva u odgovarajućoj promenljivoj. Primer:  $y = (m * x) + c;$

Dakle, znak jednakosti (=) predstavlja, u stvari, operator dodele vrednosti. Pored operatora dodele vrednosti postoje još i dodatni operatori dodele vrednosti, a prikazani su u tabeli 2. U tabeli 2 mogu se videti i primeri korišćenja dodatnih operatora dodele vrednosti.

**Tabela 2.** Dodatni operatori dodele vrednosti

OPERATOR	PRIMER KORIŠĆENJA
$+=$	Ako je $X=2$ , vrednost izraza $X+=2$ je $X=4$
$-=$	Ako je $X=2$ , vrednost izraza $X-=2$ je $X=0$
$*=$	Ako je $X=2$ , vrednost izraza $X*=2$ je $X=4$
$/=$	Ako je $X=2$ , vrednost izraza $X/=2$ je $X=1$
$\%=$	Ako je $X=2$ , vrednost izraza $X\%=2$ je $X=0$

#### 1.3.4.2. Aritmetički operatori

Aritmetičke operatore koristim kada želimo da izvršimo određenu matematičku operaciju nad jednim, dva ili više promenljivih, ili matematički rečeno operanata.

**Tabela 3.** Aritmetički operatori

OPERATOR	VRSTA OPERATORA
$+$	sabiranje
$-$	oduzimanje

*	množenje
/	deljenje
%	ostatak pri deljenju

#### 1.3.4.3. Inkrementiranje i dekrementiranje

Inkrementiranje predstavlja aritmetičku operaciju uvećanja promenljive za jedan, a dekrementiranje aritmetičku operaciju umanjenja promenljive za jedan.

NAPOMENA:

Ispis  $x++$  se naziva postincrement. To znači da kompjuter uzima trenutnu vrednost promenljive  $x$  prilikom raznih izračunavanja i tek po završenim izračunavanjima inkrementira promenljivu  $x$  (uvećava za jedan).

Ispis  $++x$  se naziva preincrement. To znači da kompjuter prvo uveća vrednost promenljive za jedan i onda tako uvećanu vrednost promenljive  $x$  koristi u daljim izračunavanjima.

Redosled operatora

Što se tiče redosleda izvršavanja operatora zapamtite da su aritmetički operatori stariji, što znači da se pre izvršavaju u odnosu na operatore poređenja.

#### 1.3.4.4. Relacijski operatori

Relacijske operatore, ili kako još možemo da ih pronademo u literaturi kao operatore poređenja, koristimo kada želimo da tok programa usmerimo zavisno od nekog uslova. Proučite sami tipove operatora poređenja narednih deset sekundi.

*Tabela 4. Relacijski operatori*

OPERATOR	VRSTA OPERATORA
>	veće
<	manje
$\geq$	veće ili jednako
$\leq$	manje ili jednako

<code>==</code>	ekvivalencija (jednakost)
<code>!=</code>	neekvivalencija (različito)

### 1.3.4.5. Logički operatori

*Tabela 5. Logički operatori*

OPERATOR	VRSTA OPERATORA
!	logičko NE (NOT)
<code>&amp;&amp;</code>	logičko I (AND)
<code>  </code>	logičko ILI (OR)

Logički operatori se koriste za izvršavanje operacija Bulove algebre, kao što su endovanje, orovanje, negacija i slično. Logički operator NE primenjen na neku logičku vrednost vraća istu u suprotno stanje.

### 1.3.5. Naredbe

Naredbe u C# predstavljaju celovitu programsku instrukciju i završavaju se (;).

```
int x; x
= 14;
int z = x;
```

#### 1.3.5.1. Naredbe grananja

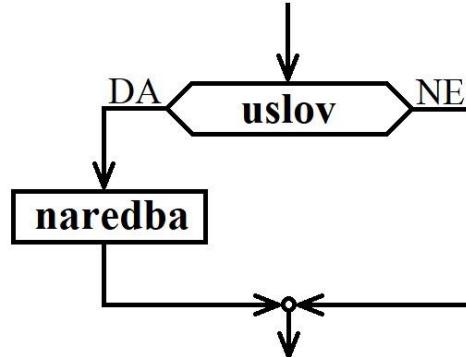
U programskom jeziku postoji više naredbi koje mogu da uslove različite tokove i izvrše grananja. Takođe, postoje i različite petlje koje će omogućiti višestruko prolaženje kroz kod kada je to potrebno. U narednom odeljku dat je prikaz tih struktura.

##### 1.3.5.1.1. Naredba granjanja *if*

Opšti oblik sekcije *if* je isti kao i u programskom jeziku C, jer je programski jezik C# nasledio sintaksu iz programskog jezika C. Opšti oblik sekcije *if* glasi:

**if** (uslov) naredba

Na slici 7. je prikazan dijagram toka if naredbe.



*Slika 7. Dijagram toka if naredbe*

Ako je uslov ispunjen, izvršiće se naredba (grana DA na dijagramu sa slike 7), a ako nije uslov ispunjen, program nastavlja sa radom (grana NE na dijagramu sa slike 7).

Primer – if:

```
if(x >
0)
{
brojac++;
}
```

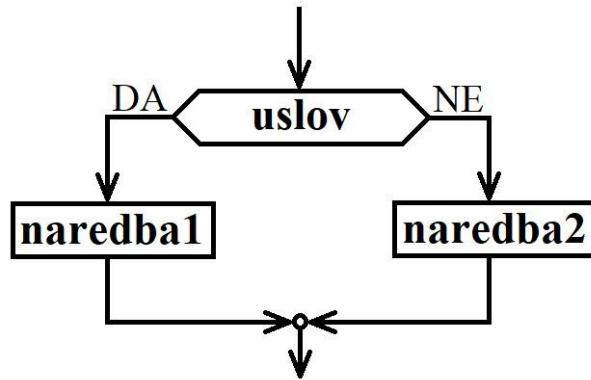
#### 1.3.5.1.2. Naredba grananja if – else

Kao što smo videli za sekciju if, sekcija if-else isto funkcioniše kao u programskom jeziku C.

Opšti oblik sekcijske if-else glasi:

**if** (uslov) naredba1 **else** naredba2

Na slici 8. je prikazan dijagram toka if-else naredbe.



*Slika 8. Dijagram toka if-else naredbe*

Ako je uslov ispunjen, izvršiće se naredba1 (grana DA na dijagramu sa slike 8), u suprotnom izvršava se naredba2 (grana NE na dijagramu sa slike 8).

Primer - *if-else*:

```
if (! (x <= y) && (x >= 0))
{
    rez = 1;
} else
{
    rez = -1;
}
```

SAVET: Dobro je uvek pisati *{}* za if (kao i *while*, *do while*, *for*), jer se može desiti da, kada se naknadno želi dodati deo, kod u selekciji zaboravi da otvoriti i zatvoriti vitičaste zagrade, pa onda kod neće biti korektni, jer će se izvršavati samo prva linija koda.

Primer - kaskadnog *if-else* (tj. naredbe višestrukog granjanja)

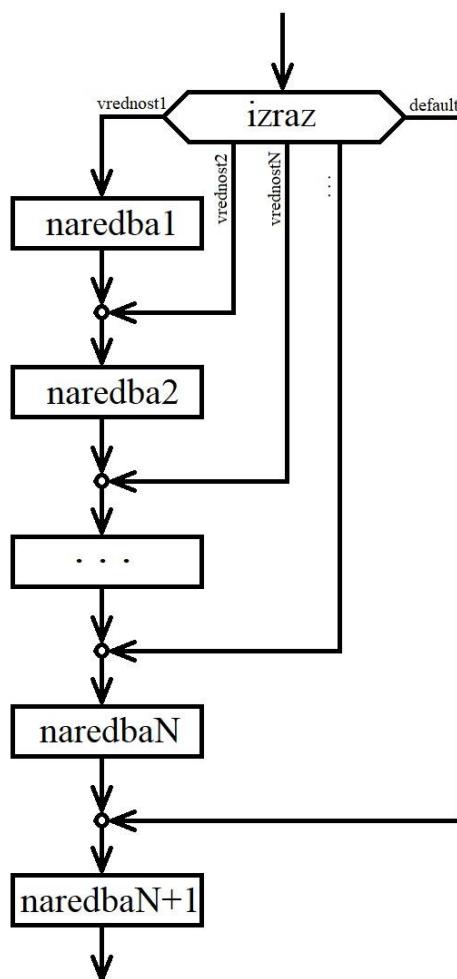
```
if (levaStrana.Godina < desnaStrana.Godina)
rez = -1;
else if (levaStrana.Godina > desnaStrana.Godina)
rez = 1;
else if (levaStrana.Mesec < desnaStrana.Mesec)
rez = -1;
else if (levaStrana.Mesec > desnaStrana.Mesec)
rez = 1;
else if (levaStrana.Datum < desnaStrana.Datum)
rez = -1;
else if (levaStrana.Datum > desnaStrana.Datum)
rez = 1; else
rez = 0;
```

### 1.3.5.1.3. Naredba višestrukog granjanja switch

Opšti oblik *switch* naredbe u programskom jeziku C# je isti kao i u programskom jeziku C, i glasi:

```
switch (izraz)
{
    case vrednost1: naredba1;
    case vrednost2:
        naredba2; . . .
    case
        vrednostN: naredbaN;
    default: naredbaN+1;
}
```

Na slici 9 je prikazan dijagram toka *switch* naredbe.



Slika 9. Dijagram toka switch naredbe

Ako je vrednost celobrojnog izraza *vrednost1*, izvršenje počinje od *naredba1*, ako je *vrednost2* od *naredba2*, itd., ako je *vrednostN* od *naredbaN* i na kraju će se izvršiti *naredbaN+1*. Ako vrednost celobrojnog izraza ne odgovara nijednoj *case* vrednosti, izvršiće se *default* naredba.

Ako ne želimo da nam se naredbe izvršavaju jedna za drugom, već želimo da se izvrši za samo jednu vrednost *case*-a, koristimo rezervisanu reč *break* za izlazak iz naredbe višestrukog grananja (primer korišćenja službene reči *break* se može videti u narednom primeru).

Primer naredbe višestrukog grananja *switch*:

```
switch (pom)
{    case '<':          rezultat.Text += "manje";
    break;    case '>':
rezultat.Text += "vece";    break;
case '&&':          rezultat.Text += "logico i";
break;    case '|':
rezultat.Text += "logicko ili";
break;    case '>=':
rezultat.Text += "vece jednako";
break;    default:
rezultat.Text += pom;    break;
}
```

*break* se koristi za svaki *case* (nije obavezno, ali može da se koristi ako ne želimo da nam se sve naredbe izvrše), i za *default*.

labela – pom:

1. mora biti tipa *int* ili *string*
2. ne smeju biti dve labele sa istom vrednošću
3. labele se ne mogu grupisati, iako izvršavaju iste iskaze
4. vrednost labele se ne može izračunavati za vreme izvršavanja programa 5. ako se hoće više *case* izvršiti, mora se koristiti *goto*

### 1.3.5.2. Petlje (Ciklusi)

Niz naredbi u programu koji se može izvršiti više puta **naziva se petlja (ciklus)**.

- Niz operatora koji obrazuje ciklus nazivamo **telo ciklusa**.
- Uslov koji određuje da li će se telo ciklusa ponovo izvršiti nazivamo **izlazni kriterijum ciklusa**.

U zavisnosti od položaja izlaznog kriterijuma u odnosu na telo ciklusa, ciklusi mogu biti:

- Sa preduslovom (**WHILE**),
- Sa postuslovom (**DO WHILE**),

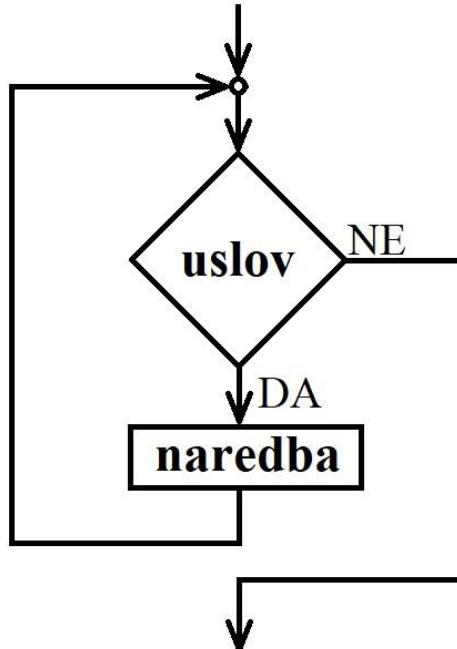
Ciklus **WHILE** se može skraćeno zapisati pomoću operatora ciklusa **FOR**. Ono što nije postojalo u programskom jeziku C, a postoji u C# je **FOREACH**.

#### 1.3.5.2.1. Petlja WHILE

Petlja (ciklus) *while* ima sledeći opšti oblik:

`while (uslov) naredba`

Na slici 10 je prikazan dijagram toka *while* naredbe:



*Slika 10. Dijagram toka while naredbe*

Kod `while` ciklusa naredba će se izvršiti zavisno od uslova (sve dok je uslov ispunjen). Ako je uslov ispunjen, izvršiće se naredba (grana DA na dijagramu sa slike 10), u suprotnom, izvršiće se grana NE (grana NE na dijagramu sa slike 10), i tada se izlazi iz petlje (ciklusa). **While** ciklus se u programiranju najčešće koristi kada ne znamo koliko puta neku radnju želimo da ponavljamo. Ovaj ciklus se naziva još i ciklus sa preduslovom.

Primer `while` ciklusa:

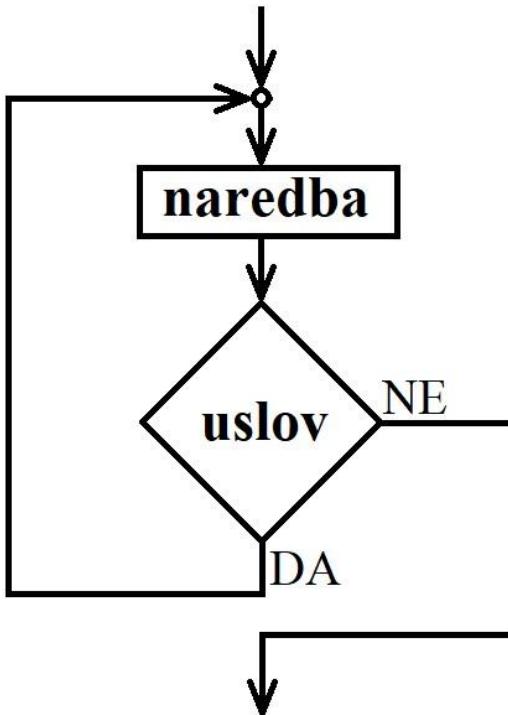
```
int x=10;
while (x>0)
{
    Console.WriteLine(x); x--
}
```

#### 1.3.5.2.2. Petlja DO WHILE

Ciklus *do while* ima sledeći opšti oblik:

`do naredba while (uslov);`

Na slici 11 je prikazan dijagram toka do while naredbe:



*Slika 11. Dijagram toka do while naredbe*

Prvo se izvršava naredba, pa se ispituje uslov. Ako je uslov ispunjen, uradiće se ponovo naredba (grana DA na dijagramu sa slike 11), u suprotnom, izvršiće se grana NE (grana NE na dijagramu sa slike 11), i tada se izlazi iz petlje (ciklusa). Do while ciklusa naredba će se izvršiti sigurno jedanput. Do while ciklus se u programiranju najčešće koristi kada ne znamo koliko puta neku radnju želimo da ponavljamo, kao i while ciklus. Ovaj ciklus se naziva još i ciklus sa postuslovom.

Primer do while ciklusa:

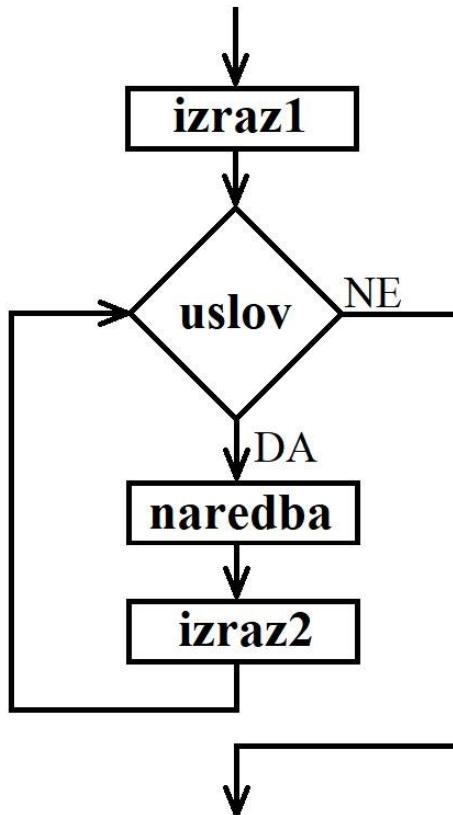
```
int
x=10; do
{
    Console.WriteLine(x); x--
}
while (x>0);
```

### 1.3.5.2.3. Petlja FOR

Petlja (ciklus) `for` ima sledeći opšti oblik:

```
for (izraz1; uslov; izraz2) naredba
```

Na slici 12 je prikazan dijagram toka for naredbe:



Slika 12. Dijagram toka for naredbe

Kod for ciklusa naredba će se izvršiti onoliko puta koliko je zadato pomoću uslova. Izraz1 predstavlja donju granicu od koje počinjemo da brojimo (tj. početnu vrednost brojača), uslov predstavlja gornju granicu do koje brojimo (tj. krajnju vrednost brojača), a izraz2 predstavlja korak brojanja između donje i gornje granice brojanja. Ako je uslov ispunjen, uradiće se grana DA (grana DA na dijagramu sa slike 12), u suprotnom, izvršiće se grana NE (grana NE na dijagramu sa slike 12), i tada se izlazi iz petlje.

Primer for ciklusa:

```
for (int i = 10; i > 0; i-- )  
{  
    Console.WriteLine(i);  
}
```

#### 1.3.5.2.4. Petlja FOREACH

Naredba foreach je nova naredba u porodici C jezika i služi da obezbedi iterativni pristup svim elementima niza ili kolekcije. Najčešće se koristi za prolazak kroz kolekcije.

Petlja (ciklus) *foreach* ima sledeći opšti oblik:

```
foreach (tip identifikator in izraz )
{ naredba
}
```

Dijagram toka za petlju (ciklus) *foreach* je isti kao i za *while* petlju (ciklus).

Primer *foreach* petlje:

Popunjavanje niza i štampanje elemenata niza.

```
int[] niz1 = new int[5]; int[]
niz2 = new int[3];

for (int i=0; i<niz2.Length; i++ )
{ niz2[i] = i + 5;
}
foreach (int i in niz1)
{
    Console.WriteLine(i);
}
foreach (int i in niz2)
{
    Console.WriteLine(i);
}
```

Više o korišćenju i radu *foreach* petlji (ciklusa) radiće se u poglavljiju liste i u narednoj školskoj godini (IV razred) u okviru predmeta Programiranje IV kada budemo radili kolekcije.

### 1.3.5.3. Naredbe skoka

Za rešavanje nekih programskeh problema neophodno je izmeniti tok normalnog izvršenja upravljačke strukture. Za izmenu toka upravljačke strukture koristimo sledeće naredbe:

- *break* • *goto* • *continue*

U programskom jeziku C je sve ovo rađeno u prvom i drugom razredu, a s obzirom na to da je C# nasledio sintaksu od programskog jezika C, pomenute naredbe se isto koriste kao u programskom jeziku C.



## 2. Osnovni koncept objektno orijentisanog programiranja

Tokom prvog i drugog razreda smo radili proceduralno programiranje (programske jezike C).

Počevši od ovog poglavlja pa sve do kraja četvrtog razreda ono što ćemo raditi je objektno orijentisano programiranje (tj. pristup) u C# programskom jeziku.

Proceduralno programiranje ima niz nedostataka u odnosu na objektno orijentisano programiranje, a jedna od najvećih je što imamo mogućnost pravljenja novih tipova, tj. složenih tipova podataka. U proceduralnom programiranju najveći nedostatak je što nemamo mogućnost da ograničimo pristup i/ili vidljivost podataka, za razliku od objektno orijentisanog programiranja koji nam daje tu mogućnost. U objektno orijentisanom programiranju polazi se od objekta sa kojima želimo da manipulišemo, a ne od logike koja je potrebna za tu manipulaciju kao u proceduralnom programiranju.

Definišu se četiri osnovna koncepta (ili principa) objektno orijentisanog programiranja:

- apstrakcija,
- enkapsulacija (učaurenje),
- nasleđivanje,
- polimorfizam.

Apstrakcija je pojednostavljinjanje karakteristika stvarnog objekta, gde se zanemaruju detalji i uzimaju se samo zajedničke karakteristike klase u zavisnosti od potreba programa. Prostije rečeno, predstavlja odabir svojstava – polja koja će opisivati dati objekat. Jedan isti objekat može da se apstrahuje na različite načine.

Enkapsulacija ili učaurivanje je koncept po kojem je informacija u klasi zaštićena od direktnog pristupa i jedini način da se promeni je kroz definisane metode. Ovakav koncept programiranja naziva se „crna kutija“- poznat je ulaz i izlaz, ali ne i proces koji se odvija unutar funkcije.

Enkapsulacija se ostvaruje podelom članova klase na javne i privatne. Javni članovi mogu slobodno da se koriste u bilo kom delu programa (van tela same klase), dok su privatni dostupni samo u okviru klase u kojoj su definisani. Polja su najčešće privatna, dok su metode većinom javne [1].

**Nasleđivanje se odnosi isključivo na klase.** Nasleđivanje je najvažniji koncept objektno orijentisanog programiranja. Po njemu se iz jedne definisane klase (bazne, roditeljske, predak, nadklase) može izvesti nova klasa (izvedena, dete, potomak, potklasa). Ona nasleđuje sve članove

klase i uvodi svoja specifična polja i metode. Ako neki član klase ima modifikator pristupa *private*, nasleđuje se, ali ne može da mu se pristupi u izvedenoj klasi.

**Polimorfizam se odnosi isključivo na metode.** Polimorfizam je osobina da ista metoda deluje različito, u zavisnosti od tipa ili broja parametara koji joj se prosleđuju. Možemo još reći da je polimorfizam osobina da metode ima isto ime u izvedenoj i baznoj klasi, a drugačiju logiku izvršenja.

Uvođenjem objektno orijentisanog programiranja (tj. pristupa), uvedeni su pojmovi klase i objekta.

## 2.1. Klasa i objekat

Pokušajmo da odgovorimo na pitanje: „Šta su to klase i objekti?”

**Klase je struktura podataka koju bismo trebali posmatrati kao novi tip, i ona prestavlja generičku definiciju objekata koji imaju zajedničku strukturu i ponašanje.**

**Nasuprot klesi, objekat je instanca klase i on se definiše kao entitet koji je sposoban da čuva svoja stanja i koji okolini stavlja na raspolaganje skup operacija preko kojih se tim stanjima pristupa.**

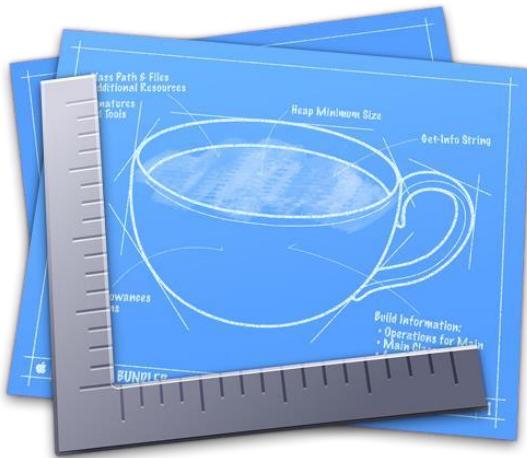
Prema tome, objekat karakterišu:

- identitet – omogućava razlikovanje objekata među sobom,
- ponašanje – dinamički aspekt objekta, definiše se metodama koje sadrži objekat,
- stanje – staticki aspekt objekta, definiše se podacima objekta i njegovim vezama sa drugim objektima u sistemu [1].

### 2.1.1. Klasa. Sastavni elementi klase

Za klasu možemo da koristimo i sledeću definiciju: **Klase je složeni tip podataka koji predstavlja šablon prema kojem se izrađuju objekti. Klasom se definišu osobine i ponašanje tih objekata.**

Na slici 13 je prikazan primer jedne klase (šablon) na osnovu koje će se kreirati objekat.



Slika 13. Primer klase **Soljica** (šablon)

Kada kreiramo neku sopstvenu klasu, potrebno je da definišemo: njena svojstva (properties) ili varijable kojima će dodeljivati vrednosti i metode koje određuju njenu ponašanje.

Naredba za kreiranje nove klase počinje ključnom reči *class*, zatim dolazi naziv klase, deklaracija svojstava (properties) koje se definišu za tu klasu i na kraju metode koje pripadaju klasi.

```
public class Soljica
{
    //deklaracija i inicializacija svojstava ili varijabli
public int mojaVarijabla;

    //metode
    public void mojaMetoda()
    {
...
    }
}
```

Gornjim naredbama kreirali smo klasu Soljica, i njoj pripadajuću celobrojnu varijablu mojaVarijabla kojoj će se unutar klase mojaKlase dodeljivati vrednosti.

U nastavku, još jedan primer kreiranja klase, ovoga puta, klasa Račun [1].

```
public class Racun
{
    private double brojRacuna;
private double stanje;           private
double provizija;
```

```

public void Uplata(double iznos)
{
    ...
}
public void Isplata(double iznos)
{
    ...
}
public void Prikazi()
{
...
}
}

```

### 2.1.2. Objekat

U prethodnom delu smo definisali kreiranje klase i vrlo je bitno da shvatimo da je klasa referentni tip. Samim tim, jasno nam je da se deklaracijom klase ne kreira instanca klase (objekat), već referenca.

Kreiranje objekata se realizuje na drugačiji način. Instanciranje klase, odnosno kreiranje objekta vrši se iz dva koraka:

- alokacija memorije se vrši pomoću operatora *new*.
- inicijalizacija objekta se vrši pomoću konstruktora kojim se alocirana memorija „pretvara” u objekat i objekat inicijalizuje.

Službena reč *new* je operator za stvaranje (instanciranje) novih objekata.

Na slici 14 je prikazan instanca (objekat) soljica, koja je izrađena (napravljena) od klase Soljica.



**Slika 14.** Primer objekta *soljica* (objekat napravljen od klase *Soljica*)

Na sledećem primeru je prikazan način kreiranja objekta soljica pomoću klase Soljica [1].

```
public void Main()
{
    Soljica soljica = new Soljica();
    soljica.mojaVarijabla = 3;
}
```

## 2.2. Modifikatori pristupa

Pre nego što uvedemo pojam enkapsulacija i njeno značenje, moramo prvo da uvedemo pojam modifikatori pristupa. Vrlo je bitno da razlučimo šta su modifikatori pristupa, koja su njihova ograničenja i shvatimo njihovu važnost.

Modifikatori pristupa u jeziku C# služe da odrede koje metode i promenljive članice drugih klasa određena klasa može da vidi i da koristi.

Postoji sledećih 5 modifikatora pristupa:

- *public*,
- *private*,
- *protected*,
- *internal*,
- *protected internal*.

### Public

Modifikatorom public svi elementi klase koji su njim modifikovani su vidljivi svim metodama, svih klasa.

### Private

Modifikatorom private je članovima klase moguće pristupiti samo metodama iz klase **Protected**

Modifikatorom protected svi članovi klase dostupni su samo metodama klase i metodama klasa koje su izvedene iz klase. **Internal**

Članovi klase su dostupni svim metodama svih klasa iz programskog sklopa u kome je klasa.

### Protected Internal

Članovi klase su dostupni svim metodama klase, metodama izvedenim iz klase i klasama programskog sklopa u kome je klasa.

## 2.3. Enkapsulacija

Enkapsulacija objekata je jedan od osnovnih koncepata objektno orijentisanog programiranja i predstavlja dodatnu apstrakciju kojom se „sakrivaju” detalji implementacije objekta.

Postoje dva bitna aspekta enkapsulacije:

- objedinjavanje podataka i funkcija u jedinstven entitet (klasa),
- kontrola mogućnosti pristupa članovima entiteta (modifikatori pristupa).

NAPOMENA: Direktan pristup podacima je potpuno nepotreban i nepoželjan.

Objedinjavanje podataka i funkcija u jedinstven entitet se ostvaruje pomoću klase, a određuje granicom entiteta.

Kontrola mogućnosti pristupa članovima entiteta ostvaruje se navođenjem modifikatora pristupa. Uvođenjem modifikatora pristupa omogućava se razdvajanje klase na javni deo koji čine članovi koji su označeni sa modifikatorom pristupa *public* (pristup nije ograničen) i privatni deo koji čine članovi koji su označeni sa modifikatorom pristupa *private* (mogu mu pristupiti samo članovi klase).

Na osnovu principa objektno orijentisanog programiranja i dobre programerske prakse, preporuka je da podaci objekta treba da se nalaze u privatnom delu. Naravno, ukoliko to ne želite da uradite ne morate, ali tako sami sebi usložnjavate održavanje, a samim tim i mogućnost nastanka *bug-ova*.

Od mesta deklaracije određenog člana zavisi i vrsta modifikatora pristupa koji se može koristiti.

Ukoliko nije naveden, određuje se podrazumevani modifikator:

- za imenovani prostor nije dozvoljeno navođenje modifikatora jer se podrazumeva da su javni (*public*),
- tipovi koji se deklarišu unutar imenovanog prostora mogu biti *public* ili *internal*; podrazumevani modifikator je *internal*,
- članovi klase mogu da imaju bilo koji od navedenih pet modifikatora; podrazumevani modifikator je *private*,
- članovi zapisa mogu biti *public*, *internal* ili *private*; podrazumevani modifikator je *private*,
- članovi interfejsa ne mogu imati modifikatore pristupa; implicitno su *public*,

- članovi nabranja ne mogu imati modifikatore pristupa; implicitno su *public*.

Domen iz koga se može pristupiti određenom tipu nikad ne sme biti uži od domena iz koga se može pristupiti članu koji je deklarisan unutar tog tipa.

```
namespace primer
{
    internal class Racun
    {
        public double stanje; //greška
    }
}
```

Postoje dva razloga:

- omogućavanje kontrole korišćenja – Objekat se može koristiti isključivo preko javnih metoda i
- smanjenje uticaja promena – Ukoliko su detalji implementacije objekta privatni mogu se promeniti, a da te promene ne utiču direktno na korisničke objekte (koje jedino mogu da pristupe javnim metodama) [1].

```
public class Racun {
    public double DajStanje()
    {
        return stanje;
    }
    public void Prikazi(){...}
    private double stanje;
}
public class Klijent
{
    public void Prikazi()
    {
        Console.WriteLine( "Klijent ..." );
        Console.WriteLine( string.Format("Stanje: {0}" racun.DajStanje() );
    }
    private Racun racun; }
```

Ovim promenama se ne utiče na korisnika klase.

```
public class Racun {
    public double DajStanje()
```

## 2.4. Nasleđivanje

Nasleđivanje je koncept objektno orijentisanog programiranja koji omogućava da se na osnovu postojeće klase izvede nova klasa. **Nova izvedena klasa nasleđuje sve članove bazne klase.**

### 2.4.1. Izvedena klasa

Za baznu klasu imena koja se još koriste su roditeljske (parent *class*), predaklase, nadklase, a izvedena klasa se naziva još dete (child *class*), potomak, potklasa. Na slici 15 je prikazano direktno nasleđivanje i smer nasleđivanja, gde se može videti da izvedena klasa nasleđuje od bazne klase, a ne obrnuto. Nasleđivanje funkcioniše isto kao i u stvarnom životu.



Slika 15. Dijagram direktnog nasleđivanja

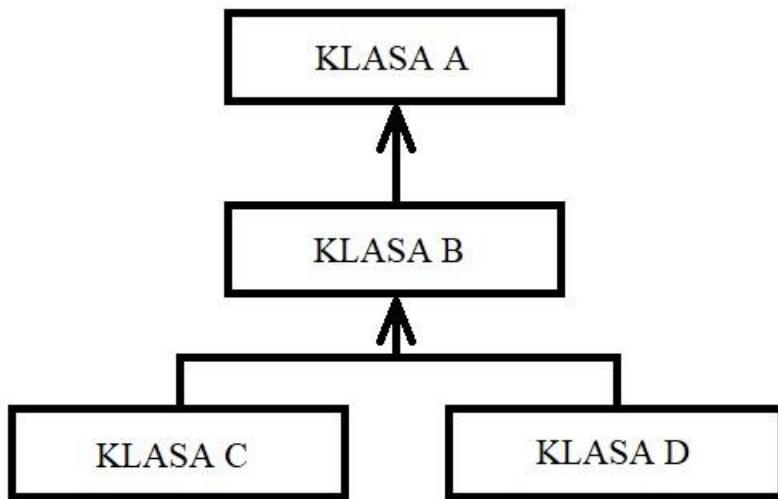
Bitne činjenice koje se moraju shvatiti su da se nasleđivanjem ostvaruje veza (odnos) između objekata i to:

- *is* veza,
- bazna klasa,
- izvedena klasa koja nasleđuje baznu klasu.

Napomena:

Treba razlikovati nasleđivanje klase i nasleđivanje interfejsa (klasa koja nasleđuje interfejs mora da implementira sve navedene funkcije).

Nasleđivanje može biti direktno i indirektno. Na slici 16 prikazan je dijagram na kom se lako može objasniti direktno i indirektno nasleđivanje. Klasa C direktno nasleđuje klasu B, a indirektno klasu A. Sve promene nad baznom klasom (klasa A) se automatski odražavaju nad izvedenim klasama (klasa B, klasa C i klasa D). Obrnuto ne važi. Jednu baznu klasu može da nasledi više izvedenih klasa. Broj izvedenih klasa za jednu baznu klasu nije ograničen. Iz klase B su izvedene klasa C i klasa D.



*Slika 16. Dijagram direktnog i indirektnog nasleđivanja*

U objektno orijentisanom programiranju postoji jednostruko nasleđivanje i višestruko nasleđivanje. Za jednostruko nasleđivanje izvedena klasa direktno nasleđuje jednu baznu klasu, dok za višestruko nasleđivanje izvedena klasa direktno nasleđuje dve ili više baznih klasa. U programskom jeziku C# nije dozvoljeno višestruko nasleđivanje.

Svaka izvedena klasa može dalje biti bazna klasa. Grupa klasa koje su povezane nasleđivanjem formiraju strukturu koja se naziva hijerarhija klasa pri čemu su klase na višim nivoima opštije, dok su one na nižim nivoima u hijerarhiji specifičnije.

Što se tiče dubine hijerarhije preporuka je da broj nivoa ne bude veći od sedam. Ukoliko nije eksplicitno navedena bazna klasa, podrazumeva se *System.Object*.

Sintaksa za nasleđivanje je sledeća:

```

class IzvedenaKlase: BaznaKlase
{
    ...
}

```

Izvedena klasa nasleđuje sve članove bazne klase osim konstruktora i destruktora. Javni članovi bazne klase su implicitno javni članovi izvedene klase. Takođe, izvedena klasa može imati nove članove.

Primer:

```

public class Racun
{
    public void Uplata(decimal iznos){...}
    public void Isplata(decimal iznos){...}
}

```

```

private double brojRacuna;           private
double stanje;
}
public class TekuciRacun: Racun
{
    private double provizija; //novi član
}
...
TekuciRacun tekuci = new TekuciRacun();
tekuci.Uplata(100);

```

**Nasleđivanje ne podrazumeva da će izvedena klasa imati pristup svim članovima bazne klase. Privatni članovi bazne klase, iako su nasleđeni, oni su dostupni isključivo članovima bazne klase.**

Primer:

```

public class Racun
{
    public void Uplata(decimal iznos){...}
    public void Isplata(decimal iznos){...}
    private double brojRacuna;           private
    double stanje;
}
public class TekuciRacun: Racun
{
    public void Prikazi();
    {
        Console.WriteLine("Stanje:", stanje); //greška
    }
}

```

**Članovi bazne klase sa modifikatorom *protected* su jedino dostupni unutar bazne klase i direktno i indirektno izvedenim klasama.**

Na osnovu principa objektno orijentisanog programiranja i dobre programerske prakse, treba težiti da sva polja klase imaju modifikator pristupa *private*, a za svako od njih navesti svojstvo sa modifikatorom pristupa *protected*.

Primer:

```

public class Racun
{
    ...
protected double stanje;
}
public class TekuciRacun: Racun

```

```

{
    public void Prikazi()
    {
        Console.WriteLine("Stanje:", stanje); // OK
    }
}

```

Metode izvedene klase ne mogu da pristupe članovima bazne klase preko reference.

Primer:

```

public class Racun
{
    ...
protected double stanje;
}
public class TekuciRacun: Racun
{
    public void Prikazi(Racun racun)

    {
        Console.WriteLine("Stanje:", racun.stanje); // greška
    }
}

```

Dostupnost izvedene klase je uslovljena dostupnošću bazne klase. Ukoliko je bazna klasa privatna, izvedena klasa ne može biti javna.

Za poziv konstruktora bazne klase iz konstruktora izvedene klase se koristi ključna reč *base*. Prvo se izvršava konstruktor bazne klase. Navođenje inicijalizatora konstruktora nije obavezno (ukoliko se ne navede, poziva se konstruktor [bez parametara] bazne klase). Konstruktor bazne klase, implicitno poziva konstruktor klase System.Object [1].

Primer:

```

public class BaznaKlasa
{
    public BaznaKlasa() {...}
}
public class IzvedenaKlasa: BaznaKlasa
{
    public IzvedenaKlasa(): base() { ... }
}

```

Službena reč koja označava da se klasa ne može naslediti je *sealed*.

Primer:

```
seald class BaznaKlasa
{
    public BaznaKlasa() {...}
}
```

## 2.5. Polimorfizam

**Sposobnost promenljive da referencira objekte različitih tipova i da automatski poziva odgovarajuću metodu objekta koji se referencira, naziva se polimorfizam.**

Polimorfizam se zasniva na sledećem konceptu: metoda koja je deklarisana u baznoj klasi može da se implementira na više različitih načina u različitim izvedenim klasama [1].

Primer:

```
BaznaKlasa promenljiva = new IzvedenaKlasa();
```

### 2.5.1. Virtual

Promenljiva tipa bazna klasa može da referencira instance direktno ili indirektno izvedenih klasa.

Polimorfizam se u jeziku C# najčešće postiže upotrebom virtuelnih metoda. Za deklaraciju virtuelne metode se koristi ključna reč *virtual*.

Primer:

```
public class Racun
{
    public virtual void Prikazi() {...}
}
```

Prilikom definisanja, virtuelne metode se moraju implementirati. Ukoliko to ne učinite, napravićete grešku.

Primer:

```
public class Racun
{
    public virtual void Prikazi(); //greška
```

}

**Privatne i statičke metode se ne mogu definisati kao virtuelne.** Ukoliko ne vodite računa o tome da virtuelne metode nisu modifikovane kao *public*, napravićete grešku.

Primer:

```
public class Racun
{
    private virtual void Prikazi(); //greška
}
```

Metode nisu implicitno virtuelne iz sledećih razloga:

- performanse – potrebno je vreme da bi se odredilo koju od reimplementiranih metoda treba pozvati, što najčešće ne utiče u velikoj meri na performanse. Veći problem je nemogućnost optimizacije koda prilikom kompajliranja. Za nevirtuelne metode ova informacija je dostupna u vreme kompajliranja, što znači da se prilikom kompajliranja;
- mogu sprovesti određene optimizacije (npr. inline);
- dizajn.

Metode klase koje su namenjene za internu upotrebu i koje se odnose isključivo na dizajn date klase ne treba da se reimplementiraju u izvedenim klasama, pa samim tim ne treba ni da budu virtuelne [1].

### 2.5.2. Overriding

Overriing je uslovno jedan od osnovnih principa objektnoorjentisanog programiranja. **Proces implementacije virtuelne metode u izvedenoj klasi se naziva overriding.** Najbanalnije objašnjenje overridinga je nadjačavanje osnovne metode u baznoj klasi implementiranom metodom u izvedenoj klasi. Za definisanje overriinga koristi se ključna reč *override*.

Primer:

```
public class Racun
{
    public virtual void Prikazi() {...}
```

```

}
public class TekuciRacun: Racun
{
    public override void Prikazi() {...}
}

```

Virtuelna i reimplementirana metoda moraju biti identične i moraju imati:

- isti naziv,
- isti modifikator pristupa,
- isti tip rezultata i
- iste parametre.

Koja metoda će biti pozvana, metoda bazne ili izvedene klase određuje se na osnovu tipa instance, koju promenljiva referencira, a ne na osnovu tipa same promenljive.

Primer:

```

public class Racun
{
    public virtual void Prikazi()
    {
        Console.WriteLine("Racun: {0}", brojRacuna);
    }
}
public class TekuciRacun: Racun
{
    public override void Prikazi()
    {
        Console.WriteLine("Tekuci racun: {0}", brojRacuna);
    }
}
Racun racun = new Racun();
racun.Prikazi(); racun =
new TekuciRacun();
racun.Prikazi();

```

Reimplementirana metoda se, takođe, može reimplementirati. Reimplementirana metoda je implicitno virtuelna i ne može se eksplisitno tako deklarisati [1].

## 2.6. Objektni jezik C#

U prethodna dva poglavlja obradili smo već deo C# programskog jezika. U okviru ovog poglavlja obradićemo nizove, liste, strukture, metode i static.

## **2.6.1. Definisanje nizovnih promenljivih i korišćenje sistemskih metoda za rad sa nizovima**

Nizovi su indeksirane kolekcije objekata istog tipa. U programskom jeziku C# postoji bitna razlika od ostalih programskega jezika iz C porodice – nizovi u programskom jeziku C# predstavljaju objekte.

Upravo zbog ove činjenice nizovi imaju svoje metode i svojstva (`BinarySearch()`, `Clear()`, `Copy()`, `CreateInstance()`, `IndexOf()`, `LastIndexOf()`, `Reverse()`, `Sort()`, `IsFixedSize()`, `IsReadOnly()`, `IsSynchronized()`, `Length`, `Rank`, `SyncRoot`, `GetEnumerator()`, `GetLength()`, `GetLowerBound()`, `GetUpperBound()`, `Initialize()` i `SetValue()`). Vrste nizova: jednodimenzionalni, dvodimenzionalni, višedimenzionalni, ugnježdeni, pravougaoni.

Kada se deklarše neki niz u C#-u, automatski se kreira objekat u ugrađenoj klasi `System.Array` (korišćenje ove klase možete videti u naredbi `using` na početku svake aplikacije – `using System.Array`) [1].

### **2.6.1.1. Deklaracija niza**

Deklaracija nizova se realizuje prema sledećoj sintaksi:

```
tip[] ime_niza;
```

### **2.6.1.2. Konstrukcija niza**

Po izvršenoj deklaraciji niza, deklarisani niz i dalje ne postoji. Da bi se napravio taj niz, potrebno ga je konstruisati. Konstrukcija niza se realizuje na sledeći način:

```
int[] mojNiz = new int[5];
```

Bitna stvar je napraviti razliku između samog niza i elemenata niza. Konstruisanjem niza sa `new int[5]` mi smo u stvari deklarisali i konstruisali niz od pet celobrojnih varijabli, a elementi tog niza (tih 5 varijabli) imaju podrazumevane vrednosti. Odnosno, konstruisali smo niz vrednosnog

tipa sa 5 elemenata default vrednosti – 0. Takođe, bitna razlika je i to da se elementi niza referentnog tipa ne inicijalizuju svojim podrazumevanim vrednostima, nego je podrazumevana vrednost *null*. Ovo je veoma važna činjenica jer će se prilikom pokušaja upotrebe elementa izazvati izuzetak [1].

#### 2.6.1.3. Inicijalizacija niza

Nakon izvršene deklaracije i konstrukcije niza, elementi tog novonastalog niza imaju svoje difoltne vrednosti. Inicijalna default vrednost za brojeve, bez obzira da li su celi ili realni, iznosi 0. Inicijalna realna vrednost za string-ove je prazan string.

Inicijalna vrednost za objekte je null, što znači da objekat ne postoji. U mnogim situacijama je potrebno da se definišu konkretne vrednosti niza na početku rada programa.

Definisanje vrednosti elemenata niza se zove inicijalizacija, i to se može uraditi kao što je prikazano na primeru ispod [1].

```
int[] niz = new int[4];
niz[0] = 2; niz[1] = 4;
niz[2] = 6; niz[3] = 8;
```

#### 2.6.2. Dvodimenzionalni i višedimenzionalni nizovi

Da bismo razumeli pojam dvodimenzionalnih, a pogotovo višedimenzionalnih nizova, najlakše je da nizove zamislimo kao redove mesta na koje se smeštaju vrednosti. Ako zamislimo nekoliko takvih redova, zamislili smo jedan dvodimenzionalni niz. Analogno tome, nizovima se može dodati i treća dimenzija i četvrta i tako dalje, ali je takvo stanje malo teže prikazati.

C# podržava dva tipa višedimenzionalnih nizova, i to: pravougaone i nazubljene. Pravougaoni niz je niz sa dve ili više dimenzija. U klasičnom dvodimenzionalnom nizu prva dimenzija je broj redova, a druga je broj kolona.

Deklaracija i konstrukcija niza bi bila realizovana na sledeći način:

```
int[,] pravougliNiz = new int[2, 3];
```

Popunjavanje ovog niza pomoću *for* petlje i prikaz bi se realizovali na sledeći način:

```

static void Main(string[]
args)
{
    int
redovi = 3;           int
kolone = 4;

    //deklaracija niza sa celobrojnim elementima
int[,] Niz = new int[redovi, kolone];

    //popunjavanje niza elementima koji predstavljaju zbir rednog broja i
kolone
    for (int i = 0; i < redovi; i++) {
for (int j = 0; j < kolone; j++) {
    Niz[i,j] = i + j;
}
}

    //prikaz sadrzaja niza
(int i = 0; i < redovi; i++) {           for
(int j = 0; j < kolone; j++) {           for
    Console.WriteLine("Niz[" + i + "," + j +"]=" + i+j+";\n");
}
}
System.Console.Read();
}

```

Zadatke sa nizovnim vrednostima ćemo raditi u petom poglavlju ove skripte, kada budemo naučili izuzetke [1].

### 2.6.3. Kolekcije

Kolekcije su standardne strukture podataka koje dopunjuju nizove. Nizovi su jedina kolekcija koja je ugrađena u C# jezik. Sve ostale kolekcije su ugrađene u prostor imena *System.Collections*.

*Prostor imena System.Collections*

Prostor imena *System.Collections* sadrži klase i interfejsse koje definišu različite kolekcije objekata. Od klasa treba pomenuti *ArrayList*, *Queue*, *Stack*, *Hashtable*. Od interfejsa tu su *ICollection*, *IEnumerable*, *IEnumerator*, *IDictionaryEnumerator*, *Ilist* itd. [1].

#### 2.6.3.1. Klasa ArrayList

Elementima liste pristupa se preko indeksa kao i kod niza. Za razliku od niza nije neophodno unapred poznavati broj elemenata niza. Najčešće korišćene metode ove klase su: metoda

*Add(object)* dodaje objekat na kraj liste, metoda *Clear()* briše sve elemente iz liste, metoda *Insert(pozicija, vrednost)* ubacuje objekat vrednosti na poziciju i metoda *RemoveAt(index)* briše elemenat sa indeksom index iz liste. *GetEnumerator()* metoda vraća iterator koji se koristi za iteraciju (prolaz) kroz elemente liste. Metoda *Sort* sortira elemente liste. Metoda *Reverse()* prikazuje elemente liste u inverznom redosledu. Metoda *ToArray()* kopira elemente liste u jednodimenzionalan niz [1]. **Upotreba ArrayList kolekcije**

U nastavku je prikazan način kreiranja *ArrayList* kolekcije, način dodavanja članova u listu kao i način brisanja članova sa kraja liste.

```
public class ArrayListPrikaz
{
    ArrayList al = new ArrayList();

    public void metod()
    {
        al.Add(7);
        al.Add("Bosko");
        al.Add(12.123);
        al.RemoveAt(0);

        I Enumerator mojEnumerator = al.GetEnumerator();

        while (mojEnumerator.MoveNext())
        {
            Console.WriteLine(mojEnumerator.Current.ToString());
        }
    }
}
```

Iteraciju kroz listu realizuje metoda *GetEnumerator* koja vraća enumerator liste. While petlja služi za štampanje elemenata liste. Enumerator se na početku pozicionira ispred prvog elementa kolekcije. Prvim pozivom metode *MoveNext* on se pozicionira na prvi element liste, sledećim pozivom na drugi itd. Kada se stigne do kraja liste metoda *MoveNext* vraća false. Property *Current* daje tekući element liste [1].

Da bi korišćenje kolekcija bilo moguće, učitavamo prostor imena *Collections*:

```
using System.Collections;
```

### 2.6.3.2. Red (Queue)

Redovi predstavljaju realizaciju FIFO (first-in, first-out) strukture podataka. Metoda *Enqueue* služi za dodavanje elemenata u red. Ulazni parametar ove metode je tipa *object*, tj. može biti bilo koji tip podataka. Skidanje elemenata iz reda vrši se korišćenjem metode *Dequeue* koja nema ulazne

parametre. Kada se pozove ova metoda, iz reda se izbacuje najstariji član, tj. član koji je prvi ubačen u red. Za jednostavan prolazak kroz red koristi se metoda *GetEnumerator* koja vraća enumerator reda.

U nastavku je ilustrovana upotreba reda. Najstariji član u redu je karakter '**A**', pa se pozivom metode *Dequeue* on izbacuje iz reda. Rezultat izvršavanja sekvene koda prikazan je ispod koda [1].

```
public class QueuePrikaz
{
    public void metod()
    {
        Queue mojRed = new Queue();
        mojRed.Enqueue('E');
        mojRed.Enqueue(12);
        mojRed.Enqueue('F');
        mojRed.Enqueue(17);
        mojRed.Dequeue();
        Ienumerator mojEnumerator = mojRed.GetEnumerator();
        while(mojEnumerator.MoveNext())
        {
            Console.WriteLine(mojEnumerator.Current.ToString());
        }
    }
}
```

#### 2.6.3.3. Stek (Stack)

Stack je LIFO (last-in, first-out) struktura. Element koji se poslednji stavi na stek, prvi se skida sa steka. Metoda *Push* stavlja element na vrh steka. Metoda *Pop* skida poslednje stavljeni element sa steka. Metoda *GetEnumerator*, takođe, daje enumerator steka koji služi za jednostavan prolaz kroz elemente steka.

Ovde je prikazan primer upotrebe stack strukture, dodavanje elemenata na stack, skidanje elemenata sa stack-a, prolazak kroz elemente steka [1].

```
public class StackPrikaz
{
    public void metod()
    {
        Stack mojStack = new Stack();
        mojStack.Push("Hello");           mojStack.Push(1);
        mojStack.Push("World");          mojStack.Push("!");
        mojStack.Pop();                 mojStack.Pop();
        Ienumerator mojEnumerator = mojStack.GetEnumerator();
        while (mojEnumerator.MoveNext())
        {
            Console.WriteLine(mojEnumerator.Current);
        }
    }
}
```

```
    }
}
```

#### 2.6.3.4. Hash tabela (Hashtable)

Hash tabela je struktura podataka dizajnirana za brzo pretraživanje. To se postiže dodeljivanjem ključa svakom objektu koji se čuva u tabeli. Stringu „Idi na sever“ dodeljuje se ključ 'N'. Vrednosti koja odgovara ključu 'N' se pristupa korišćenjem izraza *tabela['N']*, gde je tabela instanca klase *Hashtable*. Metoda *GetEnumerator()*, takođe, vraća enumerator koji je tipa *IDictionaryEnumerator*. Pomoću ovog enumeratorsa se može prikazati ključ i njemu pridružena vrednost za sve elemente tabele [1].

```
public class HashTablePrikaz
{
    public void metod()
    {
        Hashtable tabela = new Hashtable();
tabela.Add('N', "Idi na sever");
tabela.Add('S', "Idi na jug");          tabela.Add('W',
"Idi na zapad");          tabela.Add('E', "Idi na
istok");          tabela.Add('Q', "Dovidjenja");
Console.WriteLine(tabela['N']);
        IDictionaryEnumerator mojEnumerator = tabela.GetEnumerator();
while (mojEnumerator.MoveNext())
        {
            Console.WriteLine(mojEnumerator.Key.ToString() + "---->" +
mojEnumerator.Value.ToString());
        }
    }
}
```

#### 2.6.4. Definisanje nabrojivog i strukturnog tipa

##### 2.6.4.1. Strukture podataka

Struktura (engl. *struct*) je jednostavan korisnički definisan tip, lakša alternativa klasi. Strukture su slične klasama po tome što mogu imati konstruktore, svojstva, metode, polja, operatore, ugnježdene tipove i indeksere. Međutim, postoje i značajne razlike između klasa i struktura. Na primer, strukture ne podržavaju nasleđivanje, desturktore, a **najvažnija razlika je u tome što su klase referentni tipovi, dok su strukture vrednosni tipovi**. Samim tim, primena struktura je odlična za predstavljanje objekata kojima nije potrebna semantika referenci.

#### 2.6.4.1.1. Definisanje strukture

Sintaksa za deklarisanje strukture vrlo je slična sintaksi za kreiranje klase:

```
[atributi] [modifikator pristupa] struct identifikator [:lista interfejsa] { clanovi }
```

#### 2.6.4.1.2. Kreiranje strukture

U primeru ispod kreirana je struktura koja ilustruje njenu upotrebu.

```
using System;
using System.Collections.Generic; using
System.Text;

namespace CreatingAStruct
{
    public struct Location
    {
        private int xVal;
        private int yVal;

        public Location(int xCoordinate, int yCoordinate)
        {
            xVal =
xCoordinate;           yVal =
yCoordinate;
        }

        public int x
        {
            get { return xVal; }
            set { xVal = value; }
        }

        public int y
        {
            get { return yVal; }
            set { yVal = value; }
        }

        {
            public void myFunc( Location loc )
            {
                loc.x = 50;
                loc.y = 100;
                Console.WriteLine( "In MyFunc loc: {0}", loc );
            }
        }

        static void Main()
        {
            Location loc1 = new Location( 200, 300 );
            Console.WriteLine( "Loc1 location: {0}", loc1 );
Tester t = new Tester();
            t.myFunc( loc1 );
            Console.WriteLine("Loc1 location: {0}", loc1 );
        }
}
```

```
 } }
```

#### 2.6.4.2. Enumeratori

Nabranja (engl. *enumerations*) su moćna alternativa konstantama. Predstavljaju vrednosni tip podataka koji predstavlja skup imenovanih konstanti.

Deklaracija i upotreba *enum-a*:

```
enum  
Temperature  
{  
    ApsolutnaNula = -273,  
    TackaSmrzavanja = 0,  
    TackaKlucanja = 100,  
};
```

Ovom enumeracijom postižemo bolju logičku vezu u kodu, nego da smo, recimo, kompletну funkcionalnost uradili bez enumeratora. Ista stvar bi mogla da se realizuje na sledeći način:

```
const int ApsolutnaNula = -273; const  
int tackaSmryavanja = 0;  
const int tackaKlucanja = 100;
```

ali bismo ovakvom upotrebom u kasnijem delu povećali složenost održavanja, a smanjili logičku povezanost.

#### 2.6.4. Ključna reč *static*

Članovi klase (promenljive, metode, događaji, indeksi, itd.) mogu biti članovi instance ili statički članovi. Članovi instance povezani su sa instancama tipa, dok se statički članovi smatraju delom klase. Statičkim članovima pristupa se preko imena klase u kojoj su deklarisani.

U C# programskom jeziku nije dozvoljeno pristupiti statičkoj metodi ili promenljivoj članici preko instance. Ako pokušate to da uradite, prevodilac će upozoriti na grešku.

Statičke metode se ponašaju skoro isto kao globalne metode, jer ih možemo pozvati i mimo instance objekta. Međutim, prednost statičke metode je u tome što joj je oblast važenja određena klasom. Statičke metode imaju pristup privatnim članovima klase, jer im je oblast važenja klasa, a ne globalan imenski prostor.

Pored statičkih metoda i statičkih promenljivih, postoje još i statički konstruktori, statičke klase i statička polja [5].

Primer:

```
public class Racun
{
    public static int Br;
    public static void Prikazi()
    {
        Console.WriteLine("Broj: {0}", Br);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Broj: {0}", Racun.Br);
        Racun.Prikazi();
        Console.ReadKey();
    }
}
```

### 2.6.5. Metode

Metodama u programskom jeziku C# postižemo funkcionalnost nekog dela programa, odnosno njom izvršavamo neku akciju.

Osnovna sintaksa bilo kakve metode je:

**modifikator\_pristupa povratna\_vrednost ime\_metode (argumenti\_metoda) telo\_metode**

#### 2.6.6.1. Argumenti metoda i povratne vrednosti

U programskom jeziku postoje različite varijacije metoda koje se mogu pojaviti u upotrebi, pa tako imamo nekoliko različitih varijanti:

Najjednostavnija metoda je metoda koja ne vraća nikakvu vrednost, a ne prosleđuju joj se nikakvi parametri.

Primer je:

```
private void metodaBezArgumenataNeVracaVrednost()
{
    UradiNesto();
```

```
}
```

Važno je zapaziti da za ovakve metode mora biti definisana povratna vrednost na *void*.

Nešto složenija je metoda koja vraća nekakvu povratnu vrednost:

Primer je:

```
private String metodaBezArgumenataVracaStringovnuVrednost()
{
    String nasaVrednost = "testVrednost";
return nasaVrednost;
}
```

Sa druge strane, moguće je imati metodu koja ne vraća nikakvu vrednost, ali kao argumente prima određene vrednosti:

```
private void metodaSaArgNeVracaVrednost (String nasaVrednost, String drugaVrednost)
{
    nasaVrednost += drugaVrednost;

}
```

Postoji mogućnost da nam je potrebno da metodi prosledimo argumente i da nam ta metoda vrati odgovarajuću vrednost.

U tom slučaju ćemo imati sledeće:

```
private String metodaSaArgVracaVrednost (String nasaVrednost, String drugaVrednost)
{
    String sabraniStringovi;
    sabraniStringovi = nasaVrednost + drugaVrednost;
return sabraniStringovi;
}
```

Primer upotrebe metoda možemo videti i prilikom realizacije osnovnih matematičkih operacija, gde pozivanjem metode *matematickeOperacije* vršimo sabiranje brojeva 2 i 4 tako što svakoj metodi koja vrši određenu operaciju prosleđujemo kao argumente prvi i drugi broj, a kao rezultat rada metode prihvatamo vrednost [1].

```
public class MatematickeOperacije
{
    public String matematickeOperacije()
    {
        int prviBroj = 2;
int drugiBroj = 4;
        String zbir = "Zbir je " + sabiranje(prviBroj, drugiBroj) + ".";
        String razlika = "Razlika je " + oduzimanje(prviBroj, drugiBroj) + ".";
    }
}
```

```

        String proizvod = "Proizvod je " + mnozenje(prviBroj, drugiBroj) + ".";
String kolicnik = "Kolicnik je " + deljenje(prviBroj, drugiBroj) + ".";
return zbir + "\n" + razlika + "\n" + proizvod + "\n" + kolicnik; }
    public int sabiranje(int prviBroj, int drugiBroj)
    {
        int zbir = prviBroj + drugiBroj;
return zbir;
    }
    public int oduzimanje(int prviBroj, int drugiBroj)
    {
        int razlika = prviBroj - drugiBroj;
return razlika;
    }
    public int mnozenje(int prviBroj, int drugiBroj)
    {
        int proizvod = prviBroj * drugiBroj;
return proizvod;
    }
    public double deljenje(int prviBroj, int drugiBroj)
    {
        double kolicnik = prviBroj / drugiBroj;
return kolicnik;
    }
}

```

## 3. Klase

U drugom poglavlju uveli smo pojam klase i dali definiciju klase. U okviru ovog poglavlja obradiće se osnovni elementi klase koji nisu pominjani u drugom poglavlju.

### 3.1. Razlika između klase i strukture

Postoje značajne razlike između klasa i struktura. Na primer, strukture ne podržavaju nasleđivanje, destruktore, a **najvažnija razlika je u tome što su klase referentni tipovi, dok su strukture vrednosni tipovi.**

### 3.2. Konstruktor i destruktor

Obrađena je sintaksa konstruktora u okviru metoda, ali nismo zašli „ispod haube” i nismo videli karakteristike konstruktora, njihove osnovne primene i nismo dali nikakve preporuke. Kako je

sintaksa već obrađena, na nju se nećemo vraćati, ali ćemo konstruktore obraditi za potrebe razumevanja njihovog korišcenja [1].

### 3.2.1. Konstruktori

**Konstruktori su specijalne metode koje služe za inicijalizaciju objekata nakon njihovog kreiranja.** Oni obezbeđuju da objekat ima dobro definisano početno stanje pre nego što se upotrebi. Ako ne uspe inicijalizacija neće postojati objekat.

Za imenovanje konstruktora koristi se ime klase (metoda ima isto ime kao i klasa) iza koga slede zagrade. Konstruktori nemaju povratnu vrednost, pa čak ni tipa *void*.

Postoje dve vrste konstruktora i to:

- konstruktori instance (vrše inicijalizaciju objekata)
- statički konstruktori (vrše inicijalizaciju klase) [1].

#### 3.2.1.1. Podrazumevani konstruktor

Kada se kreira objekat .NET kompjuler, ukoliko nije eksplisitno naveden konstruktor, automatski generiše podrazumevani konstruktor.

Primer kroz koji možemo da vidimo navedene činjenice je sledeći:

```
class Datum
{
    private int godina;
private int mesec;           private
int dan;
    // public Datum () { ... } default konstruktor koji se automatski generiše
pošto nije naveden konstruktor
}
class Test
{
    static void Main()
    {
        Datum danas = new Datum(); //kreira se objekat i poziva konstruktor
        ...
    }
}
```

Vrlo je bitno da zapamtite osobine podrazumevanih konstruktora zbog njegove dalje upotrebe u raznim vašim programima (koristićete ih u svakoj klasi koju napravite).

Osobine podrazumevanog konstruktora su:

- podrazumevani konstruktor ne prima parametre,
- podrazumevani konstruktor implicitno inicijalizuje sva nestatička polja na njihove podrazumevane vrednosti i to:
  - numerička polja (int, double,decimal) na nulu,
  - logička polja na *false*,
  - polja referentnog tipa na *null*,
  - polja tipa zapis tako da su svi elementi zapisa inicijlizovani na njihove podrazumevane vrednosti,
- modifikator pristupa je *public* [1].

### 3.2.1.2. Konstruktor sa parametrima

Konstruktor može primiti jedan ili više parametara koji se koriste za inicijalizaciju polja. Ako se u klasi deklariše bar jedan konstruktor, kompjajler neće generisati podrazumevani konstruktor.

```
public class Datum
{
    public int godina, mesec, dan;
public Datum(int g, int m, int d)
{
    godina = g;
mesec = m;           dan
= d;
}
}

class Test
{
    static void Main()
{
    Datum danas = new Datum(2011,09,21);
}
}
```

Sva polja koja nisu inicijalizovana u korisnički definisanom konstruktoru zadržavaju svoju podrazumevanu inicijalizaciju [1].

```

public class Datum
{
    public int godina, mesec, dan;
public Datum(int g, int m, int d)
{
    godina = g;
mesec = m;           dan
= d;
}
public class Primer

{
    static void Main()
{
    Datum danas = new Datum(2011, 09, 21);
    Console.WriteLine(danas.godina); // ispisuje 2010
    Console.WriteLine(danas.mesec); //ispisuje 9
    Console.WriteLine(danas.dan); // ispisuje 21
}
}

```

### 3.1.2.3. Konstruktor kopije

Za jednu klasu može se definisati više konstruktora.

NAPOMENA: Lista parametara svakog od njih mora biti jedinstvena, ili po broju ili po tipu parametara (odnosno moraju imati različite potpise).

Inicijalizatorska lista omogućava da jedan konstruktor poziva drugi koji je deklarisan unutar iste klase čime se omogućava da se konstruktor implementira pozivanjem preklopljenog konstruktora. Sintaksa za ove navode je sledeća:

Prilikom pravljenja konstruktora navode se dve tačke (:) iza kojih sledi ključna reč *this*, a u zagradi su navedeni parametri [1].

NAPOMENA: Inicijalizatorske liste se mogu koristiti samo kod konstruktora.

```

public class Datum
{
    public int godina, mesec, dan;
public Datum() :
this(2010, 3, 30)
{
}
public Datum(int g, int m, int d)
{
    godina = g;
mesec = m;           dan
= d;
}

```

```
    }  
}
```

### 3.2.2. Destruktori

Destruktori su metode koje su u potpunosti suprotne konstruktorima. **Služe da inicirane objekte unište, i to im je osnovna namena.** U velikoj meri destruktore nećete koristiti jer programski jezik C# vrši destrukciju objekata, ali nekada ćete imati potrebu da eksplicitno uništite objekat. U tom slučaju ćete upotrebiti destruktur.

Ukoliko nemate adekvatni konstruktor u klasi prilikom aktiviranja destruktora, destruktur će uništiti instancirani objekat podrazumevanog konstruktora (setite se priče o podrazumevanim konstruktorima) [1].

Osnovne osobine destruktora su sledeće:

- destruktori se ne mogu definisati na strukturama, koriste sa isključivo sa klasama,
- klasa može imati samo jedan destruktur,
- destruktur ne može da se nasleđuje,
- destruktur se ne može pozvati; On se automatski poziva,
- destruktur ne koristi modifikatore ni parametre.

Primer korišćenja destruktora je sledeći:

```
class Auto  
{  
    ~Auto() // destructor klase Auto  
    {  
        // naredbe čišćenja  
    } }
```

Ako idemo malo dalje u razmatranje, možemo postaviti i drugačiji primer:

```
class PrvaKlasa  
{  
    ~PrvaKlasa()  
    {  
        System.Diagnostics.Trace.WriteLine("Desktruktor prve klase je pozvan.");  
    }  
}  
class DrugaKlasa : PrvaKlasa  
{  
    ~Drugaklasa()  
    {  
        System.Diagnostics.Trace.WriteLine("Desktruktor druge klase je pozvan.");  
    } }
```

```

}
class TrecaKlasa : DrugaKlasa
{
    ~TrecaKlasa()
    {
        System.Diagnostics.Trace.WriteLine("Desktruktor treće klase je pozvan.");
    }
}
class TestirajDestruktore
{
    static void Main()
    {
        TrecaKlasa trecaKlasa = new TrecaKlasa();
    } }

```

Startovanjem ovog malog primera dobijamo sledeći izlaz:

Desktruktor treće klase je pozvan.

Desktruktor druge klase je pozvan.

Desktruktor prve klase je pozvan.

Iz njega jasno vidimo da destruktore ne pozivamo, ne prosleđujemo nikakve vrednosti, ali da oni vrše realizaciju bez obzira što mi to nismo uradili [1].

Šta u stvari destruktori rade prilikom realizacije?

Destruktor implicitno poziva metodu Finalize na osnovu bazne klase objekta. Ako pogledamo naš primer klase Auto, upotreba destruktora će se implicitno prevesti u sledeći kod:

```

protected override void Finalize()
{
try
{
    // naredbe čišćenja
}
finally
{
    base.Finalize();
}
}

```

### 3.3. Ključna reč *this*

Ključna reč *this* odnosi se na trenutnu instancu neke klase (ili drugog objekta). ***this* je skriveni pokazivač (pointer) na svaku nestatičku metodu neke klase.**

Postoje tri slučaja upotrebe ključne reči *this*:

1. slučaj upotrebe: U svrhu kvalifikacije instance ako se ona zove isto kao i parametar koji se prosleđuje nekoj metodi, pa tada *this* omogućava da se odredi na koju se vrednost misli.

```
public void NasMetod (int varijabla)
{ this.varijabla = varijabla;
}
```

2. slučaj upotrebe: U svrhu prosleđivanja trenutno aktivnog objekta kao parametra drugoj metodi.

```
public void FirstMethod(OtherClass otherObject)
{ otherObject.SecondMethod(this);
}
```

Ovaj primer uspostavlja dve klase; prva – klasa koja ima metodu *FirstMethod()*, i druga – klasa *OtherClass* koja ima metodu *SecondMethod()*.

Ako unutar prve metode želimo da pozovemo metodu koja pripada drugoj klasi, tada tu drugu metodu moramo pozvati iz te druge klase. Tu klasu ne možemo koristiti preko njenog naziva, nego moramo proslediti parametar – varijablu koja je tipa *OtherClass*.

3. slučaj upotrebe: korišćenjem indeksera Primer korišćenja ključne reči *this*:

Ako *this* koristimo u svrhu određivanja čije svojstvo (property) će se koristiti:

```
label4.Text = this.ImePrezimeStudenta;
```

Gornjom naredbom u tekstu labele *label4* će se ispisati vrednost svojstva *ImePrezimeStudenta* koje pripada aktuelnoj klasi (u kojoj se nalazi ta metoda i naredba).

```
textBox1.Text = this.Ocena;
```

Gornjom naredbom u tekstu *textBox1* će se ispisati vrednost svojstva *Ocena* koje pripada aktuelnoj klasi (u kojoj se nalazi ta metoda i naredba) [1].

### 3.4. Enkapsulacija podataka

U drugom poglavlju smo već pominjali enkapsulaciju i koja je prednost „sakrivanja“ podataka u odnosu na proceduralno programiranje. Da se podsetimo: Enkapsulacija je koncept po kojem je informacija u klasi zaštićena od direktnog pristupa i jedini način da se promeni je kroz definisane metode. Ovakav koncept programiranja naziva se „crna kutija“ – poznat je ulaz i izlaz, ali ne i proces koji se odvija unutar funkcije.

Enkapsulacija se ostvaruje podelom članova klase na javne i privatne. Javni članovi mogu slobodno da se koriste u bilo kom delu programa (van tela same klase), dok su privatni dostupni samo u okviru klase u kojoj su definisani.

Polja su najčešće privatna, dok su metode većinom javne. Poljima ćemo pristupati pomoću get i set metode, tj. pomoću pametnih polja (propertis-a tj. „svojstva“) [1].

### 3.5. Propertiji (svojstva)

Svojstva (ili još i pametna polja) ili preciznije propertiji su članovi strukture, klase, interfejsa koji obezbeđuju fleksibilan mehanizam za čitanje, pisanje ili izračunavanje vrednosti nekog polja.

Propertiji liče na polja ali se ponašaju kao metode. Propertije koristite kao da su članovi javnih promenjivih, ali imajte u vidu da su propertiji posebne metode koje zovemo i *Accessors* – pristupne metode. Savetujemo svakog programera da uvek koriste propertije umesto javno promenjivih u svojim klasama i radi bezbednosnih razloga. Propertije možete koristiti samo za čitanje, samo za pisanje, i za čitanje i pisanje. Kraće pisanje propertija nazivamo auto propertiji. Tokom evolucije C# programskega jezika do danas verzije 6.0; propertiji su napredovali do toga da danas možete inicijalizovati vrednost auto propertijima ili pisati auto propertije samo za čitanje.

Opšti oblik propertija je:

```
public class ImeKlase
{
    private tip_podatka propreti_promenljiva;
    public tip_podatka Ime_propretija
    {
        get {return propreti_promenljiva;}
        set { propreti_promenljiva = value;}
    }
}
```

```
}
```

Naravno, ova sintaksa se odnosi na svojstvo koje se čita i piše. Kad vam treba propertij samo za čitanje izostavite set metodu iz propertija ili ako vam treba properti samo za pisanje, izostavite get metodu iz propertija. Imajte u vidu kada imate propertij samo za čitanje da taj properti možete samo čitati, dok propertij za pisanje možete koristiti samo za pisanje, ali ne možete videti šta piše. Postavlja se pitanje zašto bi onda neko uopšte koristio properti samo za pisanje. To se uglavnom radi kada želite da koristite properti za skrivene informacije poput šifre. Pisanje na kraći način propertija se zove auto propertiji i sintaksa za auto properti je sledeća:

```
public tip_podatka Ime_propretija {get; set;}
```

Piše se bez tačke – zareza (;) iza veliki zagrada za kraj reda. Propertiji mogu biti u domenu *public*, *private* ili *protected*. Metode propertija ne moraju da se posebno deklarišu, ali zato metoda *get* ili *set* može biti definisan drugačije od propertija. Samo jedna metoda *get* ili *set* može biti drugačije definisana, ali morate voditi računa da modifikator ne sme da bude manje restriktivan od pristupa svojstvu. Npr. ako je properti deklarisan kao *private*, ne možete njegovu metodu *set* ili *get* definisati kao *public*, dok obratno možete.

U propertijima možete imati samo jednu *get* i samo jednu *set* metodu. Propertije ne možete da koristite kao ref ili out argument u metodi i properti ne može da sadrži druge metode, polja ili svojstva. Pristupne metode *get* i *set* ne mogu da imaju parametre niti možete da deklarišete properti kao konstantu. Podatak koji se dodeljuje propertiju se automatski prosleđuje pristupnoj metodi set koristeći uvek promenjivu koja se zove *value*. Ona ne može imati drugi naziv [1].

Primer:

```
public class Class1
{
    private string ime;
public string Ime
{
    get {return ime;}
set {ime = value;}
}
public string Prezime {get; set;}
```

### 3.6. Preklapanje metode u klasi

Često se dešava da dolazi do preklapanja imena metoda.

Potpis metode određen je njenim imenom i listom parametara. Dve metode imaju različite potpise ako nemaju identične liste parametara i ime. Liste parametara mogu da se razlikuju po broju i tipovima parametara.

Primer:

```
public int zbirParametara (int br1)
{ return br1;
}
public int zbirParametara (int br1, int br2)
{ return br1 + br2;
}
public int zbirParametara (int br1, int br2, int br3)
{ return br1 + br2 + br3;
}
```

Klase može da ima proizvoljan broj metoda, ali s različitim potpisom [5].

### 3.7. Pojam parcijalne klase

Klase može da sadrži više metoda, polja i konstruktora. Neke klase mogu biti glomazne. Zato, u jeziku C# postoji mogućnost da podelite izvorni kod klase u nekoliko odvojenih fajlova (datoteka) tako da neka glomazna klasa može da se organizuje kao skup nekoliko manjih klasa.

Kada se neka klasa podeli u nekoliko fajlova, onda se definišu tzv. „parcijalne klase“ pomoću ključne reči **partial**.

Primer: Klasa Datum može da se podeli u dva fajla: *Datum1.cs* (koja sadrži konstruktore) i *datum2.cs* (koja sadrži metode i polja):

```
partial class Datum
{
    public Datum(){ } // podrazumevani konstruktor
public Datum(){ ... } // nepodrazumevani konstruktor
}
partial class Datum
{
    public int godina, mesec, dan;
public int Dan ()
```

```
{  
    return dan;  
}  
}
```

## 4. Rukovanje izuzecima

### 4.1. Rukovanje, prijavljivanje i prihvatanje (obrada) izuzetaka

Greške su uvek moguće, mogu nastati u bilo kom trenutku izvršavanja programa. Ovom činjenicom stalno moramo da se rukovodimo. Upravo zbog ove činjenice postoji glavna razlika između dobrog i lošeg softvera, odnosno, dobrog i lošeg programa. Dobar program je onaj koji je u stanju da identificuje i obradi nastalu grešku. Jedan od načina obrade grešaka je i obrada izuzetaka.

U .NET Frameworku postoji ugrađeni mehanizam koji obrađuje izuzetke. Njegove osnovne karakteristike su:

- služi za obradu grešaka,
- pruža dovoljno informacija o nastaloj grešci,
- omogućava da se za svaki tip greške kreira odgovarajući način obrade, • omogućava odvajanje logike programa od koda kojim se obrađuju greške,
- bazira se na predstavljanju izuzetaka pomoću objekata.

Kod tradicionalne provere grešaka morali smo da pišemo kod sličan ovome:

```
int kodGreške = 0;  
FileInfo izvor = new FileInfo("kod.cs");  
if (kodGreške == -1) goto greška; int  
dužina = (int)izvor.Length; if  
(kodGreške == -2) goto greška; char[]  
sadržaj = new char[dužina]; if  
(kodGreške == -3) goto greška; ...  
greška: ...;
```

Vrednost promenljive kodGreške ukazuje na to da li je nastala greška, i ako jeste, kog je tipa. Kod ovakvog pristupa ispoljilo se dosta nedostataka, kao na primer:

- logika i obrada grešaka se prepliću,
- instrukcije za otkrivanje grešaka su veoma slične,

- sve one testiraju istu promenljivu (kodGreške) korišćenjem naredbe *if*,
- dosta ponovljenog koda,
- kodovi grešaka sami po sebi nisu jasni,
- šta označava vrednost -1 ?
- vrednost koda greške, celobrojna vrednost, nema eksplisitno značenje odnosno ne opisuje grešku koju predstavlja,
- gledanje dokumentacije je zamorno i podložno greškama,
- kodovi grešaka se lako mogu predvideti,
- česta je situacija da se provera koda greške zanemari, a samim tim i obrada greške,
- potreban je fleksibilniji mehanizam koji pruža dovoljno informacija o grešci.

Da bi se prevazišao problem nedostatka informacija o nastalim greškama, .NET definiše niz različitih klasa izuzetaka. Kako bismo mogli uopšte da pričamo dalje, potrebno je da definišemo šta je to izuzetak. Izuzetak je objekat koji se kreira ili podiže (*throw*) kada dođe do određene greške i sadrži informacije koje bi trebale omoguće identifikaciju greške. Za potrebe obrade izuzetaka kreirana je kompletna hijerarhija klasa koje se bave isključivo izuzecima.

Osnovne karakteristike ovih klasa su da svaka klasa može da se nađe unutar posebne izvorne datoteke i nezavisna je od drugih klasa i svaka klasa može da sadrži i njoj svojstvene podatke (npr. *FileNotFoundException* klasa mogla bi da sadrži ime datoteke koja nije pronađena).

Ove klase obezbeđuju sledeće prednosti:

- poruke o greškama se više ne predstavljaju brojevima, umesto njih koriste se odgovarajuće klasе izuzetaka (npr. *OutOfMemoryException* klasа umesto -3) i
- generišu se deskriptivne poruke o greškama. Svaka klasа se odnosi na određenu grešku i daje dovoljno jasan opis.

U hijerarhiji postoje dve osnovne klase koje nasledjuju *System.Exception*:

- *SystemException* klasа - klasа iz koje se direktno ili indirektno izvode sve klasе sistemskih izuzetaka i
- *ApplicationException* klasа - klasа iz koje se direktno ili indirektno izvode sve korisnički definisane klasе izuzetaka.

Specifičnost ove hijerarhije klasа izuzetaka se ogleda u tome što većina klasа nema nikakvu dodatnu funkcionalnost u odnosu na svoje osnovne klasе. Kada je reč o obradi izuzetaka najčešći

razlog nasleđivanja je da se ukaže na specifičnosti pojedinih grešaka i stoga nema potrebe da se dodaju nove metode ili pregaze nasleđene (iako nije redak slučaj dodavanja novih svojstava koja pružaju dodatne informacije o greškama).

Princip korišćenja ovih klasa na obradi izuzetaka je sledeći:

Kritični kod se deli na tri dela i to:

- *try* blok sadrži kod koji se odnosi na logiku programa u kome se mogu javiti ozbiljne greške,
- *catch* blok sadrži kod koji obrađuje različite tipove grešaka.
- *finally* blok sadrži kod kojim se oslobađaju resursi ili preduzimaju bilo kakve druge akcije koje bi trebalo da budu izvršene na kraju *try* ili *catch* blokova. Veoma je važno znati da se ovaj blok izvršava u svakom slučaju (bez obzira na to da li je podignut izuzetak).

Tok izvršavanja je sledeći:

- tok izvršavanja prelazi na *try* blok,
- ukoliko ne dođe do bilo kakve greške, izvršavanje se normalno nastavlja sve do kraja ovog bloka kada se izvršavanje prenosi na *finally* blok. Međutim, ukoliko dođe do greške unutar *try* bloka izvršavanje se prenosi *catch* blok,
- u *catch* bloku se vrši obrada greške,
- na kraju *catch* bloka izvršavanje se automatski prenosi na *finally* blok,
- *finally* blok se izvršava.

Načelno, realizacija hvatanja izuzetaka se realizuje prema sledećem:

```
try
{
    //logika programa
}

catch
{
    // obrada greške
}

finally
{
    // oslobođanje resursa
}
```

I pored ovoga, moguće je da nam *finally* blok nije potreban, pa se on može izostaviti. Takođe, moguće je da imamo potrebu da obradim više tipova izuzetaka u jednom problematičnom *try* bloku. U tom slučaju možemo kreirati situaciju sa višestrukim *catch* blokovima.

Najjednostavniji primer je prilikom parsiranja ulaza i deljenja tih brojeva.

```
try
{
    int i = int.Parse(Console.ReadLine());
    int j = int.Parse(Console.ReadLine());           int
    k = i / j;
}
catch (OverflowException izuzetak1)
{
    Console.WriteLine(izuzetak1);
}

catch (DivideByZeroException izuzetak2)
{
    Console.WriteLine(izuzetak2);
}
```

Takođe, možemo imati i potrebu da definišemo izuzetak za koji ne znamo njegovu prirodu. U tom slučaju ćemo koristiti opšti *catch* blok.

Primer korišcenja opšteg *catch* bloka:

```
catch (Exception izuzetak)
{
    //kod koji rukuje sa bilo kojim izuzetkom
}
```

## 4.2. Često korišćene klase izuzetaka

*ArithmetiсException* – osnovna klasa za izuzetke koji nastaju prilikom izvršavanja aritmetičkih operacija kao što su *DivideByZeroException* i *OverflowException*.

*DivideByZeroException* – podiže se prilikom pokušaja deljenja celobrojne vrednosti nulom.

*ArrayTypeMismatchException* – podiže se prilikom pokušaja da se u niz ubaci elemenat čiji tip nije kompatibilan sa tipom elemenata niza.

*IndexOutOfRangeException* – podiže se prilikom pokušaja pristupa nepostojećem elementu niza.

*InvalidCastException* – podiže se kada eksplisitna konverzija osnovnog tipa ili interfejsa u izvedeni tip ne može da se izvrši.

*NullReferenceException* – podiže se prilikom pokušaja korišćenja nepostojećeg objekta (vrednost reference je *null*).

*OutOfMemoryException* – podiže se prilikom neuspešnog pokušaja alociranja memorije (putem *new*).

*OverflowException* – podiže se kada aritmetička operacija izazove overflow (ukoliko je checked opcija uključena).

*StackOverflowException* – podiže se kada nema više mesta na steku usled prevelikog broja pozvanih metoda (npr. rekurzija).

*FileNotFoundException* – podiže se prilikom pokušaja da se pristupi datoteci koja ne postoji.

*ArgumentException* – podiže se kada vrednost nekog od prosleđenih parametara nije ispravna.

*ArgumentNullException* – podiže se kada se kao vrednost parametra prosledi *null* vrednost, a to nije dozvoljeno.

## 5. Izvedene klase

### 5.1. Izvedene klase

Izvedene klase smo već obradili u drugom poglavlju kada smo govorili o nasleđivanju, tako da u ovom poglavlju nema potrebe da ponavljamo. Na slikama 15 i 16 koje se nalaze u drugom poglavlju govorili smo o izvedenim klasama. Izvedena klasa se dobija nasleđivanjem svih članova bazne klase.

### 5.2. Apstraktna klasa

Kada uređujete klase u hijerarhiju nasleđivanja, obično su „više” klase apstraktnije i uopštenije, dok su „niže” klase konkretnije i specifičnije. Često polazna klasa i nema instance, već služi kao izvor informacija koje podklase koriste. Ovakve klase se nazivaju apstraktnim i deklarišu se pomoću ključne reči *abstract*.

Definicija apstraktne klase izgleda ovako:

```
public abstract class ApstrakntaKlasa
{
    ...
}
```

**Apstraktne klase ne mogu da imaju objekte.** Ukoliko pokušate da kreirate instancu apstraktne klase, izazvaćete grešku kompjerala. Međutim, one mogu da sadrže sve što i normalne klase, uključujući klasne i objektne promenljive i metode sa bilo kojim nivoom zaštite [1].

Primer:

```
using System;

namespace ApstraktneKlase
{
    public interface IRacun
    {
        void IsplatiSaRacuna(double iznos);
        void UplatiNaRacun(double iznos);
    }
}
```

```

        string VratiPodatkeORacunu();           double
VratiStanje();
    }

public abstract class AbstractRacun : ApstraktneKlase.IRacun
{
    private double stanje;
private string brojRacuna;
    public AbstractRacun(string brojRacuna) : this(brojRacuna, 0) { }

    public AbstractRacun(string brojRacuna, double pocetnoStanje)
    {
        this.brojRacuna = brojRacuna;
this.stanje = pocetnoStanje;
    }
    public double VratiStanje()
    {
        return stanje;
    }
    public void UplatiNaRacun(double iznos)
    {
        stanje += iznos - ProvizijaNaUplatu(iznos);
    }
    public void IsplatiSaRacuna(double iznos)
    {
        stanje -= iznos + ProvizijaNaIsplatu(iznos);
    }
    protected abstract double
ProvizijaNaUplatu(double iznos);           protected abstract double
ProvizijaNaIsplatu(double iznos);

    public virtual string VratiPodatkeORacunu()
    {
        string podaci = "Racun broj: " + brojRacuna + "\nIznos na racunu: " +
stanje;
        return podaci;
    }
    public class TekuciRacun : AbstractRacun
    {
        public TekuciRacun(string brojRacuna) : base(brojRacuna) { }
        public TekuciRacun(string brojRacuna, double pocetnoStanje)
:
base(brojRacuna, pocetnoStanje) { }

        protected override double ProvizijaNaUplatu(double iznos)
        {
            return 0;
        }

        protected override double ProvizijaNaIsplatu(double iznos)
        {
            double obracunataProvizija = 100 + (iznos * 3 / 100);
return obracunataProvizija;
        }

        public override string VratiPodatkeORacunu()
        {
            return base.VratiPodatkeORacunu() + "\nTip racuna: TEKUCI";
        }
    }
}

```

```

        }

    }

    public class DevizniRacun : AbstractRacun
    {
        public DevizniRacun(string brojRacuna) : base(brojRacuna) { }
        public DevizniRacun(string brojRacuna, double pocetnoStanje) :
        base(brojRacuna, pocetnoStanje) { }
            protected override double ProvizijaNaUplatu(double iznos)
            {
                double obracunataProvizija = 100 + (iznos * 5 / 100);
            return obracunataProvizija;
            }

            protected override double ProvizijaNaIsplatu(double iznos)
            {
                double obracunataProvizija = 100 + (iznos * 5 / 100);
            return obracunataProvizija;
            }

        public override string VratiPodatkeORacunu()
        {
            return base.VratiPodatkeORacunu() + "\nTipRacuna: DEVIZNI";
        }
    }

    public static void Main(string[] args)
    {
        AbstractRacun[] racuni = new AbstractRacun[2];
        racuni[0] = new TekuciRacun("392-411232-417", 0);
        racuni[1] = new DevizniRacun("396-411232-417", 0);           for
        (int i = 0; i < 2; i++)
        {
            Console.WriteLine("Uplata na " + i + ". racun u iznosu od 1000");
        racuni[i].UplatiNaRacun(1000);
            Console.WriteLine("Podaci o racunu posle update: \n" +
            racuni[i].VratiPodatkeORacunu());
            Console.WriteLine();
        }
    }
}

```

### 5.3. Interfejsi i nasleđivanje interfejsa

Interfejs predstavlja „ugovor“ kojim se garantuje da će se klasa, koja je nasledila taj interfejs, ponašati na određeni način. Dakle, klasa garantuje da prodržava metode, svojstva (*properties*), događaje (*events*) i indeksere nekog interfejsa. **Sintaksno interfejs predstavlja klasu koja sadrži samo apstraktne metode.**

Sintaksa za implementaciju interfejsa izgleda ovako:

```
class PrimerKlasa : PrimerInterface
{
    ...
}
```

Kada klasa implementira interfejs, ona mora da implementira sve njene metode.

Moguće je da jedna klasa implementira više interfejsa [1].

NAPOMENA: Višestruko nasleđivanje klase nije dozvoljeno u C#, ali jeste višestruko nasleđivanje interfejsa.

Primer:

```
using System;

interface IBazniInterface
{
    void BazniInterfaceMetod();
}

interface IMojInterface : IBazniInterface
{
    void MetodaZaImplementaciju();
}

class InterfaceImplementer : IMojInterface
{
    static void Main()
    {
        InterfaceImplementer iImp = new InterfaceImplementer();
        iImp.MetodaZaImplementaciju();           iImp.BazniInterfaceMetod();
    }

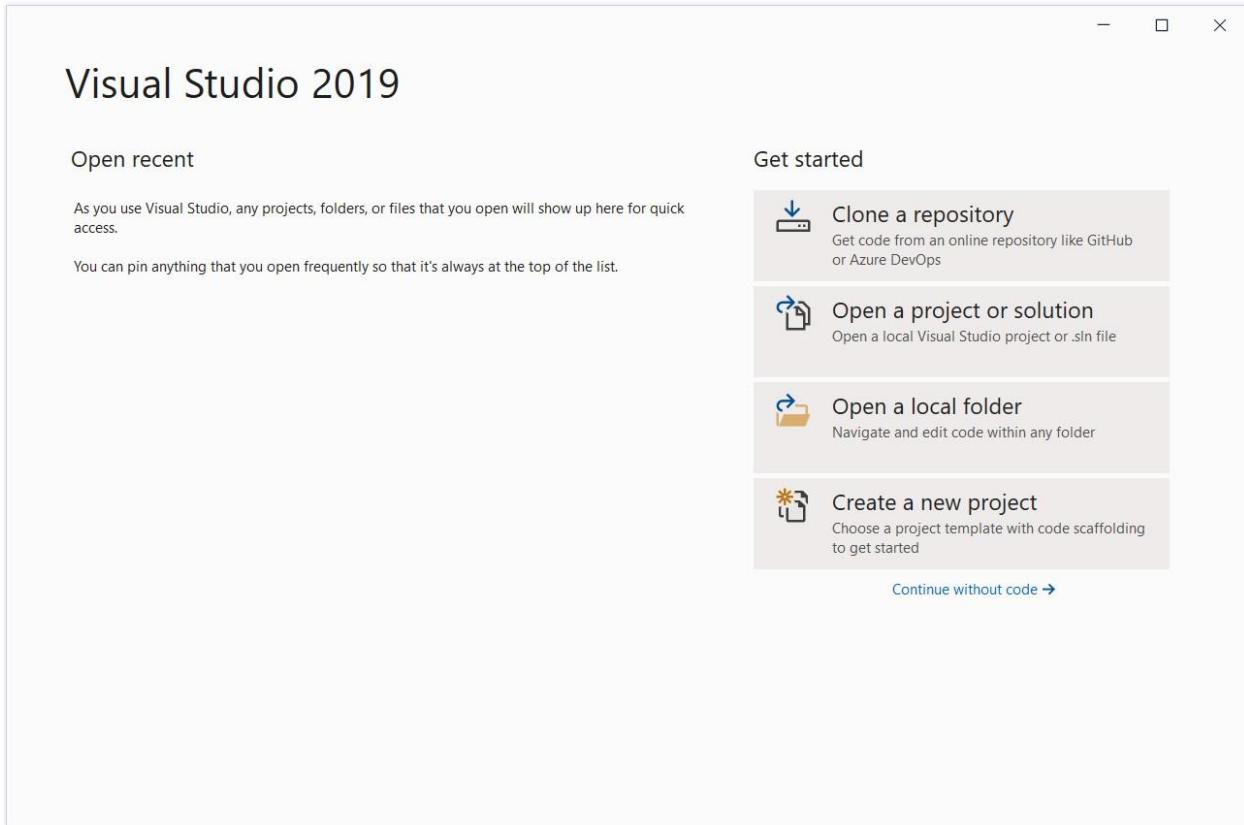
    public void MetodaZaImplementaciju()
    {
        Console.WriteLine("MetodaZaImplementaciju() je pozvana.");
    }

    public void BazniInterfaceMetod()
    {
        Console.WriteLine("BazniInterfaceMetod() je pozvan.");
    }
}
```

## 6. Biblioteka komponenata

### 6.1. Izrada projekta – Kreiranje WinFoms aplikacije u Visual Studio

Startovati MS Visual Studio 2019 [3].

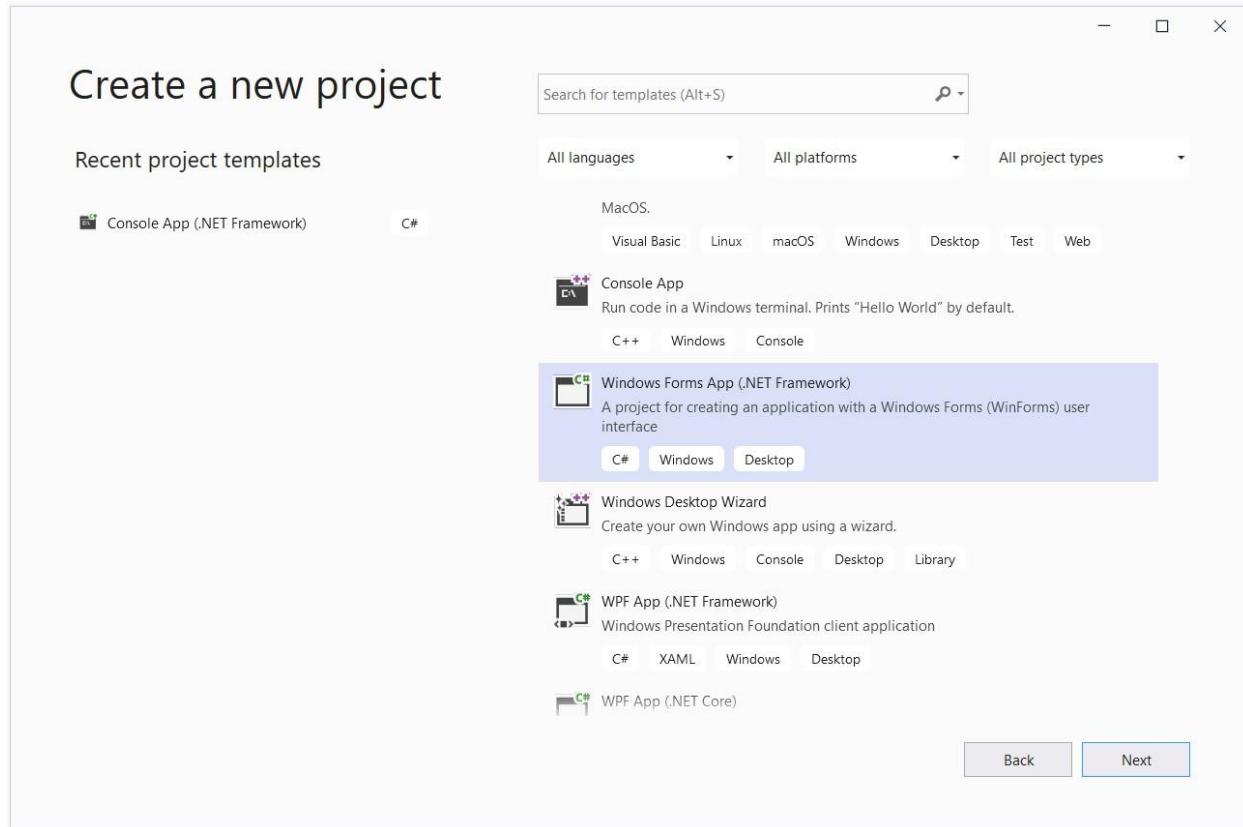


*Slika 17. Izgled osnovnog prozora u MS Visual Studio okruženju*

## 6.2. Forma – Definisanje projekta kao WinForms

Da bi se kreirao projekat, potrebno je izabrati i tip aplikacije koja će se razvijati. Sa leve strane slike 18 se nalaze kategorije, dok se u centralnom delu, u zavisnosti od izabrane kategorije, prikazuju tipovi aplikacija. WinForms aplikacije spadaju u grupu Windows desktop aplikacija, pa u levom delu treba tu kategoriju i selektovati, a u centralnom delu izabrati opciju *Windows Forms Application*.

U donjem delu se definiše naziv projekta (polje *Name*), kao i lokacija (*Browse*). *Solution Name* je podrazumevano identičan nazivu projekta, ali je moguće definisati i neki drugi naziv. To je naziv koji će se videti u *Solution Explorer*-u (slika 19) [3].

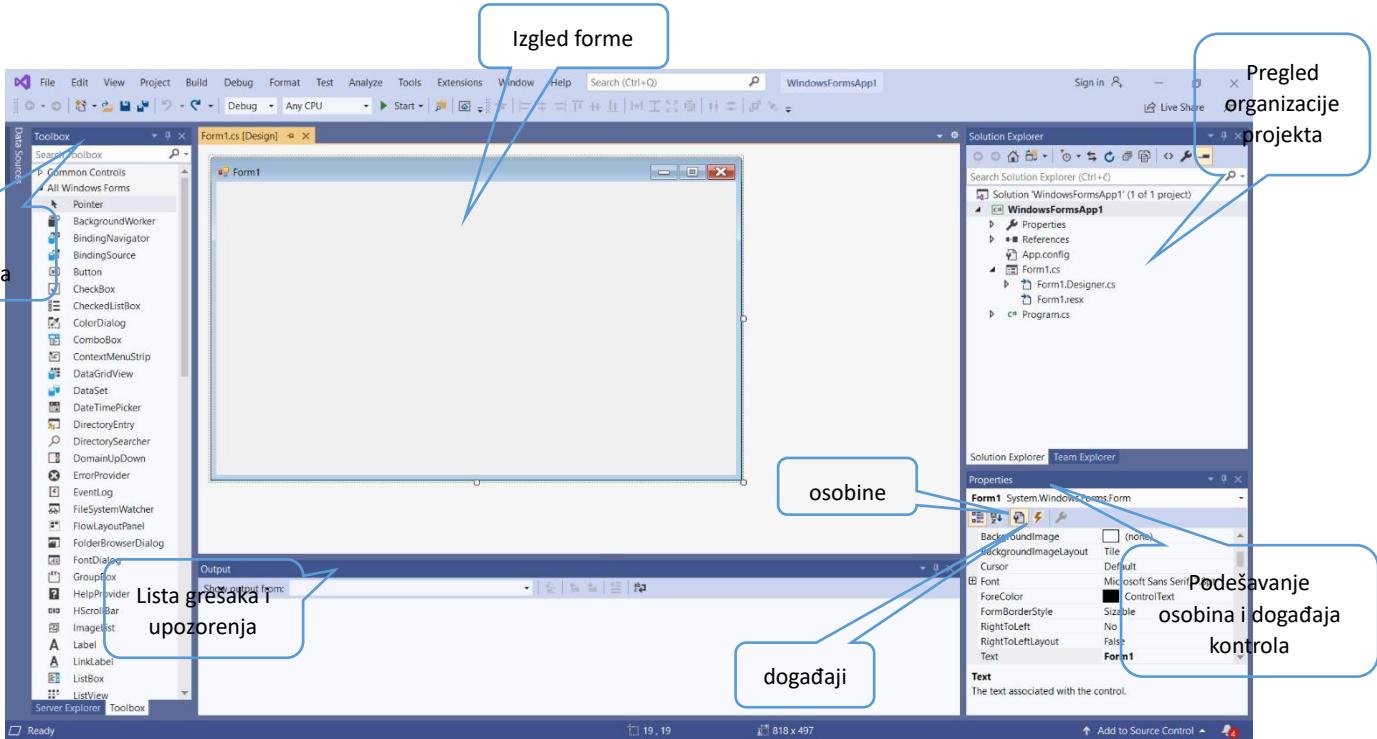


*Slika 18. Izbor tipa projekta*

### 6.3. Izgled radne površine

Na slici 19 je prikazan izgled radne površine. Sa leve strane se nalazi paleta sa kontrolama (*Toolbox*), koji je radi lakšeg pregleda podeljen po kategorijama. Centralni deo se sastoji iz dva dela. Gornji deo služi za pregled i manipulaciju u *Design* modu (fizički izgled formi) ili implementaciju klase, događaja, metoda itd. direktnim kucanjem C# koda (ekstenzija *.cs*). U donjem delu se vidi spisak grešaka i upozorenja (*Error List*) i izlaznih vrednosti (*Output*). Error

List je bitan deo jer sadrži spisak svih sintaksnih grešaka obeleženih crvenim kružićem sa belim znakom putače (x) u sredini. Svaka stavka u toj listi (*Errors*) sadrži tačnu lokaciju gde se nalazi u kodu (redni broj linije koda), kao i opis same greške. Dokle god kod sadrži neku od Error-a neće moći da se startuje (tj. moći će da se startuje sa prikazom prozora u kome obaveštava korisnika da postoje greške i da aplikacija neće validno raditi). U ovom delu se prikazuje i spisak upozorenja (*Warnings*) koja upućuju na moguće greške u *run time* modu (u toku samog izvršavanja programa). Na primer, može doći do deljenja sa nulom, ili da je moguće da neka promenljiva nije inicijalizovana. Aplikacija može da se validno startuje ako ima upozorenja, s tim da su ona "sigurno obezbeđena" u samom C# kodu od strane programera [3].



*Slika 19. Osnovna radna površina*

Desni deo radne površine (slika 19) je podeljen na dva dela. U gornjem delu je prikazan *Solution Explorer* koji sadrži prikaz logičke organizacije projekta (spisak fajlova i foldera prikazanih u vidu stabla). Kao što se na pomenutoj slici vidi, svaka forma ima ekstenziju .cs i sadrži tri fajla:

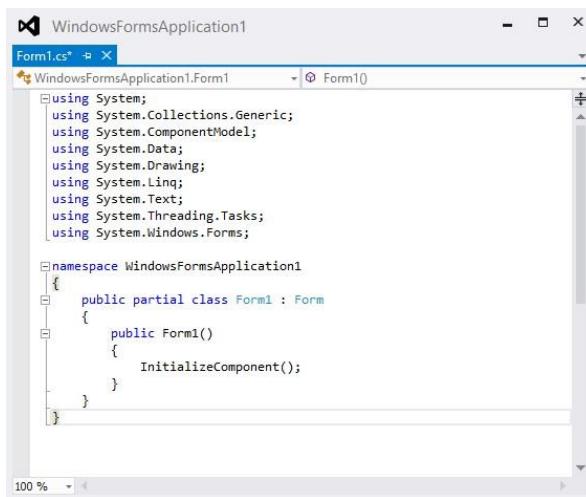
- sa ekstenzijom .Designer.cs čije se ime automatski menja, tj. ubacuje mu se odgovarajući C# kod (npr. kad se prevuče kontrola iz *Toolbox*-a na formu ili kada se podeše svojstva kontrole pomoću *Properties* prozora),
- sa ekstenzijom .resx za dodavanje dodatnih resursa,
- klasa koja sadrži izgled (opis) same klase (polja i metode) [3].

Donji desni deo radne površine (slika 19) sadrži prikaz *Properties* prozora koji omogućava „ručno“ podešavanje osobina i događaja kontrola sa forme. U zavisnosti koja kontrola je na formi selektovana taj prozor se menja, jer različite kontrole imaju različite osobine i događaje. Za sve što se ovde podeši automatski se generiše odgovarajući kod koji se smešta u fajl sa ekstenzijom .Designer.cs za odgovarajuću formu [3].

## 6.4. Otvaranje prozora za pisanje koda

Kao što je ranije napomenuto, neki delovi koda se automatski generišu i to je najčešće deo vezan za fizički izgled forme i kontrola. Ali samu „život“ aplikaciji daju događaji (akcije) koje se izvršavaju kada korisnik inicira neku akciju (klik mišem, izbor *radio button*-a, selekcija stavke u padajućoj listi itd.). Sve ovo programer mora iskodirati direktnim kucanjem C# koda.

Kako bi se otvorio prozor za pisanje koda, potrebno je u Solution Explorer-u desnim klikom miša kliknuti na formu za koju se želi pisati kod (npr. Form1.cs) i izabrati opciju View Code. Pri čemu se otvara prozor sličan ovom koji je prikazan na slici 20 [3].



```
WindowsFormsApplication1
Form1.cs*  X
WindowsFormsApplication1.Form1  Form1()
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

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

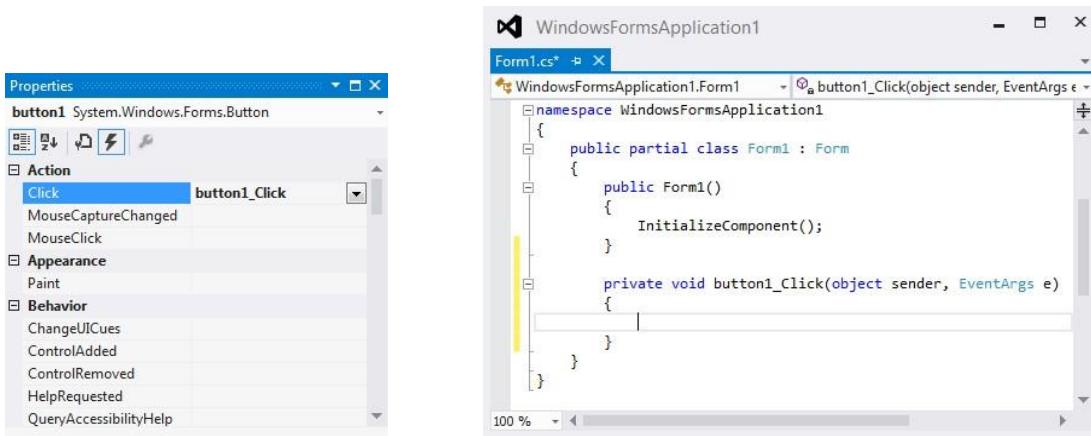
*Slika 20. Prozor za pisanje koda*

Na pomenutoj slici je prikazan *default*-ni kod koji je automatski generisan u fajlu Form1.cs.

## 6.5. Događaji

Kako bi se „oživila“ kontrola neophodno je inicirati željeni događaj za nju. Ovo se postiže sledećim postupkom: Na formi (u Design-u) selektovati kontrolu za koju se želi aktivirati događaj. Potom sa desne strane u Properties prozoru prebaciti se na događaje (klik na znak munje  ) i na kraju izabrati odgovarajući događaj iz spiska dostupnih događaja za tu kontrolu. Slika 21 prikazuje aktivaciju događaja *Click* za Button kontrolu.

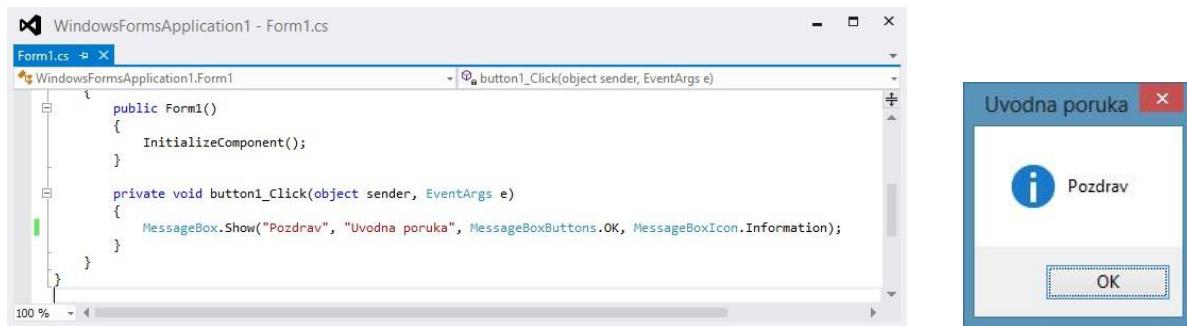
Dupli klik na naziv izabranog događaja otvara se prozor za pisanje koda uz generisano zaglavljeno događaja koji se implementira (slika 22).



*Slika 21. Aktivacija događaja*

*Slika 22. Automatski generisano zagлавље događaja*

Na slici 23. je prikazan deo koda koji omogućava prikaz poruke u Message Box-u (slika 24) kada se klikne na Button (odnosno kada se u run-time inicira događaj Click za kontrolu Button). Telo događaja (tj. metode koja se poziva kada se inicira događaj) programer kuca sam [3].



*Slika 23. Implementacija događaja*

*Slika 24. Izlazni prozor*

Za svaku kontrolu koja je planirana da se obraduje u okviru predmeta Programiranje 3, tabelarno će se prikazati svi događaji te kontrole.

### 6.5.1. Događaji miša

U nameri da ne ponavljamo ono što svi znaju nećemo da objašnjavamo šta je miš i kako radi. Pomenućemo samo osnovne događaje vezane za ovaj uređaj koji služi za unos podataka.

Upotreba miša dovodi do: promene pozicije kursora, pritisak tastera (levo, desno, srednje), otpuštanje istih tastera i na kraju skrolovanje ako miš ima točkić. Takođe, posebno se detektuje i dvosturki klik. Pritisak na levi taster je *Click*, a na desni *Right-Click*.

`MouseDown` je poruka koja se događa kada se pritisne levi taster miša. `MouseEventArgs` je tip ove poruke. Analogno je i za otpuštanje tastera tj. `MouseUp`, odnosno `MouseEventArgs` koji se prosleđuje `MouseEventHandler` na obradu. Pomeraj miša se detektuje porukom `MouseMove` [6].

### **6.5.2. Događaji tastature**

Posebna klasa događaja koje windows kontrole obrađuju su događaji sa tastature. Tastatura predstavlja jedan od dva osnovna uređaja za unos podataka u kontrole. Drugi je miš.

Kad god se pritisne neki taster aktivira se događaj `KeyDown`, a tip poruke je `KeyEventArgs`.

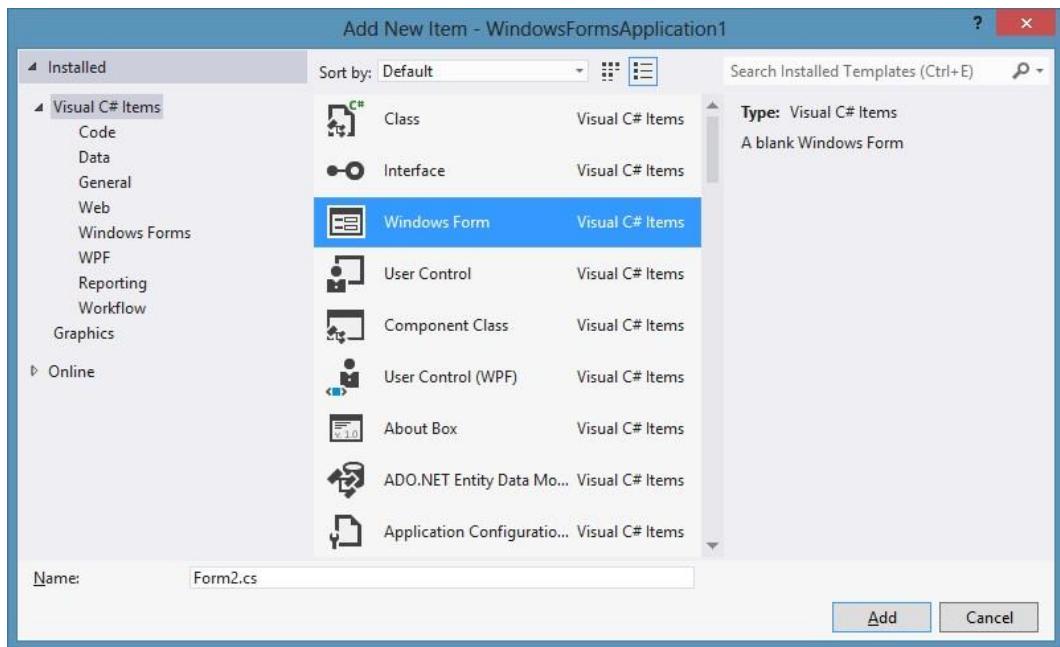
Analogno postoji i poruka kada se taster „otpusti” - `KeyUp` i ona je takođe tipa `KeyEventArgs`. Ukoliko korisnika interesuje da detektuje koji karakter je pritisnut, prethodne dve poruke nisu dobar izbor. `KeyPress` poruka se emituje ako korisnik pritisne taster koji je tipa karaktera. Dakle, ovo mora biti neki prepoznatljiv simbol. Mala slova, brojevi, znaci interpunkcije itd. Ukoliko se na primer pritisne `Shift+5` tj. taster % detektuje se kao jedan karakter tj. `KeyChar=37 '%'` [6].

## **6.6. Dodavanje novih *item-a* u projekat**

Jedan projekat može sadržati više item-a, tj. foldera i fajlova, najčešće su to klase i forme (kod WinForms aplikacija) [3].

Dodavanje forme je jednostavno, desni klik miša na naziv projekta u *Solution Explorer*-u i opcija *Add/Windows Forms*. Slično je i dodavanje klase, desni klik miša na naziv projekta u *Solution Explorer*-u i opcija *Add/Class* [3].

Analogno se dodaje item bilo kog tipa, desni klik miša na naziv projekta u *Solution Explorer*-u i opcija *Add/New Item*, pri čemu se prikazuje prozor sa slike 25 [3].



*Slika 25. Dodavanje novog item-a*

## 6.7. Build i run aplikacije

Pokretanje aplikacije (kompajliranje, tj. prevođenje koda) se vrši izborom opcije *Build Solution* ili klik na funkcionalni taster F6 što omogućava pokretanje kompajlera. Rezultati kompajliranja se vide na donjoj liniji okvira prozora. Ukoliko je kod napisan bez grešaka, dobija se poruka *Build Succeeded*, a ukoliko greške postoje u delu prozora za Error List (donji deo centralnog prozora) se prikazuje opis greške. Sintaksne greške se mogu uočiti i ranije, jer su podvučene crvenom linijom u kodu.

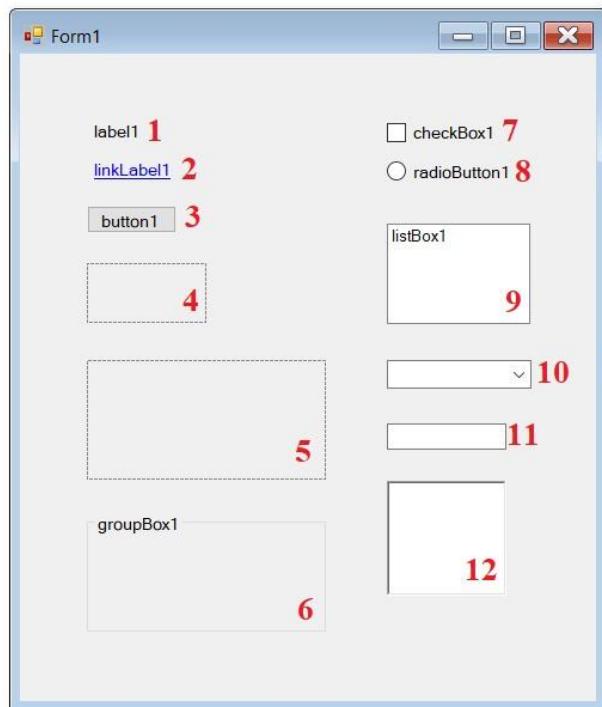
Kada se uklone sve greške iz Error List aplikacija se može „pokrenuti“ tj. startovati. Startovanje se može izvršiti bez pokretanja dibagera (padajuća lista *Debug/Start Without Debugging* ili *Ctrl+F5*) ili sa njim (padajuća lista *Debug/ Debugging* ili *F5*) [3].

## 6.8. Metode

O metodama smo već govorili u trećem poglavljju. Način pisanja metoda je isti kao u trećem poglavljju, kao i imena metode koje postoje u već napravljenim bibliotekama. U Praktikumu rešenih zadataka iz Programiranja 3 više ćemo koristiti i objasniti metode, a za svaku kontrolu koja je planirana nastavnim planom i programom da se radi iz Programiranja 3, tabelarno će se dati sve metode za te kontrole.

## 6.9. Windows kontrole

Na slici 26 su prikazane kontrole koje ćemo obraditi u okviru kursa Programiranje 3.



*Slika 26. Windows kontrole*

U tabeli 6 je pored broja kontrole dato ime kontrole.

*Tabela 6. Windows kontrole*

Broj kontrole sa slike 26	Ime kontrole
1	Label
2	LinkLabel
3	Button
4	PictureBox
5	Panel
6	GroupBox
7	CheckBox
8	RadioButton
9	ListBox
10	ComboBox
11	TextBox
12	RichTextBox

#### 6.9.1. Kontrola Button

[Button](#) je Windows kontrola koja se koristi za iniciranje jedne akcije. Akcija se izvodi kada se mišem klikne na dugme. Često se klasično dugme naziva i komandno dugme. Naravno, postoje i druge kontrole koje mogu inicirati akciju tako što se klikne na njih baš kao i kod dugmeta.

Vizuelno, dugme je kontrola od koje korisnik programa očekuje neku akciju kada izvrši „pritisak“ na njega.

Najčešće dugme u Windows aplikacijama je pravougaona kontrola (kontrola broj 3 sa slike 26) koja prikazuje reč ili kratku rečenicu. Tekst na dugmetu obično opisuje akciju koja sledi pritiskom na to dugme. U .Net aplikacijama ova kontrola se realizuje korišćenjem *Button* kontrole. Postavlja se na klasičan način korišćenjem liste *toolbox* ili programski. Kada postavite neko dugme na formu, nadalje možete izvesti njeno podešavanje korišćenjem svojstava ove kontrole preko *Properties* prozora [6].

Na dugme se može kliknuti mišem, tasterom *ENTER* ili *SPACEBAR* ako dugme ima fokus.

Podesite svojstvo forme *AcceptButton* ili *CancelButton* da korisnicima omogući da kliknu na dugme pritiskom na tastere *ENTER* ili *ESC*, čak i ako dugme nema fokus.

Kada prikažete formu pomoću metode *ShowDialog*, možete koristiti svojstvo *DialogResult* dugmeta da odredite povratnu vrednost *ShowDialog*.

Najbitnije svojstvo dugmeta za korisnika je poruka na njemu. Ova poruka zapisana je u svojstvu koje se naziva *Text*. Tekst se uvek prikazuje na vrhu kontrole i može biti dodatno formatiran.

Uobičajeno je da opisuje akciju koju dugme pokreće/izvršava. Najčešći tekstovi koje dugmad prikazuju su *OK* i *Cancel*. Poruka *OK* se obično prikazuje na dijalozima koji informišu korisnika o nekoj grešci, nekom stanju, ili je to samo informacija koja zahteva potvrdu od korisnika. Ukoliko se korisniku daje mogućnost da odustane od nastavka neke akcije, obično se nudi *Cancel* dugme.

Naravno, tekst na dugmetu može se programski menjati u toku izvršavanja aplikacije, u zavisnosti od stanja u kome se aplikacija nalazi [6].

Postoji još jedno svojstvo veoma značajno upravo za dugme. To je podrazumevano dugme. Samo jedno dugme na kontejneru kontrola može biti podrazumevano. Ono ima to svojstvo da pritiskom na taster Enter, kada je aktivna forma tog dugmeta, biva pritisnuto, tj. startuje se akcija koju ono inicira. Tačnije, ako neko drugo dugme ima fokus (vidljiv je tačkasti pravougaonik na njemu) izvršiće se akcija tog dugmeta. Inače, izvršava se akcija dugmeta koje je „podrazumevano“ (vidljiva je tamna ivica) [6].

Kada korisnik klikne na neko dugme da bi zatvorio dijalog, važno je da u kodu postoji informacija o tome na koje dugme je kliknuto. .Net okruženje pruža mehanizam kojim se može izvesti ova

identifikacija. Ovo se postiže korišćenjem svojstva *DialogResult*. Vrednost može biti neka iz određenog skupa koji je definisan. Moguće vrednosti su *None*, *OK*, *Cancel*, *Abort*, *Retry*, *Ignore*, *Yes*, i *No*. Podešavanjem neke od ovih vrednosti preko Properties prozora definišete vrednost koju dijalog vraća, ako se pritiskom na vaše dugme izvede zatvaranje istog dijaloga [6].

U tabeli 7 su data sva svojstva za kontrolu *Button*.

**Tabela 7.** Tabela svojstava za kontrolu *Button* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultActionDescription</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">AutoEllipsis</a>
<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>	<a href="#">AutoSizeMode</a>	<a href="#">BackColor</a>
<a href="#">BackgroundImage</a>	<a href="#">BackgroundImageLayout</a>	<a href="#">BindingContext</a>	<a href="#">Bottom</a>
<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>
<a href="#">CanSelect</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>	<a href="#">ClientRectangle</a>
<a href="#">ClientSize</a>	<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>
<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>
<a href="#">CreateParams</a>	<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>
<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>
<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>
<a href="#">DialogResult</a>	<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">Dock</a>
<a href="#">DoubleBuffered</a>	<a href="#">Enabled</a>	<a href="#">Events</a>	<a href="#">FlatAppearance</a>
<a href="#">FlatStyle</a>	<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>
<a href="#">ForeColor</a>	<a href="#">Handle</a>	<a href="#">HasChildren</a>	<a href="#">Height</a>
<a href="#">Image</a>	<a href="#">ImageAlign</a>	<a href="#">ImageIndex</a>	<a href="#">ImageKey</a>
<a href="#">ImageList</a>	<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>
<a href="#">IsAccessible</a>	<a href="#">IsDefault</a>	<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>
<a href="#">IsMirrored</a>	<a href="#">LayoutEngine</a>	<a href="#">Left</a>	<a href="#">Location</a>
<a href="#">Margin</a>	<a href="#">MaximumSize</a>	<a href="#">MinimumSize</a>	<a href="#">Name</a>
<a href="#">Padding</a>	<a href="#">Parent</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>
<a href="#">ProductVersion</a>	<a href="#">RecreatingHandle</a>	<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>
<a href="#">ResizeRedraw</a>	<a href="#">Right</a>	<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>

<a href="#">ShowFocusCues</a>	<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>	<a href="#">Size</a>
<a href="#">TabIndex</a>	<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>
<a href="#">TextAlign</a>	<a href="#">TextImageRelation</a>	<a href="#">Top</a>	<a href="#">TopLevelControl</a>
<a href="#">UseMnemonic</a>	<a href="#">UseCompatibleTextRendering</a>	<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>
<a href="#">WindowTarget</a>	<a href="#">UseVisualStyleBackColor</a>	<a href="#">Width</a>	

U tabeli 8 su date sve metode za kontrolu *Button*.

**Tabela 8.** Tabela metoda za kontrolu *Button* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">BeginInvoke(Delegate)</a>
<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>	<a href="#">Contains(Control)</a>
<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>
<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>	<a href="#">CreateObjRef(Type)</a>
<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>

<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>
<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>	<a href="#">FindForm()</a>
<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>
<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>
<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a>
<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>
<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>
<a href="#">Hide()</a>	<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>
<a href="#">Invalidate()</a>	<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>
<a href="#">Invalidate(Rectangle, Boolean)</a>	<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>

<a href="#">Invoke(Delegate)</a>	<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>
<a href="#">InvokeLostFocus(Control, EventArgs)</a>	<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>
<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>	<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>
<a href="#">LogicalToDeviceUnits(Int32)</a>	<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>
<a href="#">MemberwiseClone(Boolean)</a>	<a href="#">NotifyDefault(Boolean)</a>	<a href="#">NotifyInvalidate(Rectangle)</a>
<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>	<a href="#">OnBackgroundImageChange d(EventArgs)</a>
<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>	<a href="#">OnCausesValidationChanged(EventArgs)</a>
<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>	<a href="#">OnClientSizeChanged(EventArgs)</a>
<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>	<a href="#">OnControlAdded(ControlEventArgs)</a>
<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>	<a href="#">OnCursorChanged(EventArgs)</a>

<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>	<a href="#">OnDpiChangedAfterParent(EventArgs)</a>
<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>	<a href="#">OnDragEnter(DragEventArgs)</a>
<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>	<a href="#">OnEnabledChanged(EventArgs)</a>
<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>	<a href="#">OnForeColorChanged(EventArgs)</a>
<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>	<a href="#">OnHandleCreated(EventArgs)</a>

<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(HelpEventArgs)</a>	<a href="#">OnImeModeChanged(EventArgs)</a>
<a href="#">OnInvalidate(InvalidEventArgs)</a>	<a href="#">OnKeyDown(KeyEventEventArgs)</a>	<a href="#">OnKeyPress(KeyPressEventEventArgs)</a>
<a href="#">OnKeyUp(KeyEventEventArgs)</a>	<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>
<a href="#">OnLocationChanged(EventArgs)</a>	<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArgs)</a>
<a href="#">OnMouseCaptureChanged(EventArgs)</a>	<a href="#">OnMouseClick(MouseEventEventArgs)</a>	<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>
<a href="#">OnMouseDown(MouseEventEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>
<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEventEventArgs)</a>	<a href="#">OnMouseUp(MouseEventArgs)</a>
<a href="#">OnMouseWheel(MouseEventEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>
<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>
<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>
<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>
<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>
<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>

<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>
<a href="#">OnRightToLeftChanged(EventArgs)</a>	<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnStyleChanged(EventArgs)</a>
<a href="#">OnSystemColorsChanged(EventArgs)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>	<a href="#">OnTabStopChanged(EventArgs)</a>

<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>
<a href="#">OnVisibleChanged(EventArgs)</a>	<a href="#">PerformClick()</a>	<a href="#">PerformLayout()</a>
<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>	<a href="#">PointToScreen(Point)</a>
<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>	<a href="#">ProcessCmdKey(Message, Keys)</a>
<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>	<a href="#">ProcessKeyEventArgs(Message)</a>
<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>
<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>
<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>
<a href="#">RectangleToScreen(Rectangle)</a>	<a href="#">Refresh()</a>	<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>
<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>	<a href="#">ResetCursor()</a>
<a href="#">ResetFlagsandPaint()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>
<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>
<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>
<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>

<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>
<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>
<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>
<a href="#">ToString()</a>	<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>
<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>	

U tabeli 9 su dati svi događaji za kontrolu *Button*.

**Tabela 9.** Tabela događaja za kontrolu *Button* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>
<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>
<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidated</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>
<a href="#">MouseCaptureChanged</a>	<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>
<a href="#">MouseDown</a>	<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>
<a href="#">MouseLeave</a>	<a href="#">MouseMove</a>	<a href="#">MouseUp</a>
<a href="#">MouseWheel</a>	<a href="#">Move</a>	<a href="#">PaddingChanged</a>
<a href="#">Paint</a>	<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>

<a href="#">QueryAccessibilityHelp</a>	<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>
<a href="#">Resize</a>	<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>	<a href="#">Validated</a>
<a href="#">Validating</a>	<a href="#">VisibleChanged</a>	

### 6.9.2. Kontrola RadioButton

Radio dugme ili [RadioButton](#) (kontrola broj 8 sa slike 26), neko ga naziva još i opciono dugme, kontrola je kružnog oblika koja se pojavljuje u grupama sa drugima kontrolama istog tipa, tj. više radio dugmadi pojavljuju se zajedno. Svako dugme sastoji se od malog praznog kruga i labele koja opisuje dugme, tj. daje korisniku neku informaciju o akciji. Kada korisnik pritisne jedno od dugmića koji su u grupi, ono dobija oblik , a ostali ostaju prazni. Ovo znači da se grupom radio dugmića realizuje korisnička akcija izbora samo jedne stavke iz grupe [6].

Kontrola RadioButton može prikazati tekst, sliku ili oboje.

Kada korisnik izabere jedno dugme sa opcijama (poznato i kao radio dugme) unutar grupe, ostale se automatski brišu. Sve kontrole *RadioButton*-a u datom okviru, čine grupu. Da bi se stvorile više grupa u jednoj formi, koriste se kontrole *GroupBox* (kontrola broj 6 sa slike 26) ili panel (kontrola broj 5 sa slike 26).

Kontrole *RadioButton* i *CheckBok* imaju sličnu funkciju: nude izbor koji korisnik može odabrati ili obrisati. Razlika je u tome što se istovremeno može odabrati više kontrola *CheckBok*, za razliku od kontrole *RadioButton*, gde je uvek samo jedna kontrola čekirana.

Ovo svojstvo definiše da li je radio dugme selektovano ili ne. Ukoliko jeste, to znači da su sva ostala neselektovana. Osim za čitanje stanja u kome se dugme nalazi isto svojstvo možete upotrebiti i u „suprotnom pravcu”. Ovim svojstvom možete programski podešavati izbor nekog radio-dugmeta [6].

Podrazumevano podešavanje izgleda radio dugmeta je takvo da je ono ispunjeno velikom tačkom ako korisnik selektuje to dugme. Opciono, možete podesiti da ono radi kao *toggle* dugme. Šta to znači? Samo jedno dugme u grupi može biti u donjem položaju (stanje selektovano) dok su ostala u gornjem položaju. Ako se neko drugo dugme pritisne, ono ide dole, a dugme koje je prethodno

bilo u donjem položaju ide gore, tj. sva ostala su otpuštena. Promena podrazumevanog prikazivanja ove kontrole izvodi se preko svojstva *Apearance* [6].

U tabeli 10 su data sva svojstva za kontrolu *RadioButton*.

**Tabela 10.** Tabela svojstava za kontrolu *RadioButton* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultAction</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">Description</a>			
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">Appearance</a>
<a href="#">AutoCheck</a>	<a href="#">AutoEllipsis</a>	<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>
<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>	<a href="#">BackgroundImageLayout</a>	<a href="#">BindingContext</a>
<a href="#">Bottom</a>	<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>
<a href="#">CanRaiseEvents</a>	<a href="#">CanSelect</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>
<a href="#">CheckAlign</a>	<a href="#">Checked</a>	<a href="#">ClientRectangle</a>	<a href="#">ClientSize</a>
<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>
<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>	<a href="#">CreateParams</a>
<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>	<a href="#">DefaultImeMode</a>
<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultPadding</a>
<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>	<a href="#">DisplayRectangle</a>
<a href="#">Disposing</a>	<a href="#">Dock</a>	<a href="#">DoubleBuffered</a>	<a href="#">Enabled</a>
<a href="#">Events</a>	<a href="#">FlatAppearance</a>	<a href="#">FlatStyle</a>	<a href="#">Focused</a>
<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>	<a href="#">Handle</a>
<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">Image</a>	<a href="#">ImageAlign</a>
<a href="#">ImageIndex</a>	<a href="#">ImageKey</a>	<a href="#">ImageList</a>	<a href="#">ImeMode</a>
<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>	<a href="#">IsDefault</a>
<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>	<a href="#">LayoutEngine</a>
<a href="#">Left</a>	<a href="#">Location</a>	<a href="#">Margin</a>	<a href="#">MaximumSize</a>
<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>	<a href="#">Parent</a>
<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>	<a href="#">RecreatingHandle</a>
<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>	<a href="#">Right</a>
<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ShowFocusCues</a>	<a href="#">ShowKeyboardCues</a>
<a href="#">Site</a>	<a href="#">Size</a>	<a href="#">TabIndex</a>	<a href="#">TabStop</a>

<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#">TextAlign</a>	<a href="#">TextImageRelation</a>
<a href="#">Top</a>	<a href="#">TopLevelControl</a>	<a href="#">UseCompatibleTextRendering</a>	<a href="#">UseMnemonic</a>
<a href="#">UseVisualStyleBackColor</a> <a href="#">lor</a>	<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>	<a href="#">Width</a>
<a href="#">WindowTarget</a>			

U tabeli 11 su date sve metode za kontrolu *RadioButton*.

**Tabela 11.** Tabela metoda za kontrolu *RadioButton* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">BeginInvoke(Delegate)</a>
<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>	<a href="#">Contains(Control)</a>
<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>
<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>	<a href="#">CreateObjRef(Type)</a>
<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>
<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>
<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>	<a href="#">FindForm()</a>
<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>
<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>
<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a> <a href="#">GetPreferredSize(Size)</a>
<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>	<a href="#">GetStyle(ControlStyles)</a>
<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>	<a href="#">Hide()</a>
<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>

<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>
<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>
<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>
<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>
<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>
<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnCheckedChanged(EventArgs)</a>
<a href="#">OnClickEventArgs)</a>	<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>
<a href="#">OnContextMenuStripChanged(EventArgs)</a>	<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>
<a href="#">OnCreateControl()</a>	<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDockChanged(EventArgs)</a>
<a href="#">OnDoubleClick(EventArgs)</a>	<a href="#">OnDpiChangedAfterParent(EventArgs)</a>	<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>
<a href="#">OnDragDrop(DragEventArgs)</a>	<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>
<a href="#">OnDragOver(DragEventArgs)</a>	<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>
<a href="#">OnFontChanged(EventArgs)</a>	<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>
<a href="#">OnGotFocus(EventArgs)</a>	<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>

<a href="#">OnHelpRequested(HelperEventArgs)</a>	<a href="#">OnImeModeChanged(EventArgs)</a>	<a href="#">OnInvalidate(InvalidateEventArgs)</a>
<a href="#">OnKeyDown(KeyEventEventArgs)</a>	<a href="#">OnKeyPress(KeyPressEventArgs)</a>	<a href="#">OnKeyUp(KeyEventEventArgs)</a>
<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>

<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnMouseCaptureChanged(EventArgs)</a>
<a href="#">OnMouseClick(MouseEventArgs)</a>	<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>	<a href="#">OnMouseDown(MouseEventArgs)</a>
<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>	<a href="#">OnMouseLeave(EventArgs)</a>
<a href="#">OnMouseMove(MouseEventArgs)</a>	<a href="#">OnMouseUp(MouseEventArgs)</a>	<a href="#">OnMouseWheel(MouseEventArgs)</a>
<a href="#">OnMove(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>	<a href="#">OnPaddingChanged(EventArgs)</a>
<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>	<a href="#">OnParentBackColorChanged(EventArgs)</a>
<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>	<a href="#">OnParentChanged(EventArgs)</a>
<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>	<a href="#">OnParentFontChanged(EventArgs)</a>
<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>	<a href="#">OnParentVisibleChanged(EventArgs)</a>
<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>
<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>
<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>

<a href="#">OnTabIndexChanged(EventArgs)</a>	<a href="#">OnTabStopChanged(EventArgs)</a> )	<a href="#">OnTextChanged(EventArgs)</a>
<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnVisibleChanged(EventArgs)</a> )
<a href="#">PerformClick()</a>	<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>
<a href="#">PointToClient(Point)</a>	<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>
<a href="#">PreProcessMessage(Message)</a>	<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>

<a href="#">ProcessDialogKey(Keys)</a>	<a href="#">ProcessKeyEventArgs(Message)</a> )	<a href="#">ProcessKeyMessage(Message)</a> )
<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>
<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>
<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle)</a> )
<a href="#">Refresh()</a>	<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>
<a href="#">ResetBindings()</a>	<a href="#">ResetCursor()</a>	<a href="#">ResetFlagsandPaint()</a>
<a href="#">ResetFont()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetImeMode()</a>
<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>	<a href="#">ResetText()</a>
<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>	<a href="#">RtlTranslateAlignment(ContentAlignment)</a>
<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>	<a href="#">RtlTranslateContent(ContentAlignment)</a>
<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>	<a href="#">Scale(Single)</a>

<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>	<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>
<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>	<a href="#">Select()</a>
<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>	<a href="#">SendToBack()</a>
<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>
<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>	<a href="#">SetStyle(ControlStyles, Boolean)</a>
<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>	<a href="#">Show()</a>
<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>	<a href="#">ToString()</a>
<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>	<a href="#">UpdateZOrder()</a>
<a href="#">WndProc(Message)</a>		

U tabeli 12 su dati svi događaji za kontrolu RadioButton.

**Tabela 12. Tabela događaja za kontrolu RadioButton [7]**

<a href="#">AppearanceChanged</a>	<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>
<a href="#">BackgroundImageChanged</a>	<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>
<a href="#">CausesValidationChanged</a>	<a href="#">ChangeUICues</a>	<a href="#">CheckedChanged</a>
<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>	<a href="#">ContextMenuChanged</a>
<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>	<a href="#">ControlRemoved</a>
<a href="#">CursorChanged</a>	<a href="#">Disposed</a>	<a href="#">DockChanged</a>
<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>	<a href="#">DpiChangedBeforeParent</a>
<a href="#">DragDrop</a>	<a href="#">DragEnter</a>	<a href="#">DragLeave</a>
<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>	<a href="#">Enter</a>
<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>	<a href="#">GiveFeedback</a>

<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>	<a href="#">HandleDestroyed</a>
<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>	<a href="#">Invalidate</a>
<a href="#">KeyDown</a>	<a href="#">KeyPress</a>	<a href="#">KeyUp</a>
<a href="#">Layout</a>	<a href="#">Leave</a>	<a href="#">LocationChanged</a>
<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>	<a href="#">MouseCaptureChanged</a>
<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>	<a href="#">MouseDown</a>
<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>	<a href="#">MouseLeave</a>
<a href="#">MouseMove</a>	<a href="#">MouseUp</a>	<a href="#">MouseWheel</a>
<a href="#">Move</a>	<a href="#">PaddingChanged</a>	<a href="#">Paint</a>
<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>	<a href="#">QueryAccessibilityHelp</a>
<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>	<a href="#">Resize</a>
<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>	<a href="#">StyleChanged</a>
<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>	<a href="#">TabStopChanged</a>
<a href="#">TextChanged</a>	<a href="#">Validated</a>	<a href="#">Validating</a>
<a href="#">VisibleChanged</a>		

### 6.9.3. Kontrola CheckBox

Polje za potvrdu ili [CheckBox](#) je kontrola koja daje samo dve opcije: potvrđeno ili prazno (kontrola broj 7 sa slike 26). Vizuelno ova kontrola prikazuje malo kvadratno polje na koje korisnik može da klikne (tačnije, korisnik može da klikne na celu kontrolu – uključujući tekst) . Na startu polje je prazno. Ako korisnik klikne na njega, oznaka o potvrdi pojavljuje se u kvadratnom polju. Da bi se korisniku omogućilo da zna šta polje za potvrdu predstavlja, kontroli je pridružena labela koja prikazuje tekst. Kada je kvadratno polje prazno , vrednost je *false*. Kada je polje ispunjeno oznakom za potvrdu , vrednost je *true* [6].

Koristite *CheckBox* da biste dali korisniku opciju, poput *true/false* ili da/ne. Kontrola *CheckBox* može prikazati sliku ili tekst ili oboje.

Kontrole *CheckBox* i *RadioButton* imaju sličnu funkciju: omogućavaju korisniku da izabere sa liste opcija. *CheckBox* kontrole omogućavaju korisniku da odabere više opcija. Suprotno tome, *RadioButton* kontrola omogućava da korisnik izabere više opcija.

Podrazumevano ponašanje ove kontrole je da se pojavljuje neštriklirana (nepotvrđena), tj. njena vrednost je inicijalno postavljena na *False*. Da bi se promenila, korisnik treba da klikne na

kontrolu. Međutim, ako želimo da je svojstvo inicijalno postavljeno na vrednost *True*, tj. da kontrola pri prvom pojavljivanju bude potvrđena, možemo promeniti i vrednost svojstva *Checked* na *true* [6].

Da biste proverili da li je polje selektovano treba da pročitate vrednost svojstva *Checked*.

Podrazumevano, kvadratić se pojavljuje na levoj strani labele. .Net okruženje vam pruža puno različitih izbora kojima možete podešavati prikaz kontrole. To se na ovoj kontroli može potvrditi. Najčešća podešavanja tiču se pozicije kvadratića u odnosu na tekst cele kontrole. Ta podešavanja se izvode pomoću svojstva *CheckAlign*. Moguće vrednosti su: *TopLeft*, *TopCenter*, *TopRight*, *MiddleRight*, *BottomRight*, *BottomCenter* i *BottomLeft*. Programski podešavanja izvodite postavljanjem vrednosti *CheckAlign*, ali preko enumeratora *ContentAlignment* [6].

Neretko, kontrola *CheckBox* osim stanja *True*, odnosno *False*, treba da pokaže i neko međustanje. Ovo stanje obično se naziva *half-checked* ili u nekom slobodnom prevodu polupotvrđeno. U tom stanju polje za potvrdu izgleda kao da je onemogućeno (disabled). Ovakvo specifično ponašanje kontroliše se preko svojstva *CheckState*, da bi se podesilo međustanje postavlja se vrednost ovog svojstva na *Indeterminate* [6].

Jedino svojstvo *CheckState* dozvoljava postavljanje polja za potvrdu u međustanju. Ako želite da se koriste tri stanja dugmeta za potvrdu, na primer označeno, poluozačeno i neoznačeno, koristite svojstvo *ThreeState*. Postavljanjem svojstva *CheckState* (*boolean* tipa) na vrednost *True* korisnik treba da klikne dva do tri puta do željene vrednosti. Dobijanje stanja dugmeta za potvrdu u ovom slučaju izvodite preko svojstva *Indeterminate* [6].

Očekivani izgled ove kontrole je kvadratić koji može da primi komandu od korisnika aplikacije ili prikaže neku tekuću vrednost. Međutim, alternativa je da polje za potvrdu prikažete i kao standardno dugme. U tom slučaju, kada kliknete na njega, ono ostaje u donjem položaju, tj. položaj dugmeta dole ili gore određuje u stvari stanje polja za potvrdu *True* odnosno *False*. Ako korisnik klikne na dugme još jednom, ono se podiže u gornji položaj. Da biste promenili standardni prikaz treba promeniti svojstvo *Appearance*. Sa vrednosti *Normal* prebaciti na vrednost *Button* [6].

U tabeli 13 su data sva svojstva za kontrolu *CheckBox*.

**Tabela 13.** Tabela svojstava za kontrolu *CheckBox* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultActionDescription</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">Appearance</a>
<a href="#">AutoCheck</a>	<a href="#">AutoEllipsis</a>	<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>

<a href="#">BackColor</a>	<a href="#">BackgroundImageLayout</a>	<a href="#">BackgroundImage</a>	<a href="#">BindingContext</a>
<a href="#">Bottom</a>	<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>
<a href="#">CanRaiseEvents</a>	<a href="#">CanSelect</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>
<a href="#">CheckAlign</a>	<a href="#">Checked</a>	<a href="#">CheckState</a>	<a href="#">ClientRectangle</a>
<a href="#">ClientSize</a>	<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>
<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>
<a href="#">CreateParams</a>	<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>
<a href="#">DefaultImeMode</a>	<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMargin</a>
<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>
<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">DoubleBuffered</a>	<a href="#">Dock</a>
<a href="#">Enabled</a>	<a href="#">Events</a>	<a href="#">FlatAppearance</a>	<a href="#">FlatStyle</a>
<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>
<a href="#">Handle</a>	<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">Image</a>
<a href="#">ImageAlign</a>	<a href="#">ImageIndex</a>	<a href="#">ImageKey</a>	<a href="#">ImageList</a>
<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>
<a href="#">IsDefault</a>	<a href="#">IsDisposed</a>	<a href="#">sHandleCreated</a>	<a href="#">IsMirrored</a>
<a href="#">LayoutEngine</a>	<a href="#">Left</a>	<a href="#">Location</a>	<a href="#">Margin</a>
<a href="#">MaximumSize</a>	<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>
<a href="#">Parent</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>
<a href="#">RecreatingHandle</a>	<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>
<a href="#">Right</a>	<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ShowFocusCues</a>
<a href="#">Site</a>	<a href="#">ShowKeyboardCues</a>	<a href="#">Size</a>	<a href="#">TabIndex</a>
<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#"> TextAlign</a>
<a href="#">TextImageRelation</a>	<a href="#">ThreeState</a>	<a href="#">Top</a>	<a href="#">TopLevelControl</a>
<a href="#">UseMnemonic</a>	<a href="#">UseCompatibleTextRendering</a>	<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>
<a href="#">Width</a>	<a href="#">UseVisualStyleBackColor</a>	<a href="#">WindowTarget</a>	

U tabeli 14 su date sve metode za kontrolu *CheckBox*.

**Tabela 14.** Tabela metoda za kontrolu *CheckBox* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">BeginInvoke(Delegate)</a>
---------------------------------------------------------------------	----------------------------------------------------------------------------	---------------------------------------

<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>	<a href="#">Contains(Control)</a>
<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>
<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>	<a href="#">CreateObjRef(Type)</a>
<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>
<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>
<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>	<a href="#">FindForm()</a>
<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>
<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>
<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a>
<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>
<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>

<a href="#">Hide()</a>	<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>
<a href="#">Invalidate()Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>
<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>
<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>
<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>
<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAppearanceChanged(EventArgs)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>

<a href="#">OnBackColorChanged(EventArgs)</a>	<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>
<a href="#">OnBindingContextChanged(EventArgs)</a>	<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>
<a href="#">OnCheckedChanged(EventArgs)</a>	<a href="#">OnCheckStateChanged(EventArgs)</a>	<a href="#">OnClick(EventArgs)</a>
<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>
<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>
<a href="#">OnDpiChangedAfterParent(EventArgs)</a>	<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>
<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>
<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>
<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(HelpEventArgs)</a>

<a href="#">OnImeModeChanged(EventArgs)</a>	<a href="#">OnInvalidate(InvalidateEventArgs)</a>	<a href="#">OnKeyDown(KeyEventArgs)</a>
<a href="#">OnKeyPress(KeyPressEventArgs)</a>	<a href="#">OnKeyUp(KeyEventArgs)</a>	<a href="#">OnLayout(LayoutEventArgs)</a>
<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>	<a href="#">OnLostFocus(EventArgs)</a>
<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnMouseCaptureChanged(EventArgs)</a>	<a href="#">OnMouseClick(MouseEventArgs)</a>

<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>	<a href="#">OnMouseDown(MouseEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>
<a href="#">OnMouseHover(EventArgs)</a>	<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEventArgs)</a>
<a href="#">OnMouseUp(MouseEventArgs)</a>	<a href="#">OnMouseWheel(MouseEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>
<a href="#">OnNotifyMessage(Message)</a>	<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>
<a href="#">OnPaintBackground(PaintEventArgs)</a>	<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>
<a href="#">OnParentBindingContextChanged(EventArgs)</a>	<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>
<a href="#">OnParentEnabledChanged(EventArgs)</a>	<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>
<a href="#">OnParentRightToLeftChanged(EventArgs)</a>	<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>
<a href="#">OnPrint(PaintEventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a>
<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>	<a href="#">OnSizeChanged(EventArgs)</a>
<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>
<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnValidated(EventArgs)</a>
<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnVisibleChanged(EventArgs)</a>	<a href="#">PerformLayout()</a>
<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>	<a href="#">PointToScreen(Point)</a>

<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>	<a href="#">ProcessCmdKey(Message, Keys)</a>
<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>	<a href="#">ProcessKeyEventArgs(Message)</a>

<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>
<a href="#">}</a>		
<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>
<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>
<a href="#">RectangleToScreen(Rectangle)</a>	<a href="#">Refresh()</a>	<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>
<a href="#">}</a>		
<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>	<a href="#">ResetCursor()</a>
<a href="#">ResetFlagsandPaint()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>
<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>
<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalIAignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalIAignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>
<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>
<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>
<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>
<a href="#">ToString()</a>	<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>

<a href="#">UpdateBounds(Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>
<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>	

U tabeli 15 su dati svi događaji za kontrolu *CheckBox*.

**Tabela 15.** Tabela događaja za kontrolu *CheckBox* [7]

<a href="#">AppearanceChanged</a>	<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>
<a href="#">BackgroundImageChanged</a>	<a href="#">BackgroundImageLayoutChange</a>	<a href="#">BindingContextChanged</a>
<a href="#">CausesValidationChanged</a>	<a href="#">ChangeUICues</a>	<a href="#">CheckedChanged</a>
<a href="#">CheckStateChanged</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>
<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>
<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidated</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>
<a href="#">MouseCaptureChanged</a>	<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>
<a href="#">MouseDown</a>	<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>
<a href="#">MouseLeave</a>	<a href="#">MouseMove</a>	<a href="#">MouseUp</a>
<a href="#">MouseWheel</a>	<a href="#">Move</a>	<a href="#">PaddingChanged</a>
<a href="#">Paint</a>	<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>
<a href="#">QueryAccessibilityHelp</a>	<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>
<a href="#">Resize</a>	<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>	<a href="#">Validated</a>

<a href="#">Validating</a>	<a href="#">VisibleChanged</a>	
----------------------------	--------------------------------	--

#### 6.9.4. Kontrola Label

Kontrola [Label](#) (kontrola broj 1 sa slike 26) obično se koriste za prikaz teksta. Na primer, možete da koristite *Label* kontrolu za dodavanje opisnog teksta za *TextBox* kontrolu da bi se informisao korisnik o vrsti podataka koji se očekuju da se unese u *TextBox* kontrolu. Kontrola *Label* se takođe, koristiti i za dodavanje opisnog teksta za pružanje korisnih informacija o aplikaciji. Na primer, možete se u svojstvo *Text* dodati tekst kao uputstvo za korišćenje aplikacije, a kontrola *Label* može da se nalazi na vrhu ili dnu aplikacije.

U tabeli 16 su data sva svojstva za kontrolu *Label*.

**Tabela 16.** Tabela svojstava za kontrolu *Label* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleDefaultActionDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">AutoEllipsis</a>
<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>
<a href="#">BackgroundImageLayout</a>	<a href="#">BindingContext</a>	<a href="#">BorderStyle</a>	<a href="#">Bottom</a>
<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>
<a href="#">CanSelect</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>	<a href="#">ClientRectangle</a>
<a href="#">ClientSize</a>	<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>
<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>
<a href="#">CreateParams</a>	<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>
<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultSize</a>
<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultPadding</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>
<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">DoubleBuffered</a>	<a href="#">Dock</a>
<a href="#">Enabled</a>	<a href="#">Events</a>	<a href="#">FlatStyle</a>	<a href="#">Focused</a>
<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>	<a href="#">Handle</a>
<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">Image</a>	<a href="#">ImageAlign</a>
<a href="#">ImageIndex</a>	<a href="#">ImageKey</a>	<a href="#">ImageList</a>	<a href="#">ImeMode</a>
<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>	<a href="#">IsDisposed</a>
<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>	<a href="#">LayoutEngine</a>	<a href="#">Left</a>

<a href="#">LiveSetting</a>	<a href="#">Location</a>	<a href="#">MaximumSize</a>	<a href="#">Margin</a>
<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>	<a href="#">Parent</a>
<a href="#">PreferredHeight</a>	<a href="#">PreferredSize</a>	<a href="#">PreferredWidth</a>	<a href="#">ProductName</a>
<a href="#">ProductVersion</a>	<a href="#">RecreatingHandle</a>	<a href="#">RenderRightToLeft</a>	<a href="#">Region</a>
<a href="#">RenderTransparent</a>	<a href="#">ResizeRedraw</a>	<a href="#">RightToLeft</a>	<a href="#">Right</a>
<a href="#">ScaleChildren</a>	<a href="#">ShowFocusCues</a>	<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>
<a href="#">Size</a>	<a href="#">TabIndex</a>	<a href="#">TabStop</a>	<a href="#">Tag</a>
<a href="#">Text</a>	<a href="#"> TextAlign</a>	<a href="#">TopLevelControl</a>	<a href="#">Top</a>
<a href="#">UseMnemonic</a>	<a href="#">UseWaitCursor</a>	<a href="#">UseCompatibleTextRendering</a>	<a href="#">Visible</a>
<a href="#">Width</a>	<a href="#">WindowTarget</a>		

U tabeli 17 su date sve metode za kontrolu *Label*.

**Tabela 17.** Tabela metoda za kontrolu *Label* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">BeginInvoke(Delegate)</a>
<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>	<a href="#">CalcImageRenderBounds(Image, Rectangle, ContentAlignment)</a>
<a href="#">Contains(Control)</a>	<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>
<a href="#">CreateControlsInstance()</a>	<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>
<a href="#">CreateObjRef(Type)</a>	<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>
<a href="#">Dispose()</a>	<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>
<a href="#">DrawImage(Graphics, Image, Rectangle, ContentAlignment)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>	<a href="#">EndInvoke(IAsyncResult)</a>
<a href="#">Equals(Object)</a>	<a href="#">FindForm()</a>	<a href="#">Focus()</a>
<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>	<a href="#">GetChildAtPoint(Point)</a>

<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>	<a href="#">GetHashCode()</a>
<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a>	<a href="#">GetPreferredSize(Size)</a>

<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>	<a href="#">GetStyle(ControlStyles)</a>
<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>	<a href="#">Hide()</a>
<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>
<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>
<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>
<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>
<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>
<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>
<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>
<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>
<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>

<a href="#">OnDpiChangedAfterParent(Even tArgs)</a>	<a href="#">OnDpiChangedBeforeParent(Eve ntArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>
<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>
<a href="#">OnEnabledChanged(EventArgs)</a> }	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventAr gs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackE ventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>
<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(Helper Args)</a>

<a href="#">OnImeModeChanged(EventAr gs)</a>	<a href="#">OnInvalidate(InvalidEventAr gs)</a>	<a href="#">OnKeyDown(KeyEventArgs)</a>
<a href="#">OnKeyPress(KeyPressEventArgs)</a> }	<a href="#">OnKeyUp(KeyEventArgs)</a>	<a href="#">OnLayout(LayoutEventArgs)</a>
<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>	<a href="#">OnLostFocus(EventArgs)</a>
<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnMouseCaptureChanged(Event Args)</a>	<a href="#">OnMouseClick(MouseEventAr gs)</a>
<a href="#">OnMouseDoubleClick(MouseEvent EventArgs)</a>	<a href="#">OnMouseDown(MouseEventArg s)</a>	<a href="#">OnMouseEnter(EventArgs)</a>
<a href="#">OnMouseHover(EventArgs)</a>	<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEvent Args)</a>
<a href="#">OnMouseUp(MouseEventArgs)</a>	<a href="#">OnMouseWheel(MouseEventArg s)</a>	<a href="#">OnMove(EventArgs)</a>
<a href="#">OnNotifyMessage(Message)</a>	<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>
<a href="#">OnPaintBackground(PaintEvent Args)</a>	<a href="#">OnParentBackColorChanged(Eve ntArgs)</a>	<a href="#">OnParentBackgroundImageC hanged(EventArgs)</a>
<a href="#">OnParentBindingContextChan ged(EventArgs)</a>	<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(Eve ntArgs)</a>
<a href="#">OnParentEnabledChanged(Eve ntArgs)</a>	<a href="#">OnParentFontChanged(EventArg s)</a>	<a href="#">OnParentForeColorChanged( EventArgs)</a>

<a href="#">OnParentRightToLeftChanged(EventArgs)</a>	<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>
<a href="#">OnPrint(PaintEventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a>
<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>	<a href="#">OnSizeChanged(EventArgs)</a>
<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>
<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextAlignChanged(EventArgs)</a>	<a href="#">OnTextChanged(EventArgs)</a>
<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnVisibleChanged(EventArgs)</a>
<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>

<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>
<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>
<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>
<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>
<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>
<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle)</a>	<a href="#">Refresh()</a>
<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>
<a href="#">ResetCursor()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>
<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>

<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>
<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>
<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>
<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>
<a href="#">ToString()</a>	<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>
<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>	

U tabeli 18 su dati svi događaji za kontrolu *Label*.

**Tabela 18.** Tabela događaja za kontrolu *Label* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>

<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>
<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidated</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>
<a href="#">MouseCaptureChanged</a>	<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>
<a href="#">MouseDown</a>	<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>
<a href="#">MouseLeave</a>	<a href="#">MouseMove</a>	<a href="#">MouseUp</a>
<a href="#">MouseWheel</a>	<a href="#">Move</a>	<a href="#">PaddingChanged</a>
<a href="#">Paint</a>	<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>
<a href="#">QueryAccessibilityHelp</a>	<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>
<a href="#">Resize</a>	<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#">TextAlignChanged</a>	<a href="#">TextChanged</a>
<a href="#">Validated</a>	<a href="#">Validating</a>	<a href="#">VisibleChanged</a>

#### 6.9.5. Kontrola TextBox

Kontrola [TextBox](#) koristi se za prikaz tekstualnih poruka ili za unošenje teksta (kontrola broj 11 sa slike 26). Tekst polje je kontrola pravougaonog oblika. Površina za prikaz, odnosno unos teksta, zauzima celu površinu kontrole. Zbog toga uz ovo tekst polje, uobičajeno, ide label kontrola kojom se opisuje namena tekstualnog polja [6].

*TextBox* kontrola ima dodatnu funkciju koja se ne nalazi u standardnoj Windows kontroli teksta, uključujući multiline uređivanje i maskiranje znakova lozinke.

Obično se *TextBox* kontrola koristi da prikaže ili prihvati kao ulaz jedan redak teksta. Može se koristiti svojstvo *Multiline* i *ScrollBars* da bi se omogućilo prikazivanje ili unošenje više redova teksta. Podešavanjem svojstva *AcceptsTab* i *AcceptsReturn* na *true*, omogućava se veća manipulacija tekstrom u višelinjskoj *TextBox* kontroli.

Količinu teksta unetog u *TextBox* kontrolu može se ograničiti podešavanjem svojstva *MaxLength* na određeni broj znakova. *TextBox* kontrola se takođe može koristiti za prihvatanje lozinke i drugih osjetljivih informacija. Može se koristiti svojstvo *PasswordChar* za maskiranje znakova unetih u jednorednoj verziji kontrole. Korišćenjem svojstva *CharacterCasing* omogućavamo korisniku da u tekstualnu kontrolu unese samo velika slova, samo mala slova ili kombinaciju velikih i malih slova.

Da bi se prelistao sadržaj *TextBox*-a dok se pokazivač ne nađe u vidljivoj oblasti kontrole, može se koristiti metoda *ScrollToCaret*. Da bi se odabrao opseg teksta u tekstnom polju, može da se koristite metoda *Select*.

Da bi se ograničio unos teksta u *TextBox* kontrolu, može da se koristi događaj *KeyDown*, da bi se potvrdio svaki znak koji se unosi u kontrolu. Takođe, može se ograničiti sav unos podataka u *TextBox* kontrolu tako što će se postaviti svojstvo *ReadOnly* na *true*.

U tabeli 19 su data sva svojstva za kontrolu *TextBox*.

**Tabela 19.** Tabela svojstava za kontrolu *TextBox* [7]

<a href="#">AcceptsReturn</a>	<a href="#">AcceptsTab</a>	<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultAction</a> <a href="#">Description</a>
<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>	<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>
<a href="#">Anchor</a>	<a href="#">AutoCompleteCustomSource</a>	<a href="#">AutoCompleteMode</a>	<a href="#">AutoCompleteSource</a>
<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>
<a href="#">BackgroundImageLayout</a>	<a href="#">BindingContext</a>	<a href="#">BorderStyle</a>	<a href="#">Bottom</a>
<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>
<a href="#">CanSelect</a>	<a href="#">CanUndo</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>
<a href="#">CharacterCasing</a>	<a href="#">ClientRectangle</a>	<a href="#">ClientSize</a>	<a href="#">CompanyName</a>
<a href="#">Container</a>	<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>
<a href="#">Controls</a>	<a href="#">Created</a>	<a href="#">CreateParams</a>	<a href="#">Cursor</a>

<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>	<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>
<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>
<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>	<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>
<a href="#">Dock</a>	<a href="#">DoubleBuffered</a>	<a href="#">Enabled</a>	<a href="#">Events</a>
<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>
<a href="#">Handle</a>	<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">HideSelection</a>
<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>
<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>	<a href="#">LayoutEngine</a>
<a href="#">Left</a>	<a href="#">Lines</a>	<a href="#">Location</a>	<a href="#">Margin</a>
<a href="#">MaximumSize</a>	<a href="#">MaxLength</a>	<a href="#">MinimumSize</a>	<a href="#">Modified</a>
<a href="#">Multiline</a>	<a href="#">Name</a>	<a href="#">Padding</a>	<a href="#">Parent</a>
<a href="#">PasswordChar</a>	<a href="#">PlaceholderText</a>	<a href="#">PreferredHeight</a>	<a href="#">PreferredSize</a>
<a href="#">ProductName</a>	<a href="#">ProductVersion</a>	<a href="#">ReadOnly</a>	<a href="#">RecreatingHandle</a>
<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>	<a href="#">Right</a>
<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ScrollBars</a>	<a href="#">SelectedText</a>
<a href="#">SelectionLength</a>	<a href="#">SelectionStart</a>	<a href="#">ShortcutsEnabled</a>	<a href="#">ShowFocusCues</a>
<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>	<a href="#">Size</a>	<a href="#">TabIndex</a>
<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#"> TextAlign</a>
<a href="#">TextLength</a>	<a href="#">Top</a>	<a href="#">TopLevelControl</a>	<a href="#">UseSystemPasswordChar</a>
<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>	<a href="#">Width</a>	<a href="#">WindowTarget</a>
<a href="#">WordWrap</a>			

U tabeli 20 su date sve metode za kontrolu *TextBox*.

**Tabela 20.** Tabela metoda za kontrolu *TextBox* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">AppendText(String)</a>
<a href="#">BeginInvoke(Delegate)</a>	<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>
<a href="#">Clear()</a>	<a href="#">ClearUndo()</a>	<a href="#">Contains(Control)</a>

<a href="#">Copy()</a>	<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>
<a href="#">CreateControlsInstance()</a>	<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>
<a href="#">CreateObjRef(Type)</a>	<a href="#">Cut()</a>	<a href="#">DefWndProc(Message)</a>
<a href="#">DeselectAll()</a>	<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>
<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>
<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>	<a href="#">FindForm()</a>
<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>
<a href="#">GetCharFromPosition(Point)</a>	<a href="#">GetCharIndexFromPosition(Point)</a>	<a href="#">GetChildAtPoint(Point)</a>
<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>	<a href="#">GetFirstCharIndexFromLine(Int32)</a>
<a href="#">GetFirstCharIndexOfCurrentLine()</a>	<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>
<a href="#">GetLineFromCharIndex(Int32)</a>	<a href="#">GetNextControl(Control, Boolean)</a>	<a href="#">GetPositionFromCharIndex(Int32)</a>
<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>
<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>
<a href="#">Hide()</a>	<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>
<a href="#">Invalidate()</a>	<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>
<a href="#">Invalidate(Rectangle, Boolean)</a>	<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>
<a href="#">Invoke(Delegate)</a>	<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>
<a href="#">InvokeLostFocus(Control, EventArgs)</a>	<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>

<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>	<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>
<a href="#">LogicalToDeviceUnits(Int32)</a>	<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>

<a href="#">MemberwiseClone(Boolean)</a>	<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAcceptsTabChanged(EventArgs)</a>
<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>	<a href="#">OnBackgroundImageChanged(EventArgs)</a>
<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>	<a href="#">OnBorderStyleChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>
<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>
<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>
<a href="#">OnDpiChangedAfterParent(EventArgs)</a>	<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>
<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>
<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>
<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(HelpEventArgs)</a>
<a href="#">OnHideSelectionChanged(EventArgs)</a>	<a href="#">OnImeModeChanged(EventArgs)</a>	<a href="#">OnInvalidate(InvalidateEventArgs)</a>
<a href="#">OnKeyDown(KeyEventArgs)</a>	<a href="#">OnKeyPress(KeyEventArgs)</a>	<a href="#">OnKeyUp(KeyEventArgs)</a>
<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>
<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnModifiedChanged(EventArgs)</a>

<a href="#">OnMouseCaptureChanged(EventArgs)</a>	<a href="#">OnMouseClick(MouseEventArgs)</a>	<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>
<a href="#">OnMouseDown(MouseEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>
<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEventArgs)</a>	<a href="#">OnMouseUp(MouseEventArgs)</a>
<a href="#">OnMouseWheel(MouseEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>	<a href="#">OnMultilineChanged(EventArgs)</a>
<a href="#">OnNotifyMessage(Message)</a>	<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>
<a href="#">OnPaintBackground(PaintEventArgs)</a>	<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>
<a href="#">OnParentBindingContextChanged(EventArgs)</a>	<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>
<a href="#">OnParentEnabledChanged(EventArgs)</a>	<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>
<a href="#">OnParentRightToLeftChanged(EventArgs)</a>	<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>
<a href="#">OnPrint(PaintEventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnReadOnlyChanged(EventArgs)</a>
<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>
<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>
<a href="#">OnTabIndexChanged(EventArgs)</a>	<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextAlignChanged(EventArgs)</a>
<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>
<a href="#">OnVisibleChanged(EventArgs)</a>	<a href="#">Paste()</a>	<a href="#">Paste(String)</a>

<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>
<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>
<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>
<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>
<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>
<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>
<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle)</a>	<a href="#">Refresh()</a>
<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>
<a href="#">ResetCursor()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>
<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>
<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">ScrollToCaret()</a>	<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>
<a href="#">Select(Int32, Int32)</a>	<a href="#">SelectAll()</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>

<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>
<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>
<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>
<a href="#">ToString()</a>	<a href="#">Undo()</a>	<a href="#">Update()</a>
<a href="#">UpdateBounds()</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>
<a href="#">UpdateStyles()</a>	<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>

U tabeli 21 su dati svi događaji za kontrolu *TextBox*.

**Tabela 21.** Tabela događaja za kontrolu *TextBox* [7]

<a href="#">AcceptsTabChanged</a>	<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>
<a href="#">BackgroundImageChanged</a>	<a href="#">BackgroundImageLayoutChange d</a>	<a href="#">BindingContextChanged</a>
<a href="#">BorderStyleChanged</a>	<a href="#">CausesValidationChanged</a>	<a href="#">ChangeUICTues</a>
<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>	<a href="#">ContextMenuChanged</a>
<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>	<a href="#">ControlRemoved</a>
<a href="#">CursorChanged</a>	<a href="#">Disposed</a>	<a href="#">DockChanged</a>
<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>	<a href="#">DpiChangedBeforeParent</a>
<a href="#">DragDrop</a>	<a href="#">DragEnter</a>	<a href="#">DragLeave</a>
<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>	<a href="#">Enter</a>
<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>	<a href="#">GiveFeedback</a>
<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>	<a href="#">HandleDestroyed</a>
<a href="#">HelpRequested</a>	<a href="#">HideSelectionChanged</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidate</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>

<a href="#">ModifiedChanged</a>	<a href="#">MouseCaptureChanged</a>	<a href="#">MouseClick</a>
<a href="#">MouseDoubleClick</a>	<a href="#">MouseDown</a>	<a href="#">MouseEnter</a>
<a href="#">MouseHover</a>	<a href="#">MouseLeave</a>	<a href="#">MouseMove</a>
<a href="#">MouseUp</a>	<a href="#">MouseWheel</a>	<a href="#">Move</a>
<a href="#">MultilineChanged</a>	<a href="#">PaddingChanged</a>	<a href="#">Paint</a>
<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>	<a href="#">QueryAccessibilityHelp</a>
<a href="#">QueryContinueDrag</a>	<a href="#">ReadOnlyChanged</a>	<a href="#">RegionChanged</a>
<a href="#">Resize</a>	<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#"> TextAlignChanged</a>	<a href="#">TextChanged</a>
<a href="#">Validated</a>	<a href="#">Validating</a>	<a href="#">VisibleChanged</a>

#### 6.9.6. Kontrola RichTextBox

Pomoću [RichTextBox](#) (kontrola broj 12 sa slike 26) kontrole korisnik može unositi i uređivati tekst. Kontrola takođe nudi naprednije funkcije formatiranja od standardne *TextBox* kontrole. Tekst se može dodeliti direktno kontroli ili može biti učitan iz obogaćenog tekstualnog formata (RTF) ili obične tekstualne datoteke. Tekstu unutar kontrole može se dodeliti oblik karaktera.

Kontrola *RichTextBox* pruža niz svojstava koja se koriste za editovanje bilo kojeg dela teksta u kontroli. Da bi se promenilo formatiranje teksta, prvo ga treba izabrati. Samo odabranom tekstu mogu se dodeliti formatiranje znakova i paragrafa. Jednom kada se izvrši podešavanje za izabrani odeljak teksta ceo tekst unesen nakon odabira se takođe formatira sa istim podešavanjima sve dok se ne promeni podešavanje ili ne izabere drugi odeljak dokumenta kontrole. Svojstvo *SelectionFont* omogućava da se tekst učita podebljanim ili kosim slovima. Takođe, može da se koristi ova osobina i za promenu veličine i slova teksta. Svojstvo *SelectionColor* omogućava promenu boje teksta. Da biste kreirali spiskove sa obrisima, možete koristiti svojstvo *SelectionBullet*. Formatiranje odlomaka možete prilagoditi i postavljanjem svojstava *SelectionIndent*, *SelectionRightIndent* i *SelectionHangingIndent*.

Kontrola *RichTextBox* pruža metode koje imaju funkcionalnost za otvaranje i čuvanje datoteka. Metoda *LoadFile* omogućava da se iz postojeće RTF ili ASCII tekstualne datoteke učita tekst u kontrolu. *SaveFile* metoda omogućava da se datoteka sačuva u RTF ili ASCII formatu. Kontrola *RichTextBox* takođe nudi funkcije za pronalaženje delova teksta, a metoda koja se za te potrebe koristi je metoda *Find*.

Ako tekst unutar kontrole sadrži link do neke veb-lokacije, može se koristiti svojstvo *DetectUrls* da bi se prikazao link u tekstu kontroli. Može da se koristi događaj *LinkClicked* da bi se izvršilo povezivanje sa datim linkom. Svojstvo *SelectionProtected* omogućava da se zaštitи tekst unutar kontrole od manipulacije od strane korisnika.

Aplikacija koja već koristi *TextBox* kontrolu može lako da se prilagoditi i da se koristi *RichTextBox* kontrola. Međutim, kontrola *RichTextBox* nema istu granicu od 64K karaktera kao *TextBox* kontrole. *RichTextBox* se obično koristi za manipulaciju (editovanje) tekstrom i slična je aplikacijama za obradu teksta kao što je *Microsoft Word*.

U tabeli 22 su data sva svojstva za kontrolu *RichTextBox*.

**Tabela 22.** Tabela svojstava za kontrolu *RichTextBox* [7]

<a href="#">AcceptsTab</a>	<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultActionDescription</a>	<a href="#">AccessibleDescription</a>
<a href="#">AccessibleName</a>	<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>
<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>	<a href="#">AutoWordSelection</a>	<a href="#">BackColor</a>
<a href="#">BackgroundImage</a>	<a href="#">BindingContext</a>	<a href="#">BackgroundImageLayout</a>	<a href="#">BorderStyle</a>
<a href="#">Bottom</a>	<a href="#">Bounds</a>	<a href="#">BulletIndent</a>	<a href="#">CanEnableIme</a>
<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>	<a href="#">CanRedo</a>	<a href="#">CanSelect</a>
<a href="#">CanUndo</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>	<a href="#">ClientRectangle</a>
<a href="#">ClientSize</a>	<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>
<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>
<a href="#">CreateParams</a>	<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>
<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>
<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DetectUrls</a>

<a href="#">DeviceDpi</a>	<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">Dock</a>
<a href="#">DoubleBuffered</a>	<a href="#">EnableAutoDragDrop</a>	<a href="#">Enabled</a>	<a href="#">Events</a>
<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>
<a href="#">Handle</a>	<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">HideSelection</a>
<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>
<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>	<a href="#">LanguageOption</a>
<a href="#">LayoutEngine</a>	<a href="#">Left</a>	<a href="#">Lines</a>	<a href="#">Location</a>
<a href="#">Margin</a>	<a href="#">MaximumSize</a>	<a href="#">MaxLength</a>	<a href="#">MinimumSize</a>
<a href="#">Modified</a>	<a href="#">Multiline</a>	<a href="#">Name</a>	<a href="#">Padding</a>
<a href="#">Parent</a>	<a href="#">PreferredSize</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>
<a href="#">ProductVersion</a>	<a href="#">ReadOnly</a>	<a href="#">RecreatingHandle</a>	<a href="#">RedoActionName</a>
<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">RichTextShortcutsEnabled</a>	<a href="#">ResizeRedraw</a>
<a href="#">Right</a>	<a href="#">RightMargin</a>	<a href="#">RightToLeft</a>	<a href="#">Rtf</a>
<a href="#">ScaleChildren</a>	<a href="#">ScrollBars</a>	<a href="#">SelectedRtf</a>	<a href="#">SelectedText</a>
<a href="#">SelectionAlignment</a>	<a href="#">SelectionBackColor</a>	<a href="#">SelectionCharOffset</a>	<a href="#">SelectionBullet</a>
<a href="#">SelectionColor</a>	<a href="#">SelectionFont</a>	<a href="#">SelectionHangingIndent</a>	<a href="#">SelectionIndent</a>
<a href="#">SelectionLength</a>	<a href="#">SelectionProtected</a>	<a href="#">SelectionRightIndent</a>	<a href="#">SelectionStart</a>
<a href="#">SelectionTabs</a>	<a href="#">SelectionType</a>	<a href="#">ShortcutsEnabled</a>	<a href="#">ShowFocusCues</a>
<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>	<a href="#">ShowSelectionMargin</a>	<a href="#">Size</a>
<a href="#">TabIndex</a>	<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>
<a href="#">TextLength</a>	<a href="#">Top</a>	<a href="#">TopLevelControl</a>	<a href="#">UndoActionName</a>
<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>	<a href="#">Width</a>	<a href="#">WindowTarget</a>
<a href="#">WordWrap</a>	<a href="#">ZoomFactor</a>		

U tabeli 23 su date sve metode za kontrolu *RichTextBox*.

**Tabela 23.** Tabela metoda za kontrolu *RichTextBox* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">AppendText(String)</a>
<a href="#">BeginInvoke(Delegate)</a>	<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>

<a href="#">CanPaste(DataFormats+Format)</a>	<a href="#">Clear()</a>	<a href="#">ClearUndo()</a>
<a href="#">Contains(Control)</a>	<a href="#">CreateAccessibilityInstance()</a>	<a href="#">Copy()</a>
<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>
<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>	<a href="#">CreateObjRef(Type)</a>
<a href="#">CreateRichEditOleCallback()</a>	<a href="#">Cut()</a>	<a href="#">DefWndProc(Message)</a>
<a href="#">DeselectAll()</a>	<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>
<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">Dispose(Boolean)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>
<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>	<a href="#">Find(Char[])</a>
<a href="#">Find(Char[], Int32)</a>	<a href="#">Find(Char[], Int32, Int32)</a>	<a href="#">Find(String)</a>
<a href="#">Find(String, Int32, Int32, RichTextBoxFinds)</a>	<a href="#">Find(String, Int32, RichTextBoxFinds)</a>	<a href="#">Find(String, RichTextBoxFinds)</a>
<a href="#">FindForm()</a>	<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>
<a href="#">GetAutoSizeMode()</a>	<a href="#">GetCharFromPosition(Point)</a>	<a href="#">GetCharIndexFromPosition(Point)</a>
<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>

<a href="#">GetFirstCharIndexFromLine(Int32)</a>	<a href="#">GetFirstCharIndexOfCurrentLine()</a>	<a href="#">GetHashCode()</a>
<a href="#">GetLifetimeService()</a>	<a href="#">GetLineFromCharIndex(Int32)</a>	<a href="#">GetNextControl(Control, Boolean)</a>
<a href="#">GetPositionFromCharIndex(Int32)</a>	<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>
<a href="#">GetService(Type)</a>	<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>
<a href="#">GetType()</a>	<a href="#">Hide()</a>	<a href="#">InitializeLifetimeService()</a>
<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>	<a href="#">Invalidate(Boolean)</a>
<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>	<a href="#">Invalidate(Region)</a>

<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>	<a href="#">Invoke(Delegate, Object[])</a>
<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>	<a href="#">InvokeOnClick(Control, EventArgs)</a>
<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>	<a href="#">IsInputChar(Char)</a>
<a href="#">IsInputKey(Keys)</a>	<a href="#">LoadFile(Stream, RichTextBoxStreamType)</a>	<a href="#">LoadFile(String)</a>
<a href="#">LoadFile(String, RichTextBoxStreamType)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>	<a href="#">LogicalToDeviceUnits(Size)</a>
<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>	<a href="#">NotifyInvalidate(Rectangle)</a>
<a href="#">OnAcceptsTabChanged(EventArgs)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>
<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnBorderStyleChanged(EventArgs)</a>	<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>
<a href="#">OnClick(EventArgs)</a>	<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContentsResized(ContentsResizedEventArgs)</a>
<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>	<a href="#">OnControlAdded(ControlEventArgs)</a>
<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>	<a href="#">OnCursorChanged(EventArgs)</a>

<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>	<a href="#">OnDpiChangedAfterParent(EventArgs)</a>
<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>	<a href="#">OnDragEnter(DragEventArgs)</a>
<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>	<a href="#">OnEnabledChanged(EventArgs)</a>
<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>	<a href="#">OnForeColorChanged(EventArgs)</a>

<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>	<a href="#">OnHandleCreated(EventArgs)</a>
<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(HelperEventArgs)</a>	<a href="#">OnHideSelectionChanged(EventArgs)</a>
<a href="#">OnHScroll(EventArgs)</a>	<a href="#">OnImeChange(EventArgs)</a>	<a href="#">OnImeModeChanged(EventArgs)</a>
<a href="#">OnInvalidate(InvalidateEventArgs)</a>	<a href="#">OnKeyDown(KeyEventEventArgs)</a>	<a href="#">OnKeyPress(KeyPressEventEventArgs)</a>
<a href="#">OnKeyUp(KeyEventEventArgs)</a>	<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>
<a href="#">OnLinkClicked(LinkClickedEventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>	<a href="#">OnLostFocus(EventArgs)</a>
<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnModifiedChanged(EventArgs)</a>	<a href="#">OnMouseCaptureChanged(EventArgs)</a>
<a href="#">OnMouseClick(MouseEventEventArgs)</a>	<a href="#">OnMouseDoubleClick(MouseEventEventArgs)</a>	<a href="#">OnMouseDown(MouseEventEventArgs)</a>
<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>	<a href="#">OnMouseLeave(EventArgs)</a>
<a href="#">OnMouseMove(MouseEventEventArgs)</a>	<a href="#">OnMouseUp(MouseEventEventArgs)</a>	<a href="#">OnMouseWheel(MouseEventArgs)</a>
<a href="#">OnMove(EventArgs)</a>	<a href="#">OnMultilineChanged(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>
<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>
<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>
<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>

<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>
<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>

<a href="#">OnProtected(EventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnReadOnlyChanged(EventArgs)</a>
<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>
<a href="#">OnSelectionChanged(EventArgs)</a>	<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnStyleChanged(EventArgs)</a>
<a href="#">OnSystemColorsChanged(EventArgs)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>	<a href="#">OnTabStopChanged(EventArgs)</a>
<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>
<a href="#">OnVisibleChanged(EventArgs)</a>	<a href="#">OnVScroll(EventArgs)</a>	<a href="#">Paste()</a>
<a href="#">Paste(DataFormats+Format)</a>	<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>
<a href="#">PointToClient(Point)</a>	<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>
<a href="#">PreProcessMessage(Message)</a>	<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>
<a href="#">ProcessDialogKey(Keys)</a>	<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message)</a>
<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>
<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>
<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle)</a>
<a href="#">Redo()</a>	<a href="#">Refresh()</a>	<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>
<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>	<a href="#">ResetCursor()</a>
<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>	<a href="#">ResetImeMode()</a>
<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>	<a href="#">ResetText()</a>
<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>	<a href="#">RtlTranslateAlignment(ContentAlignment)</a>

<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>	<a href="#">RtlTranslateContent(ContentAlignment)</a>
<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>	<a href="#">SaveFile(Stream, RichTextBoxStreamType)</a>
<a href="#">SaveFile(String)</a>	<a href="#">SaveFile(String, RichTextBoxStreamType)</a>	<a href="#">Scale(Single)</a>
<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>	<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>
<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>	<a href="#">ScrollToCaret()</a>
<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>	<a href="#">Select(Int32, Int32)</a>
<a href="#">SelectAll()</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>	<a href="#">SendToBack()</a>
<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>
<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>	<a href="#">SetStyle(ControlStyles, Boolean)</a>
<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>	<a href="#">Show()</a>
<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>	<a href="#">ToString()</a>
<a href="#">Undo()</a>	<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>
<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>	

U tabeli 24 su dati svi događaji za kontrolu *RichTextBox*.

**Tabela 24.** Tabela događaja za kontrolu *RichTextBox* [7]

<a href="#">AcceptsTabChanged</a>	<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>
<a href="#">BackgroundImageChanged</a>	<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>

<a href="#">BorderStyleChanged</a>	<a href="#">CausesValidationChanged</a>	<a href="#">ChangeUI_Cues</a>
<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>	<a href="#">ContentsResized</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>
<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>
<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">HideSelectionChanged</a>
<a href="#">HScroll</a>	<a href="#">ImeChange</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidate</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LinkClicked</a>	<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>
<a href="#">MarginChanged</a>	<a href="#">ModifiedChanged</a>	<a href="#">MouseCaptureChanged</a>
<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>	<a href="#">MouseDown</a>
<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>	<a href="#">MouseLeave</a>
<a href="#">MouseMove</a>	<a href="#">MouseUp</a>	<a href="#">MouseWheel</a>
<a href="#">Move</a>	<a href="#">MultilineChanged</a>	<a href="#">PaddingChanged</a>
<a href="#">Paint</a>	<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>
<a href="#">Protected</a>	<a href="#">QueryAccessibilityHelp</a>	<a href="#">QueryContinueDrag</a>
<a href="#">ReadOnlyChanged</a>	<a href="#">RegionChanged</a>	<a href="#">Resize</a>
<a href="#">RightToLeftChanged</a>	<a href="#">SelectionChanged</a>	<a href="#">SizeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>	<a href="#">Validated</a>
<a href="#">Validating</a>	<a href="#">VisibleChanged</a>	<a href="#">VScroll</a>

### 6.9.7. Kontrola ListBox

Kontrola [ListBox](#) (kontrola broj 9 sa slike 26) omogućava da se korisniku prikaže lista stavki koje korisnik može odabrati klikom. Kontrola *ListBox* može pružiti pojedinačni ili višestruki izbor koristeći svojstvo *SelectionMode*. *ListBox* kontrola takođe pruža svojstvo *MultiColumn*, tj. svojstvo koje omogućava prikaz stavki u kolonama umesto ravne vertikalne liste stavki.

Windows obično obavlja zadatak crtanja stavki koje će se prikazati u ListBok-u. Može se koristiti svojstvo *DrawMode* i rukovati *MeasureItem* i *DrawItem* događajima, tako da se nadjača automatski crtež koji pruža Windows i da korisnik sam crta stavke. Mogu se koristiti kontrolne liste *ListBox* koje su nacrtali vlasnici za prikaz stavki, slika ili druge boje u boji promenljive za tekst svake stavke na listi. Svojstva *HorizontalExtent*, *GetItemHeight* i *GetItemRectangle* takođe pomažu u crtaju sopstvenih predmeta.

Pored funkcije za prikaz i odabir, *ListBox* nudi i funkcije koje omogućavaju da se efikasno dodaju stavke u *ListBox* i da se pronađe tekst u stavkama liste. Metode *BeginUpdate* i *EndUpdate* omogućavaju da se doda veliki broj stavki u *ListBox* bez da se kontrola preispituje svaki put kada se stavka doda u listu. Metode *FindString* i *FindStringExact* omogućavaju da se traži stavka na listi koja sadrži određeni niz pretraživanja.

Svojstva *Item*, *SelectedItems* i *SelectedIndices* omogućavaju pristup kolekcijama koje koristi *ListBox*. Sledeća tabela (tabela 25) prikazuje tri kolekcije koje *ListBox* koristi i njihovu upotrebu unutar kontrole.

**Tabela 25. Tabela klasa kolekcija za *ListBox* [7]**

Klase kolekcija	Koristi se u sklopu <i>ListBox</i> kontole
<a href="#">ListBox.ObjectCollection</a>	Sadrži sve stavke sadržane u listi <i>ListBox</i> .
<a href="#">ListBox.SelectedObjectCollection</a>	Sadrži kolekciju izabralih stavki koja je podskup stavki sadržanih u listi <i>ListBox</i> .
<a href="#">ListBox.SelectedIndexCollection</a>	Sadrži kolekciju odabralih indeksa, koja je podskup indeksa <i>ListBok.ObjectCollection</i> . Ovi indeksi određuju izabrane stavke.

Metoda dodavanja klase *ListBox.ObjectCollection* omogućava dodavanje stavki u *ListBox*.

Metoda *Add* može prihvati bilo koji objekt prilikom dodavanja člana u *ListBox*. Kada se objekat dodaje u *ListBox*, kontrola koristi tekst definisan u *ToString* metodi objekta, osim ako se u svojstvu

*DisplayMember* navodi ime člana unutar objekta. Za dodavanje stavki koristi se metoda *Add* iz klase *ListBox.ObjectCollection*, a može da se doda i stavke koristeći svojstvo *DataSource*, iz klase *ListControl*.

U tabeli 26 su data sva svojstva za kontrolu *ListBox*.

**Tabela 26.** Tabela svojstava za kontrolu *ListBox* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultAction</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
	<a href="#">Description</a>		
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">AllowSelection</a>	<a href="#">Anchor</a>
<a href="#">AutoScrollOffset</a>	<a href="#">AutoSize</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>
<a href="#">BackgroundImageLayout</a>	<a href="#">BindingContext</a>	<a href="#">BorderStyle</a>	<a href="#">Bottom</a>
<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>
<a href="#">CanSelect</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>	<a href="#">ClientRectangle</a>
<a href="#">ClientSize</a>	<a href="#">ColumnWidth</a>	<a href="#">CompanyName</a>	<a href="#">Container</a>
<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>
<a href="#">Created</a>	<a href="#">CreateParams</a>	<a href="#">Cursor</a>	<a href="#">CustomTabOffsets</a>
<a href="#">DataBindings</a>	<a href="#">DataManager</a>	<a href="#">DataSource</a>	<a href="#">DefaultCursor</a>
<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>
<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>
<a href="#">DisplayMember</a>	<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">Dock</a>
<a href="#">DoubleBuffered</a>	<a href="#">DrawMode</a>	<a href="#">Enabled</a>	<a href="#">Events</a>
<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>
<a href="#">FormatInfo</a>	<a href="#">FormatString</a>	<a href="#">FormattingEnabled</a>	<a href="#">Handle</a>
<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">HorizontalExtent</a>	<a href="#">HorizontalScrollbar</a>
<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">IntegralHeight</a>	<a href="#">InvokeRequired</a>
<a href="#">IsAccessible</a>	<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>
<a href="#">ItemHeight</a>	<a href="#">Items</a>	<a href="#">LayoutEngine</a>	<a href="#">Left</a>
<a href="#">Location</a>	<a href="#">Margin</a>	<a href="#">MaximumSize</a>	<a href="#">MinimumSize</a>
<a href="#">MultiColumn</a>	<a href="#">Name</a>	<a href="#">Padding</a>	<a href="#">Parent</a>

<a href="#">PreferredHeight</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>
<a href="#">RecreatingHandle</a>	<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>
<a href="#">Right</a>	<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ScrollAlwaysVisible</a>
<a href="#">SelectedIndex</a>	<a href="#">SelectedIndices</a>	<a href="#">SelectedItem</a>	<a href="#">SelectedItems</a>
<a href="#">SelectedValue</a>	<a href="#">SelectionMode</a>	<a href="#">ShowFocusCues</a>	<a href="#">ShowKeyboardCues</a>
<a href="#">Site</a>	<a href="#">Size</a>	<a href="#">Sorted</a>	<a href="#">TabIndex</a>
<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#">Top</a>
<a href="#">TabIndex</a>	<a href="#">TopLevelControl</a>	<a href="#">UseCustomTabOffsets</a>	<a href="#">UseTabStops</a>
<a href="#">UseWaitCursor</a>	<a href="#">ValueMember</a>	<a href="#">Visible</a>	<a href="#">Width</a>
<a href="#">WindowTarget</a>			

U tabeli 27 su date sve metode za kontrolu *ListBox*.

**Tabela 27.** Tabela metoda za kontrolu *ListBox* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">AddItemsCore(Object[])</a>
<a href="#">BeginInvoke(Delegate)</a>	<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BeginUpdate()</a>
<a href="#">BringToFront()</a>	<a href="#">ClearSelected()</a>	<a href="#">Contains(Control)</a>
<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>
<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>	<a href="#">CreateItemCollection()</a>
<a href="#">CreateObjRef(Type)</a>	<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>
<a href="#">Dispose()</a>	<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>
<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>	<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">EndUpdate()</a>
<a href="#">Equals(Object)</a>	<a href="#">FilterItemOnProperty(Object)</a>	<a href="#">FilterItemOnProperty(Object, String)</a>
<a href="#">FindForm()</a>	<a href="#">FindString(String)</a>	<a href="#">FindString(String, Int32)</a>
<a href="#">FindStringExact(String)</a>	<a href="#">FindStringExact(String, Int32)</a>	<a href="#">Focus()</a>
<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>	<a href="#">GetChildAtPoint(Point)</a>

<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>	<a href="#">GetHashCode()</a>
<a href="#">GetItemHeight(Int32)</a>	<a href="#">GetItemRectangle(Int32)</a>	<a href="#">GetItemText(Object)</a>
<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a>	<a href="#">GetPreferredSize(Size)</a>
<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetSelected(Int32)</a>	<a href="#">GetService(Type)</a>
<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>
<a href="#">Hide()</a>	<a href="#">IndexFromPoint(Int32, Int32)</a>	<a href="#">IndexFromPoint(Point)</a>
<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>

<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>
<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>
<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>
<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>
<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>
<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>
<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>

<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDataSourceChanged(Event Args)</a>	<a href="#">OnDisplayMemberChanged(E ventArgs)</a>
<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>	<a href="#">OnDpiChangedAfterParent(E ventArgs)</a>
<a href="#">OnDpiChangedBeforeParent(Even tArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>	<a href="#">OnDragEnter(DragEventArgs)</a>
<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>	<a href="#">OnDrawItem(DrawItemEvent Args)</a>
<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnFormat(ListControlConvert EventArgs)</a>	<a href="#">OnFormatInfoChanged(Event Args)</a>
<a href="#">OnFormatStringChanged(EventAr ghs)</a>	<a href="#">OnFormattingEnabledChange d(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedba ckEventArgs)</a>
<a href="#">OnGotFocus(EventArgs)</a>	<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventAr gs)</a>

<a href="#">OnHelpRequested(HelperEventArgs)</a>	<a href="#">OnImeModeChanged(EventA rgs)</a>	<a href="#">OnInvalidate(InvalidateEven tArgs)</a>
<a href="#">OnKeyDown(KeyEventEventArgs)</a>	<a href="#">OnKeyPress(KeyPressEventAr gs)</a>	<a href="#">OnKeyUp(KeyEventArgs)</a>
<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventAr gs)</a>
<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArg s)</a>	<a href="#">OnMeasureItem(MeasureItemEventArgs)</a>
<a href="#">OnMouseCaptureChanged(EventA rgs)</a>	<a href="#">OnMouseClick(MouseEventAr gs)</a>	<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>
<a href="#">OnMouseDown(MouseEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>
<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEvent Args)</a>	<a href="#">OnMouseUp(MouseEventArg s)</a>
<a href="#">OnMouseWheel(MouseEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>

<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>
<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>
<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>
<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>
<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>
<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>
<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>
<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnValidated(EventArgs)</a>
<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnValueMemberChanged(EventArgs)</a>	<a href="#">OnVisibleChanged(EventArgs)</a>
<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>

<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>
<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>
<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyEventArgs(Message)</a>
<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>
<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>
<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>
<a href="#">RectangleToScreen(Rectangle)</a>	<a href="#">Refresh()</a>	<a href="#">RefreshItem(Int32)</a>

<a href="#">RefreshItems()</a>	<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>
<a href="#">ResetBindings()</a>	<a href="#">ResetCursor()</a>	<a href="#">ResetFont()</a>
<a href="#">ResetForeColor()</a>	<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>
<a href="#">ResetRightToLeft()</a>	<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>
<a href="#">ResumeLayout(Boolean)</a>	<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>
<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>	<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>
<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>	<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>
<a href="#">Scale(SizeF)</a>	<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>
<a href="#">ScaleCore(Single, Single)</a>	<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>
<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>	<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>
<a href="#">SetClientSizeCore(Int32, Int32)</a>	<a href="#">SetItemCore(Int32, Object)</a>	<a href="#">SetItemsCore(IList)</a>
<a href="#">SetSelected(Int32, Boolean)</a>	<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>
<a href="#">SetVisibleCore(Boolean)</a>	<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>
<a href="#">Sort()</a>	<a href="#">SuspendLayout()</a>	<a href="#">ToString()</a>
<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>	<a href="#">UpdateZOrder()</a>
<a href="#">WmReflectCommand(Message)</a>	<a href="#">WndProc(Message)</a>	

U tabeli 28 su dati svi događaji za kontrolu *ListBox*.

**Tabela 28.** Tabela događaja za kontrolu *ListBox* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">DataSourceChanged</a>
<a href="#">DisplayMemberChanged</a>	<a href="#">Disposed</a>	<a href="#">DockChanged</a>
<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>	<a href="#">DpiChangedBeforeParent</a>
<a href="#">DragDrop</a>	<a href="#">DragEnter</a>	<a href="#">DragLeave</a>
<a href="#">DragOver</a>	<a href="#">DrawItem</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">Format</a>	<a href="#">FormatInfoChanged</a>	<a href="#">FormatStringChanged</a>
<a href="#">FormattingEnabledChanged</a>	<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>
<a href="#">HandleCreated</a>	<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>
<a href="#">ImeModeChanged</a>	<a href="#">Invalidated</a>	<a href="#">KeyDown</a>
<a href="#">KeyPress</a>	<a href="#">KeyUp</a>	<a href="#">Layout</a>
<a href="#">Leave</a>	<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>
<a href="#">MarginChanged</a>	<a href="#">MeasureItem</a>	<a href="#">MouseCaptureChanged</a>
<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>	<a href="#">MouseDown</a>
<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>	<a href="#">MouseLeave</a>
<a href="#">MouseMove</a>	<a href="#">MouseUp</a>	<a href="#">MouseWheel</a>
<a href="#">Move</a>	<a href="#">PaddingChanged</a>	<a href="#">Paint</a>
<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>	<a href="#">QueryAccessibilityHelp</a>
<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>	<a href="#">Resize</a>
<a href="#">RightToLeftChanged</a>	<a href="#">SelectedIndexChanged</a>	<a href="#">SelectedValueChanged</a>
<a href="#">SizeChanged</a>	<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>
<a href="#">TabIndexChanged</a>	<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>
<a href="#">Validated</a>	<a href="#">Validating</a>	<a href="#">ValueMemberChanged</a>
<a href="#">VisibleChanged</a>		

## 6.9.8. Kontrola ComboBox

[ComboBox](#) je standardna grafička kontrola koju ste sigurno već sreli (kontrola broj 10 sa slike 26). *ComboBox* prikazuje tekstualni okvir u kombinaciji sa *ListBox* kontrolom, koji korisniku omogućava da odabere stavke sa liste ili da unese novu vrednost.

Svojstvo *DropDownStyle* određuje da li se lista uvek prikazuje ili je lista prikazana u padajućem meniju. Svojstvo *DropDownStyle* takođe određuje da li se deo teksta može uređivati. Svojstvo *ComboBoxStyle* se koristi za dostupne postavke i njihove efekte. Ne postoji podešavanje za stalno prikazivanje liste i zabrane unošenja nove vrednosti. Da bi se prikazala lista kojoj se ne mogu dodati nove vrednosti, koristiće se kontrola *ListBox*.

Da bi se dodali ili uklonili objekte na listi, koriste se metode klase *ComboBox.ObjectCollection* (kroz svojstvo *Items ComboBox*). Pomoću metode *AddRange* može se dodeliti niz referenci objekta. Nakon toga lista prikazuje zadatu vrednost niza za svaki objekt. Pojedini objekti mogu se dodati metodom *Add*. Može da se obriše stavka iz kontole pomoću metodom *Remove* ili da se obrišu čitavi liste metodom *Clear*.

Pored funkcionalnosti prikaza i odabira, *ComboBox* nudi i funkcije koje omogućavaju da efikasno dodate stavke u *ComboBox* i da pronađete tekst unutar stavki liste. Pomoću metoda *BeginUpdate* i *EndUpdate* možete dodati veliki broj stavki u *ComboBox* bez da se kontrola preispituje svaki put kada se stavka doda na listu. Metode *FindString* i *FindStringExact* omogućavaju da se traži stavka na listi koja sadrži određeni niz pretraživanja.

Mogu da se koriste sledeća svojstva za upravljanje trenutno odabranom stavkom na listi, svojstvo *Text* da se odredi niz prikazan u polju za uređivanje, svojstvo *SelectedIndex* da bi se dobila ili postavila trenutna stavka, a svojstvo *SelectedItem* da bi se dobila ili postavila referenca na objekat.

U tabeli 29 su data sva svojstva za kontrolu *ComboBox*.

**Tabela 29.** Tabela svojstava za kontrolu *ComboBox* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultAction</a> <a href="#">Description</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">AllowSelection</a>	<a href="#">Anchor</a>
<a href="#">AutoCompleteCustomSource</a>	<a href="#">AutoCompleteMode</a>	<a href="#">AutoCompleteSource</a>	<a href="#">AutoScrollOffset</a>
<a href="#">AutoSize</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>	<a href="#">BackgroundImageLayout</a>

<a href="#">BindingContext</a>	<a href="#">Bottom</a>	<a href="#">Bounds</a>	<a href="#">CanEnableme</a>
<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>	<a href="#">CanSelect</a>	<a href="#">Capture</a>
<a href="#">CausesValidation</a>	<a href="#">ClientRectangle</a>	<a href="#">ClientSize</a>	<a href="#">CompanyName</a>
<a href="#">Container</a>	<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>
<a href="#">Controls</a>	<a href="#">Created</a>	<a href="#">CreateParams</a>	<a href="#">Cursor</a>
<a href="#">DataBindings</a>	<a href="#">DataManager</a>	<a href="#">DataSource</a>	<a href="#">DefaultCursor</a>
<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>
<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>
<a href="#">DisplayMember</a>	<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">Dock</a>
<a href="#">DoubleBuffered</a>	<a href="#">DrawMode</a>	<a href="#">DropDownHeight</a>	<a href="#">DropDownStyle</a>
<a href="#">DropDownWidth</a>	<a href="#">DroppedDown</a>	<a href="#">Enabled</a>	<a href="#">Events</a>
<a href="#">FlatStyle</a>	<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>
<a href="#">ForeColor</a>	<a href="#">FormatInfo</a>	<a href="#">FormatString</a>	<a href="#">FormattingEnabled</a>
<a href="#">Handle</a>	<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">ImeMode</a>
<a href="#">ImeModeBase</a>	<a href="#">IntegralHeight</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>
<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>	<a href="#">ItemHeight</a>
<a href="#">Items</a>	<a href="#">LayoutEngine</a>	<a href="#">Left</a>	<a href="#">Location</a>
<a href="#">Margin</a>	<a href="#">MaxDropDownItems</a>	<a href="#">MaximumSize</a>	<a href="#">MaxLength</a>
<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>	<a href="#">Parent</a>
<a href="#">PreferredSize</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>
<a href="#">RecreatingHandle</a>	<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>
<a href="#">Right</a>	<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">SelectedIndex</a>
<a href="#">SelectedItem</a>	<a href="#">SelectedText</a>	<a href="#">SelectedValue</a>	<a href="#">SelectionLength</a>
<a href="#">SelectionStart</a>	<a href="#">ShowFocusCues</a>	<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>
<a href="#">Size</a>	<a href="#">Sorted</a>	<a href="#">TabIndex</a>	<a href="#">TabStop</a>
<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#">Top</a>	<a href="#">TopLevelControl</a>
<a href="#">UseWaitCursor</a>	<a href="#">ValueMember</a>	<a href="#">Visible</a>	<a href="#">Width</a>
<a href="#">WindowTarget</a>			

U tabeli 30 su date sve metode za kontrolu *ComboBox*.

**Tabela 30.** Tabela metoda za kontrolu ComboBox [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">AddItemsCore(Object[])</a>
<a href="#">BeginInvoke(Delegate)</a>	<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BeginUpdate()</a>
<a href="#">BringToFront()</a>	<a href="#">Contains(Control)</a>	<a href="#">CreateAccessibilityInstance()</a>
<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>	<a href="#">CreateGraphics()</a>
<a href="#">CreateHandle()</a>	<a href="#">CreateObjRef(Type)</a>	<a href="#">DefWndProc(Message)</a>
<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>	<a href="#">Dispose(Boolean)</a>
<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>	<a href="#">EndInvoke(IAsyncResult)</a>
<a href="#">EndUpdate()</a>	<a href="#">Equals(Object)</a>	<a href="#">FilterItemOnProperty(Object)</a>
<a href="#">FilterItemOnProperty(Object, String)</a>	<a href="#">FindForm()</a>	<a href="#">FindString(String)</a>
<a href="#">FindString(String, Int32)</a>	<a href="#">FindStringExact(String)</a>	<a href="#">FindStringExact(String, Int32)</a>
<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>
<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>
<a href="#">GetHashCode()</a>	<a href="#">GetItemHeight(Int32)</a>	<a href="#">GetItemText(Object)</a>
<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a>	<a href="#">GetPreferredSize(Size)</a>
<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>	<a href="#">GetStyle(ControlStyles)</a>
<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>	<a href="#">Hide()</a>
<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>

<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>
<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>

<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>
<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>
<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>
<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>
<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>
<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDataSourceChanged(EventArgs)</a>	<a href="#">OnDisplayMemberChanged(EventArgs)</a>
<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>	<a href="#">OnDpiChangedAfterParent(EventArgs)</a>
<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>	<a href="#">OnDragEnter(DragEventArgs)</a>
<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>	<a href="#">OnDrawItem(DrawItemEventArgs)</a>
<a href="#">OnDropDown(EventArgs)</a>	<a href="#">OnDropDownClosed(EventArgs)</a>	<a href="#">OnDropDownStyleChanged(EventArgs)</a>
<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnFormat(ListControlConvertEventArgs)</a>	<a href="#">OnFormatInfoChanged(EventArgs)</a>

<a href="#">OnFormatStringChanged(EventArgs)</a>	<a href="#">OnFormattingEnabledChanged(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>
<a href="#">OnGotFocus(EventArgs)</a>	<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>
<a href="#">OnHelpRequested(HelpEventArgs)</a>	<a href="#">OnImeModeChanged(EventArgs)</a>	<a href="#">OnInvalidate(InvalidateEventArgs)</a>
<a href="#">OnKeyDown(KeyEventEventArgs)</a>	<a href="#">OnKeyPress(KeyPressEventArgs)</a>	<a href="#">OnKeyUp(KeyEventEventArgs)</a>
<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>
<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnMeasureItem(MeasureItemEventArgs)</a>
<a href="#">OnMouseCaptureChanged(EventArgs)</a>	<a href="#">OnMouseClick(MouseEventArgs)</a>	<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>
<a href="#">OnMouseDown(MouseEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>
<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEventArgs)</a>	<a href="#">OnMouseUp(MouseEventArgs)</a>
<a href="#">OnMouseWheel(MouseEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>
<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>
<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>
<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>
<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>
<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>

<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>
<a href="#">OnRightToLeftChanged(EventArgs)</a>	<a href="#">OnSelectedIndexChanged(EventArgs)</a>	<a href="#">OnSelectedItemChanged(EventArgs)</a>
<a href="#">OnSelectedValueChanged(EventArgs)</a>	<a href="#">OnSelectionChangeCommitted(EventArgs)</a>	<a href="#">OnSizeChanged(EventArgs)</a>

<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>
<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnTextUpdate(EventArgs)</a>
<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnValueMemberChanged(EventArgs)</a>
<a href="#">OnVisibleChanged(EventArgs)</a>	<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>
<a href="#">PointToClient(Point)</a>	<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>
<a href="#">PreProcessMessage(Message)</a>	<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>
<a href="#">ProcessDialogKey(Keys)</a>	<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message)</a>
<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>
<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>
<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle)</a>
<a href="#">Refresh()</a>	<a href="#">RefreshItem(Int32)</a>	<a href="#">RefreshItems()</a>
<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>
<a href="#">ResetCursor()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>

<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>
<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>	<a href="#">Select(Int32, Int32) SelectAll()</a>
<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>	<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>
<a href="#">SetClientSizeCore(Int32, Int32)</a>	<a href="#">SetItemCore(Int32, Object)</a>	<a href="#">SetItemsCore(IList)</a>
<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>
<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>
<a href="#">ToString()</a>	<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>
<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>	

U tabeli 31 su dati svi događaji za kontrolu *ComboBox*.

**Tabela 31.** Tabela događaja za kontrolu *ComboBox* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChange</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">DataSourceChanged</a>

<a href="#">DisplayMemberChanged</a>	<a href="#">Disposed</a>	<a href="#">DockChanged</a>
<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>	<a href="#">DpiChangedBeforeParent</a>
<a href="#">DragDrop</a>	<a href="#">DragEnter</a>	<a href="#">DragLeave</a>
<a href="#">DragOver</a>	<a href="#">DrawItem</a>	<a href="#">DropDown</a>
<a href="#">DropDownClosed</a>	<a href="#">DropDownStyleChanged</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">Format</a>	<a href="#">FormatInfoChanged</a>	<a href="#">FormatStringChanged</a>
<a href="#">FormattingEnabledChanged</a>	<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>
<a href="#">HandleCreated</a>	<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>
<a href="#">ImeModeChanged</a>	<a href="#">Invalidate</a>	<a href="#">KeyDown</a>
<a href="#">KeyPress</a>	<a href="#">KeyUp</a>	<a href="#">Layout</a>
<a href="#">Leave</a>	<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>
<a href="#">MarginChanged</a>	<a href="#">MeasureItem</a>	<a href="#">MouseCaptureChanged</a>
<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>	<a href="#">MouseDown</a>
<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>	<a href="#">MouseLeave</a>
<a href="#">MouseMove</a>	<a href="#">MouseUp</a>	<a href="#">MouseWheel</a>
<a href="#">Move</a>	<a href="#">PaddingChanged</a>	<a href="#">Paint</a>
<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>	<a href="#">QueryAccessibilityHelp</a>
<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>	<a href="#">Resize</a>
<a href="#">RightToLeftChanged</a>	<a href="#">SelectedIndexChanged</a>	<a href="#">SelectedValueChanged</a>
<a href="#">SelectionChangeCommitted</a>	<a href="#">SizeChanged</a>	<a href="#">StyleChanged</a>
<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>	<a href="#">TabStopChanged</a>
<a href="#">TextChanged</a>	<a href="#">TextUpdate</a>	<a href="#">Validated</a>
<a href="#">Validating</a>	<a href="#">ValueMemberChanged</a>	<a href="#">VisibleChanged</a>

### 6.9.9. Kontrola PictureBox

Obično se [PictureBox](#) (kontrola broj 4 sa slike 26) koristi za prikazivanje grafike iz bitmap, metafile, ikone, JPEG, GIF ili PNG datoteka.

Pomoću *PictureBox* kontrole sa svojstvom *Image* slika se može prikazati u vreme projektovanja ili u toku rada. Može alternativno da se odredi slika koja se postavlja svojstvom *ImageLocation* i sinhrono učitava slika metodom *Load* ili asinhrono korist metodu *LoadAsync*.

Svojstvo *SizeMode*, koje je postavljeno na vrednosti *PictureBoxSizeMode*, kontroliše isecanje i pozicioniranje slike u prikaznom području (tj. *PictureBox*-u). Može da se promeni veličina područja prikaza tokom vremena izvođenja pomoću svojstva *ClientSize*.

Podrazumevano se kontrola *PictureBox* prikazuje bez ikakvih okvira. Može se dati standardni ili trodimenzionalni okvir koristeći svojstvo *BorderStyle* da bi se razlikovao okvir za slike od ostatka forme, čak i ako on ne sadrži sliku. *PictureBox* nije kontrola koja se može birati, što znači da ne može da primi ulazni fokus.

U tabeli 32 su data sva svojstva za kontrolu *PictureBox*.

**Tabela 32.** Tabela svojstava za kontrolu *PictureBox* [7]

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultAction</a> <a href="#">Description</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">AutoScrollOffset</a>
<a href="#">AutoSize</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>	<a href="#">BackgroundImageLayout</a>
<a href="#">BindingContext</a>	<a href="#">BorderStyle</a>	<a href="#">Bottom</a>	<a href="#">Bounds</a>
<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>	<a href="#">CanSelect</a>
<a href="#">Capture</a>	<a href="#">CausesValidation</a>	<a href="#">ClientRectAngle</a>	<a href="#">ClientSize</a>
<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>
<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>	<a href="#">CreateParams</a>
<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>	<a href="#">DefaultImeMode</a>
<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultPadding</a>
<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>	<a href="#">DisplayRectangle</a>
<a href="#">disposing</a>	<a href="#">Dock</a>	<a href="#">DoubleBuffered</a>	<a href="#">Enabled</a>
<a href="#">ErrorImage</a>	<a href="#">Events</a>	<a href="#">Focused</a>	<a href="#">Font</a>

<a href="#">FontHeight</a>	<a href="#">ForeColor</a>	<a href="#">Handle</a>	<a href="#">HasChildren</a>
<a href="#">Height</a>	<a href="#">Image</a>	<a href="#">ImageLocation</a>	<a href="#">ImeMode</a>
<a href="#">ImeModeBase</a>	<a href="#">InitialImage</a>	<a href="#">InvokeRequired</a>	<a href="#">IsAccessible</a>
<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>	<a href="#">LayoutEngine</a>
<a href="#">Left</a>	<a href="#">Location</a>	<a href="#">Margin</a>	<a href="#">MaximumSize</a>
<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>	<a href="#">Parent</a>
<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>	<a href="#">RecreatingHandle</a>
<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>	<a href="#">Right</a>
<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ShowFocusCues</a>	<a href="#">ShowKeyboardCues</a>
<a href="#">Site</a>	<a href="#">Size</a>	<a href="#">SizeMode</a>	<a href="#">TabIndex</a>
<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#">Top</a>
<a href="#">TopLevelControl</a>	<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>	<a href="#">WaitOnLoad</a>
<a href="#">Width</a>	<a href="#">WindowTarget</a>		

U tabeli 33 su date sve metode za kontrolu *PictureBox*.

**Tabela 33.** Tabela metoda za kontrolu *PictureBox* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">BeginInvoke(Delegate)</a>
<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>	<a href="#">CancelAsync()</a>
<a href="#">Contains(Control)</a>	<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>
<a href="#">CreateControlsInstance()</a>	<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>
<a href="#">CreateObjRef(Type)</a>	<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>
<a href="#">Dispose()</a>	<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>
<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>	<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>
<a href="#">FindForm()</a>	<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>
<a href="#">GetAutoSizeMode()</a>	<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>

<a href="#">GetContainerControl()</a>	<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>
<a href="#">GetNextControl(Control, Boolean)</a>	<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>
<a href="#">GetService(Type)</a>	<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>
<a href="#">GetType()</a>	<a href="#">Hide()</a>	<a href="#">InitializeLifetimeService()</a>
<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>	<a href="#">Invalidate(Boolean)</a>
<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>	<a href="#">Invalidate(Region)</a>
<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>	<a href="#">Invoke(Delegate, Object[])</a>
<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>	<a href="#">InvokeOnClick(Control, EventArgs)</a>
<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>	<a href="#">IsInputChar(Char)</a>
<a href="#">IsInputKey(Keys)</a>	<a href="#">Load()</a>	<a href="#">Load(String)</a>
<a href="#">LoadAsync()</a>	<a href="#">LoadAsync(String)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>

<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>
<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>
<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>
<a href="#">OnDpiChangedAfterParent(EventArgs)</a>	<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>

<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>
<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>
<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(HelpEventArgs)</a>
<a href="#">OnImeModeChanged(EventArgs)</a>	<a href="#">OnInvalidate(InvalidateEventArgs)</a>	<a href="#">OnKeyDown(KeyEventEventArgs)</a>
<a href="#">OnKeyPress(KeyEventEventArgs)</a>	<a href="#">OnKeyUp(KeyEventEventArgs)</a>	<a href="#">OnLayout(LayoutEventArgs)</a>
<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLoadCompleted(AsyncCompletedEventArgs)</a>	<a href="#">OnLoadProgressChanged(ProgressChangedEventArgs)</a>
<a href="#">OnLocationChanged(EventArgs)</a>	<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArgs)</a>
<a href="#">OnMouseCaptureChanged(EventArgs)</a>	<a href="#">OnMouseClick(MouseEventArgs)</a>	<a href="#">OnMouseDoubleClick(MouseEventArgs)</a>
<a href="#">OnMouseDown(MouseEventEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>
<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEventArgs)</a>	<a href="#">OnMouseUp(MouseEventArgs)</a>
<a href="#">OnMouseWheel(MouseEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>
<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>

<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>
<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>
<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>
<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>

<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a> }	<a href="#">OnResize(EventArgs)</a>
<a href="#">OnRightToLeftChanged(EventArgs)</a> }	<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnSizeModeChanged(EventArgs)</a>
<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(Even Args)</a>	<a href="#">OnTabIndexChanged(EventArgs)</a>
<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextChanged(EventArgs)</a>	<a href="#">OnValidated(EventArgs)</a>
<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnVisibleChanged(EventArgs)</a> }	<a href="#">PerformLayout()</a>
<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>	<a href="#">PointToScreen(Point)</a>
<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>	<a href="#">ProcessCmdKey(Message, Keys)</a>
<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>	<a href="#">ProcessKeyEventArgs(Message)</a>
<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>	<a href="#">ProcessMnemonic(Char)</a>
<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>	<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>
<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>	<a href="#">RectangleToClient(Rectangle)</a>
<a href="#">RectangleToScreen(Rectangle)</a>	<a href="#">Refresh()</a>	<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>
<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>	<a href="#">ResetCursor()</a>
<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>	<a href="#">ResetImeMode()</a>
<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>	<a href="#">ResetText()</a>
<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>	<a href="#">RtlTranslateAlignment(ContentAlignment)</a>
<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>	<a href="#">RtlTranslateContent(Content Alignment)</a>
<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>	<a href="#">Scale(Single)</a>

<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>	<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>
<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>	<a href="#">Select()</a>
<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>	<a href="#">SendToBack()</a>
<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>
<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>	<a href="#">SetStyle(ControlStyles, Boolean)</a>
<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>	<a href="#">Show()</a>
<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>	<a href="#">ToString()</a>
<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>	<a href="#">UpdateZOrder()</a>
<a href="#">WndProc(Message)</a>		

U tabeli 34 su dati svi događaji za kontrolu *PictureBox*.

**Tabela 34.** Tabela događaja za kontrolu *PictureBox* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>
<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>

<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>
<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidate</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LoadCompleted</a>	<a href="#">LoadProgressChanged</a>	<a href="#">LocationChanged</a>
<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>	<a href="#">MouseCaptureChanged</a>
<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>	<a href="#">MouseDown</a>
<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>	<a href="#">MouseLeave</a>
<a href="#">MouseMove</a>	<a href="#">MouseUp</a>	<a href="#">MouseWheel</a>
<a href="#">Move</a>	<a href="#">PaddingChanged</a>	<a href="#">Paint</a>
<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>	<a href="#">QueryAccessibilityHelp</a>
<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>	<a href="#">Resize</a>
<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>	<a href="#">SizeModeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>	<a href="#">Validated</a>
<a href="#">Validating</a>	<a href="#">VisibleChanged</a>	

#### 6.9.10. Kontrola GroupBox

[GroupBox](#) prikazuje okvir oko grupe kontrola sa ili bez natpisa (kontrola broj 6 sa slike 26). Kontrola *GroupBox* se koristi da logički grupiše kolekciju kontrola na formi. *GroupBox* je kontrola koja se koristi za grupisanje grupe kontrola.

Tipična upotreba za *GroupBox* je da sadrži logičku grupu *RadioButton* ili *CheckBox* kontrola.

Kontrole u *GroupBox* se mogu dodati korišćenjem metode *Add* iz klase *Controls*.

Ako je potrebna kontrola slična *GroupBox*-u, može se koristiti kontrola panel, koju ćemo obraditi sledeću u okviru ovog poglavlja.

U tabeli 35 su data sva svojstva za kontrolu *GroupBox*.

**Tabela 35. Tabela svojstava za kontrolu *GroupBox* [7]**

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultActionDescription</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">AutoScrollOffset</a>

<a href="#">AutoSize</a>	<a href="#">AutoSizeMode</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>
<a href="#">BackgroundImageLa yout</a>	<a href="#">BindingContext</a>	<a href="#">Bottom</a>	<a href="#">Bounds</a>
<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>	<a href="#">CanRaiseEvents</a>	<a href="#">CanSelect</a>
<a href="#">Capture</a>	<a href="#">CausesValidation</a>	<a href="#">ClientRectangle</a>	<a href="#">ClientSize</a>
<a href="#">CompanyName</a>	<a href="#">Container</a>	<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>
<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>	<a href="#">Created</a>	<a href="#">CreateParams</a>
<a href="#">Cursor</a>	<a href="#">DataBindings</a>	<a href="#">DefaultCursor</a>	<a href="#">DefaultImeMode</a>
<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>	<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultPadding</a>
<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>	<a href="#">DeviceDpi</a>	<a href="#">DisplayRectangle</a>
<a href="#">Disposing</a>	<a href="#">Dock</a>	<a href="#">DoubleBuffered</a>	<a href="#">Enabled</a>
<a href="#">Events</a>	<a href="#">FlatStyle</a>	<a href="#">Focused</a>	<a href="#">Font</a>
<a href="#">FontHeight</a>	<a href="#">ForeColor</a>	<a href="#">Handle</a>	<a href="#">HasChildren</a>
<a href="#">Height</a>	<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>
<a href="#">IsAccessible</a>	<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>
<a href="#">LayoutEngine</a>	<a href="#">Left</a>	<a href="#">Location</a>	<a href="#">Margin</a>
<a href="#">MaximumSize</a>	<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>
<a href="#">Parent</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>
<a href="#">RecreatingHandle</a>	<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>
<a href="#">Right</a>	<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ShowFocusCues</a>
<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>	<a href="#">Size</a>	<a href="#">TabIndex</a>
<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#">Top</a>
<a href="#">TopLevelControl</a>	<a href="#">UseCompatibleTextRende ring</a>	<a href="#">UseWaitCursor</a>	<a href="#">Visible</a>
<a href="#">Width</a>	<a href="#">WindowTarget</a>		

U tabeli 36 su date sve metode za kontrolu *GroupBox*.

**Tabela 36.** Tabela metoda za kontrolu *GroupBox* [7]

<a href="#">AccessibilityNotifyClients(Accessib leEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(Acc essibleEvents, Int32, Int32)</a>	<a href="#">BeginInvoke(Delegate)</a>
--------------------------------------------------------------------------	---------------------------------------------------------------------------------	---------------------------------------

<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>	<a href="#">Contains(Control)</a>
<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>	<a href="#">CreateControlsInstance()</a>
<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>	<a href="#">CreateObjRef(Type)</a>
<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>	<a href="#">Dispose()</a>
<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>	<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>
<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>	<a href="#">FindForm()</a>
<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>	<a href="#">GetAutoSizeMode()</a>
<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>	<a href="#">GetContainerControl()</a>
<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>	<a href="#">GetNextControl(Control, Boolean)</a>
<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>	<a href="#">GetService(Type)</a>
<a href="#">GetStyle(ControlStyles)</a>	<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>
<a href="#">Hide()</a>	<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>
<a href="#">Invalidate()</a>	<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>
<a href="#">Invalidate(Rectangle, Boolean)</a>	<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>
<a href="#">Invoke(Delegate)</a>	<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>
<a href="#">InvokeLostFocus(Control, EventArgs)</a>	<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>
<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>	<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>
<a href="#">LogicalToDeviceUnits(Int32)</a>	<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>
<a href="#">MemberwiseClone(Boolean)</a>	<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>

<a href="#">OnBackColorChanged(EventArgs)</a>	<a href="#">OnBackgroundImageChange</a> <a href="#">d(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutC</a> <a href="#">hanged(EventArgs)</a>
<a href="#">OnBindingContextChanged(Event</a> <a href="#">Args)</a>	<a href="#">OnCausesValidationChanged(</a> <a href="#">EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEve</a> <a href="#">ntArgs)</a>
<a href="#">OnClick(EventArgs)</a>	<a href="#">OnClientSizeChanged(EventA</a> <a href="#">rgs)</a>	<a href="#">OnContextMenuChanged(Eve</a> <a href="#">ntArgs)</a>
<a href="#">OnContextMenuStripChanged(Eve</a> <a href="#">ntArgs)</a>	<a href="#">OnControlAdded(ControlEve</a> <a href="#">ntArgs)</a>	<a href="#">OnControlRemoved(ControlE</a> <a href="#">ventArgs)</a>
<a href="#">OnCreateControl()</a>	<a href="#">OnCursorChanged(EventArgs</a> <a href="#">)</a>	<a href="#">OnDockChanged(EventArgs)</a>
<a href="#">OnDoubleClick(EventArgs)</a>	<a href="#">OnDpiChangedAfterParent(Ev</a> <a href="#">entArgs)</a>	<a href="#">OnDpiChangedBeforeParent(</a> <a href="#">EventArgs)</a>
<a href="#">OnDragDrop(DragEventArgs)</a>	<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>
<a href="#">OnDragOver(DragEventArgs)</a>	<a href="#">OnEnabledChanged(EventArg</a> <a href="#">s)</a>	<a href="#">OnEnter(EventArgs)</a>
<a href="#">OnFontChanged(EventArgs)</a>	<a href="#">OnForeColorChanged(EventA</a> <a href="#">rgs)</a>	<a href="#">OnGiveFeedback(GiveFeedba</a> <a href="#">ckEventArgs)</a>
<a href="#">OnGotFocus(EventArgs)</a>	<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventAr</a> <a href="#">gs)</a>
<a href="#">OnHelpRequested(HelperEventArgs)</a>	<a href="#">OnImeModeChanged(EventA</a> <a href="#">rgs)</a>	<a href="#">OnInvalidate(InvalidateEven</a> <a href="#">tArgs)</a>
<a href="#">OnKeyDown(KeyEventEventArgs)</a>	<a href="#">OnKeyPress(KeyPressEventAr</a> <a href="#">gs)</a>	<a href="#">OnKeyUp(KeyEventEventArgs)</a>
<a href="#">OnLayout(LayoutEventArgs)</a>	<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventAr</a> <a href="#">gs)</a>
<a href="#">OnLostFocus(EventArgs)</a>	<a href="#">OnMarginChanged(EventArg</a> <a href="#">s)</a>	<a href="#">OnMouseCaptureChanged(Ev</a> <a href="#">entArgs)</a>
<a href="#">OnMouseClick(MouseEventArgs)</a>	<a href="#">OnMouseDoubleClick(Mouse</a> <a href="#">EventArgs)</a>	<a href="#">OnMouseDown(MouseEvent</a> <a href="#">Args)</a>

<a href="#">OnMouseEnter(EventArgs)</a>	<a href="#">OnMouseHover(EventArgs)</a>	<a href="#">OnMouseLeave(EventArgs)</a>
<a href="#">OnMouseMove(MouseEventArgs)</a>	<a href="#">OnMouseUp(MouseEventArgs)</a>	<a href="#">OnMouseWheel(MouseEventArgs)</a>
<a href="#">OnMove(EventArgs)</a>	<a href="#">OnNotifyMessage(Message)</a>	<a href="#">OnPaddingChanged(EventArgs)</a>
<a href="#">OnPaint(PaintEventArgs)</a>	<a href="#">OnPaintBackground(PaintEventArgs)</a>	<a href="#">OnParentBackColorChanged(EventArgs)</a>

<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>	<a href="#">OnParentBindingContextChanged(EventArgs)</a>	<a href="#">OnParentChanged(EventArgs)</a>
<a href="#">OnParentCursorChanged(EventArgs)</a>	<a href="#">OnParentEnabledChanged(EventArgs)</a>	<a href="#">OnParentFontChanged(EventArgs)</a>
<a href="#">OnParentForeColorChanged(EventArgs)</a>	<a href="#">OnParentRightToLeftChanged(EventArgs)</a>	<a href="#">OnParentVisibleChanged(EventArgs)</a>
<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>	<a href="#">OnPrint(PaintEventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>
<a href="#">OnRegionChanged(EventArgs)</a>	<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>
<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>
<a href="#">OnTabIndexChanged(EventArgs)</a>	<a href="#">OnTabStopChanged(EventArgs)</a>	<a href="#">OnTextChanged(EventArgs)</a>
<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArgs)</a>	<a href="#">OnVisibleChanged(EventArgs)</a>
<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>
<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(Message)</a>	<a href="#">PreProcessMessage(Message)</a>
<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>
<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message)</a>	<a href="#">ProcessKeyPreview(Message)</a>

<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>
<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>
<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle)</a> }	<a href="#">Refresh()</a>
<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>
<a href="#">ResetCursor()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>
<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>
<a href="#">RtlTranslateAlignment(ContentAlignment)</a>	<a href="#">RtlTranslateAlignment(HorizontalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRightAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlignment)</a>	<a href="#">RtlTranslateHorizontal(HorizontalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRightAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap)</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">Select()</a>	<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean, Boolean)</a>
<a href="#">SendToBack()</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>
<a href="#">SetStyle(ControlStyles, Boolean)</a>	<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>
<a href="#">Show()</a>	<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>
<a href="#">ToString()</a>	<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>
<a href="#">UpdateZOrder()</a>	<a href="#">WndProc(Message)</a>	

U tabeli 37 su dati svi događaji za kontrolu *GroupBox*.

**Tabela 37.** Tabela događaja za kontrolu *GroupBox* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>
<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>
<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidated</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>
<a href="#">MouseCaptureChanged</a>	<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>
<a href="#">MouseDown</a>	<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>
<a href="#">MouseLeave</a>	<a href="#">MouseMove</a>	<a href="#">MouseUp</a>
<a href="#">MouseWheel</a>	<a href="#">Move</a>	<a href="#">PaddingChanged</a>
<a href="#">Paint</a>	<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>
<a href="#">QueryAccessibilityHelp</a>	<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>
<a href="#">Resize</a>	<a href="#">RightToLeftChanged</a>	<a href="#">SizeChanged</a>
<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>	<a href="#">TabIndexChanged</a>
<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>	<a href="#">Validated</a>
<a href="#">Validating</a>	<a href="#">VisibleChanged</a>	

#### 6.9.11. Kontrola Panel

[Panel](#) (kontrola broj 5 sa slike 26) je kontrola koja sadrži druge kontrole. *Panel* možete da koristite za grupisanje više kontrole, kao što je grupa kontrola *RadioButton*. Kao i kod ostalih kontrola, kao što je kontrola *GroupBox* (kontrola broj 6 sa slike 26), ako je svojstvo *Enabled* kontrole *Panel* postavljeno na false, kontrole sadržane na panelu, takođe će biti onemogućene.

Kontrola panel se prikazuje podrazumevano bez ikakvih graničnika.

Kontrola panel ne prikazuje natpis. Ako je potrebna kontrola slična panel-u, a koja treba da prikazuje natpise, onda se koristiti kontrola *GroupBox*.

U tabeli 38 su data sva svojstva za kontrolu *Panel*.

**Tabela 38. Tabela svojstava za kontrolu Panel [7]**

<a href="#">AccessibilityObject</a>	<a href="#">AccessibleDefaultActionDescription</a>	<a href="#">AccessibleDescription</a>	<a href="#">AccessibleName</a>
<a href="#">AccessibleRole</a>	<a href="#">AllowDrop</a>	<a href="#">Anchor</a>	<a href="#">AutoScroll</a>
<a href="#">AutoScrollMargin</a>	<a href="#">AutoScrollMinSize</a>	<a href="#">AutoScrollOffset</a>	<a href="#">AutoScrollPosition</a>
<a href="#">AutoSize</a>	<a href="#">AutoSizeMode</a>	<a href="#">BackColor</a>	<a href="#">BackgroundImage</a>
<a href="#">BindingContext</a>	<a href="#">BackgroundImageLayout</a>	<a href="#">BindingContext</a>	<a href="#">BorderStyle</a>
<a href="#">Bottom</a>	<a href="#">Bounds</a>	<a href="#">CanEnableIme</a>	<a href="#">CanFocus</a>
<a href="#">CanRaiseEvents</a>	<a href="#">CanSelect</a>	<a href="#">Capture</a>	<a href="#">CausesValidation</a>
<a href="#">ClientRectangle</a>	<a href="#">ClientSize</a>	<a href="#">CompanyName</a>	<a href="#">Container</a>
<a href="#">ContainsFocus</a>	<a href="#">ContextMenu</a>	<a href="#">ContextMenuStrip</a>	<a href="#">Controls</a>
<a href="#">Created</a>	<a href="#">CreateParams</a>	<a href="#">Cursor</a>	<a href="#">DataBindings</a>
<a href="#">DefaultCursor</a>	<a href="#">DefaultImeMode</a>	<a href="#">DefaultMargin</a>	<a href="#">DefaultMaximumSize</a>
<a href="#">DefaultMinimumSize</a>	<a href="#">DefaultPadding</a>	<a href="#">DefaultSize</a>	<a href="#">DesignMode</a>
<a href="#">DeviceDpi</a>	<a href="#">DisplayRectangle</a>	<a href="#">Disposing</a>	<a href="#">Dock</a>
<a href="#">DockPadding</a>	<a href="#">DoubleBuffered</a>	<a href="#">Enabled</a>	<a href="#">Events</a>
<a href="#">Focused</a>	<a href="#">Font</a>	<a href="#">FontHeight</a>	<a href="#">ForeColor</a>
<a href="#">Handle</a>	<a href="#">HasChildren</a>	<a href="#">Height</a>	<a href="#">HorizontalScroll</a>
<a href="#">HScroll</a>	<a href="#">ImeMode</a>	<a href="#">ImeModeBase</a>	<a href="#">InvokeRequired</a>
<a href="#">IsAccessible</a>	<a href="#">IsDisposed</a>	<a href="#">IsHandleCreated</a>	<a href="#">IsMirrored</a>
<a href="#">LayoutEngine</a>	<a href="#">Left</a>	<a href="#">Location</a>	<a href="#">Margin</a>
<a href="#">MaximumSize</a>	<a href="#">MinimumSize</a>	<a href="#">Name</a>	<a href="#">Padding</a>

<a href="#">Parent</a>	<a href="#">PreferredSize</a>	<a href="#">ProductName</a>	<a href="#">ProductVersion</a>
<a href="#">RecreatingHandle</a>	<a href="#">Region</a>	<a href="#">RenderRightToLeft</a>	<a href="#">ResizeRedraw</a>
<a href="#">Right</a>	<a href="#">RightToLeft</a>	<a href="#">ScaleChildren</a>	<a href="#">ShowFocusCues</a>
<a href="#">ShowKeyboardCues</a>	<a href="#">Site</a>	<a href="#">Size</a>	<a href="#">TabIndex</a>
<a href="#">TabStop</a>	<a href="#">Tag</a>	<a href="#">Text</a>	<a href="#">Top</a>
<a href="#">TopLevelControl</a>	<a href="#">UseWaitCursor</a>	<a href="#">VerticalScroll</a>	<a href="#">Visible</a>
<a href="#">VScroll</a>	<a href="#">Width</a>	<a href="#">WindowTarget</a>	

U tabeli 39 su date sve metode za kontrolu *Panel*.

**Tabela 39.** Tabela metoda za kontrolu *Panel* [7]

<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32)</a>	<a href="#">AccessibilityNotifyClients(AccessibleEvents, Int32, Int32)</a>	<a href="#">AdjustFormScrollbars(Boolean)</a>
---------------------------------------------------------------------	----------------------------------------------------------------------------	-----------------------------------------------

<a href="#">BeginInvoke(Delegate)</a>	<a href="#">BeginInvoke(Delegate, Object[])</a>	<a href="#">BringToFront()</a>
<a href="#">Contains(Control)</a>	<a href="#">CreateAccessibilityInstance()</a>	<a href="#">CreateControl()</a>
<a href="#">CreateControlsInstance()</a>	<a href="#">CreateGraphics()</a>	<a href="#">CreateHandle()</a>
<a href="#">CreateObjRef(Type)</a>	<a href="#">DefWndProc(Message)</a>	<a href="#">DestroyHandle()</a>
<a href="#">Dispose()</a>	<a href="#">Dispose(Boolean)</a>	<a href="#">DoDragDrop(Object, DragDropEffects)</a>
<a href="#">DrawToBitmap(Bitmap, Rectangle)</a>	<a href="#">EndInvoke(IAsyncResult)</a>	<a href="#">Equals(Object)</a>
<a href="#">FindForm()</a>	<a href="#">Focus()</a>	<a href="#">GetAccessibilityObjectById(Int32)</a>
<a href="#">GetAutoSizeMode()</a>	<a href="#">GetChildAtPoint(Point)</a>	<a href="#">GetChildAtPoint(Point, GetChildAtPointSkip)</a>
<a href="#">GetContainerControl()</a>	<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>
<a href="#">GetNextControl(Control, Boolean)</a>	<a href="#">GetPreferredSize(Size)</a>	<a href="#">GetScaledBounds(Rectangle, SizeF, BoundsSpecified)</a>
<a href="#">GetScrollState(Int32)</a>	<a href="#">GetService(Type)</a>	<a href="#">GetStyle(ControlStyles)</a>
<a href="#">GetTopLevel()</a>	<a href="#">GetType()</a>	<a href="#">Hide()</a>

<a href="#">InitializeLifetimeService()</a>	<a href="#">InitLayout()</a>	<a href="#">Invalidate()</a>
<a href="#">Invalidate(Boolean)</a>	<a href="#">Invalidate(Rectangle)</a>	<a href="#">Invalidate(Rectangle, Boolean)</a>
<a href="#">Invalidate(Region)</a>	<a href="#">Invalidate(Region, Boolean)</a>	<a href="#">Invoke(Delegate)</a>
<a href="#">Invoke(Delegate, Object[])</a>	<a href="#">InvokeGotFocus(Control, EventArgs)</a>	<a href="#">InvokeLostFocus(Control, EventArgs)</a>
<a href="#">InvokeOnClick(Control, EventArgs)</a>	<a href="#">InvokePaint(Control, PaintEventArgs)</a>	<a href="#">InvokePaintBackground(Control, PaintEventArgs)</a>
<a href="#">IsInputChar(Char)</a>	<a href="#">IsInputKey(Keys)</a>	<a href="#">LogicalToDeviceUnits(Int32)</a>
<a href="#">LogicalToDeviceUnits(Size)</a>	<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>
<a href="#">NotifyInvalidate(Rectangle)</a>	<a href="#">OnAutoSizeChanged(EventArgs)</a>	<a href="#">OnBackColorChanged(EventArgs)</a>
<a href="#">OnBackgroundImageChanged(EventArgs)</a>	<a href="#">OnBackgroundImageLayoutChanged(EventArgs)</a>	<a href="#">OnBindingContextChanged(EventArgs)</a>
<a href="#">OnCausesValidationChanged(EventArgs)</a>	<a href="#">OnChangeUICues(UICuesEventArgs)</a>	<a href="#">OnClick(EventArgs)</a>

<a href="#">OnClientSizeChanged(EventArgs)</a>	<a href="#">OnContextMenuChanged(EventArgs)</a>	<a href="#">OnContextMenuStripChanged(EventArgs)</a>
<a href="#">OnControlAdded(ControlEventArgs)</a>	<a href="#">OnControlRemoved(ControlEventArgs)</a>	<a href="#">OnCreateControl()</a>
<a href="#">OnCursorChanged(EventArgs)</a>	<a href="#">OnDockChanged(EventArgs)</a>	<a href="#">OnDoubleClick(EventArgs)</a>
<a href="#">OnDpiChangedAfterParent(EventArgs)</a>	<a href="#">OnDpiChangedBeforeParent(EventArgs)</a>	<a href="#">OnDragDrop(DragEventArgs)</a>
<a href="#">OnDragEnter(DragEventArgs)</a>	<a href="#">OnDragLeave(EventArgs)</a>	<a href="#">OnDragOver(DragEventArgs)</a>
<a href="#">OnEnabledChanged(EventArgs)</a>	<a href="#">OnEnter(EventArgs)</a>	<a href="#">OnFontChanged(EventArgs)</a>
<a href="#">OnForeColorChanged(EventArgs)</a>	<a href="#">OnGiveFeedback(GiveFeedbackEventArgs)</a>	<a href="#">OnGotFocus(EventArgs)</a>
<a href="#">OnHandleCreated(EventArgs)</a>	<a href="#">OnHandleDestroyed(EventArgs)</a>	<a href="#">OnHelpRequested(HelpEventArgs)</a>

<a href="#">OnImeModeChanged(EventArgs)</a>	<a href="#">OnInvalidate(InvalidateEventArgs)</a>	<a href="#">OnKeyDown(KeyboardEventArgs)</a>
<a href="#">OnKeyPress(KeyPressEventArgs)</a>	<a href="#">OnKeyUp(KeyboardEventArgs)</a>	<a href="#">OnLayout(LayoutEventArgs)</a>
<a href="#">OnLeave(EventArgs)</a>	<a href="#">OnLocationChanged(EventArgs)</a>	<a href="#">OnLostFocus(EventArgs)</a>
<a href="#">OnMarginChanged(EventArgs)</a>	<a href="#">OnMouseCaptureChanged(EventArgs)</a>	<a href="#">OnMouseClick(MouseEventArgs)</a>
<a href="#">OnMouseDoubleClick(MouseEventEventArgs)</a>	<a href="#">OnMouseDown(MouseEventArgs)</a>	<a href="#">OnMouseEnter(EventArgs)</a>
<a href="#">OnMouseHover(EventArgs)</a>	<a href="#">OnMouseLeave(EventArgs)</a>	<a href="#">OnMouseMove(MouseEventArgs)</a>
<a href="#">OnMouseUp(MouseEventArgs)</a>	<a href="#">OnMouseWheel(MouseEventArgs)</a>	<a href="#">OnMove(EventArgs)</a>
<a href="#">OnNotifyMessage(Message)</a>	<a href="#">OnPaddingChanged(EventArgs)</a>	<a href="#">OnPaint(PaintEventArgs)</a>
<a href="#">OnPaintBackground(PaintEventArgs)</a>	<a href="#">OnParentBackColorChanged(EventArgs)</a>	<a href="#">OnParentBackgroundImageChanged(EventArgs)</a>
<a href="#">OnParentBindingContextChanged(EventArgs)</a>	<a href="#">OnParentChanged(EventArgs)</a>	<a href="#">OnParentCursorChanged(EventArgs)</a>
<a href="#">OnParentEnabledChanged(EventArgs)</a>	<a href="#">OnParentFontChanged(EventArgs)</a>	<a href="#">OnParentForeColorChanged(EventArgs)</a>

<a href="#">OnParentRightToLeftChanged(EventArgs)</a>	<a href="#">OnParentVisibleChanged(EventArgs)</a>	<a href="#">OnPreviewKeyDown(PreviewKeyDownEventArgs)</a>
<a href="#">OnPrint(PaintEventArgs)</a>	<a href="#">OnQueryContinueDrag(QueryContinueDragEventArgs)</a>	<a href="#">OnRegionChanged(EventArgs)</a>
<a href="#">OnResize(EventArgs)</a>	<a href="#">OnRightToLeftChanged(EventArgs)</a>	<a href="#">OnScroll(ScrollEventArgs)</a>
<a href="#">OnSizeChanged(EventArgs)</a>	<a href="#">OnStyleChanged(EventArgs)</a>	<a href="#">OnSystemColorsChanged(EventArgs)</a>

<a href="#">OnTabIndexChanged(EventArgs)</a>	<a href="#">OnTabStopChanged(EventAr gs)</a>	<a href="#">OnTextChanged(EventArgs)</a>
<a href="#">OnValidated(EventArgs)</a>	<a href="#">OnValidating(CancelEventArg s)</a>	<a href="#">OnVisibleChanged(EventArgs )</a>
<a href="#">PerformLayout()</a>	<a href="#">PerformLayout(Control, String)</a>	<a href="#">PointToClient(Point)</a>
<a href="#">PointToScreen(Point)</a>	<a href="#">PreProcessControlMessage(M essage)</a>	<a href="#">PreProcessMessage(Message )</a>
<a href="#">ProcessCmdKey(Message, Keys)</a>	<a href="#">ProcessDialogChar(Char)</a>	<a href="#">ProcessDialogKey(Keys)</a>
<a href="#">ProcessKeyEventArgs(Message)</a>	<a href="#">ProcessKeyMessage(Message )</a>	<a href="#">ProcessKeyPreview(Message)</a>
<a href="#">ProcessMnemonic(Char)</a>	<a href="#">RaiseDragEvent(Object, DragEventArgs)</a>	<a href="#">RaiseKeyEvent(Object, KeyEventArgs)</a>
<a href="#">RaiseMouseEvent(Object, MouseEventArgs)</a>	<a href="#">RaisePaintEvent(Object, PaintEventArgs)</a>	<a href="#">RecreateHandle()</a>
<a href="#">RectangleToClient(Rectangle)</a>	<a href="#">RectangleToScreen(Rectangle )</a>	<a href="#">Refresh()</a>
<a href="#">RescaleConstantsForDpi(Int32, Int32)</a>	<a href="#">ResetBackColor()</a>	<a href="#">ResetBindings()</a>
<a href="#">ResetCursor()</a>	<a href="#">ResetFont()</a>	<a href="#">ResetForeColor()</a>
<a href="#">ResetImeMode()</a>	<a href="#">ResetMouseEventArgs()</a>	<a href="#">ResetRightToLeft()</a>
<a href="#">ResetText()</a>	<a href="#">ResumeLayout()</a>	<a href="#">ResumeLayout(Boolean)</a>
<a href="#">RtlTranslateAlignment(ContentAli gnment)</a>	<a href="#">RtlTranslateAlignment(Horizo ntalAlignment)</a>	<a href="#">RtlTranslateAlignment(LeftRig htAlignment)</a>
<a href="#">RtlTranslateContent(ContentAlign ment)</a>	<a href="#">RtlTranslateHorizontal(Horizo ntalAlignment)</a>	<a href="#">RtlTranslateLeftRight(LeftRigh tAlignment)</a>
<a href="#">Scale(Single)</a>	<a href="#">Scale(Single, Single)</a>	<a href="#">Scale(SizeF)</a>
<a href="#">ScaleBitmapLogicalToDevice(Bitmap )</a>	<a href="#">ScaleControl(SizeF, BoundsSpecified)</a>	<a href="#">ScaleCore(Single, Single)</a>
<a href="#">ScrollControlIntoView(Control)</a>	<a href="#">ScrollToControl(Control)</a>	<a href="#">Select()</a>

<a href="#">Select(Boolean, Boolean)</a>	<a href="#">SelectNextControl(Control, Boolean, Boolean, Boolean)</a>	<a href="#">SendToBack()</a>
<a href="#">SetAutoScrollMargin(Int32, Int32)</a>	<a href="#">SetAutoSizeMode(AutoSizeMode)</a>	<a href="#">SetBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">SetBounds(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetBoundsCore(Int32, Int32, Int32, Int32, BoundsSpecified)</a>	<a href="#">SetClientSizeCore(Int32, Int32)</a>
<a href="#">SetDisplayRectLocation(Int32, Int32)</a>	<a href="#">SetScrollState(Int32, Boolean)</a>	<a href="#">SetStyle(ControlStyles, Boolean)</a>
<a href="#">SetTopLevel(Boolean)</a>	<a href="#">SetVisibleCore(Boolean)</a>	<a href="#">Show()</a>
<a href="#">SizeFromClientSize(Size)</a>	<a href="#">SuspendLayout()</a>	<a href="#">ToString()</a>
<a href="#">Update()</a>	<a href="#">UpdateBounds()</a>	<a href="#">UpdateBounds(Int32, Int32, Int32, Int32)</a>
<a href="#">UpdateBounds(Int32, Int32, Int32, Int32, Int32)</a>	<a href="#">UpdateStyles()</a>	<a href="#">UpdateZOrder()</a>
<a href="#">WndProc(Message)</a>		

U tabeli 40 su dati svi događaji za kontrolu *Panel*.

**Tabela 40.** Tabela događaja za kontrolu *Panel* [7]

<a href="#">AutoSizeChanged</a>	<a href="#">BackColorChanged</a>	<a href="#">BackgroundImageChanged</a>
<a href="#">BackgroundImageLayoutChanged</a>	<a href="#">BindingContextChanged</a>	<a href="#">CausesValidationChanged</a>
<a href="#">ChangeUICues</a>	<a href="#">Click</a>	<a href="#">ClientSizeChanged</a>
<a href="#">ContextMenuChanged</a>	<a href="#">ContextMenuStripChanged</a>	<a href="#">ControlAdded</a>
<a href="#">ControlRemoved</a>	<a href="#">CursorChanged</a>	<a href="#">Disposed</a>
<a href="#">DockChanged</a>	<a href="#">DoubleClick</a>	<a href="#">DpiChangedAfterParent</a>
<a href="#">DpiChangedBeforeParent</a>	<a href="#">DragDrop</a>	<a href="#">DragEnter</a>
<a href="#">DragLeave</a>	<a href="#">DragOver</a>	<a href="#">EnabledChanged</a>
<a href="#">Enter</a>	<a href="#">FontChanged</a>	<a href="#">ForeColorChanged</a>
<a href="#">GiveFeedback</a>	<a href="#">GotFocus</a>	<a href="#">HandleCreated</a>

<a href="#">HandleDestroyed</a>	<a href="#">HelpRequested</a>	<a href="#">ImeModeChanged</a>
<a href="#">Invalidate</a>	<a href="#">KeyDown</a>	<a href="#">KeyPress</a>
<a href="#">KeyUp</a>	<a href="#">Layout</a>	<a href="#">Leave</a>
<a href="#">LocationChanged</a>	<a href="#">LostFocus</a>	<a href="#">MarginChanged</a>
<a href="#">MouseCaptureChanged</a>	<a href="#">MouseClick</a>	<a href="#">MouseDoubleClick</a>
<a href="#">MouseDown</a>	<a href="#">MouseEnter</a>	<a href="#">MouseHover</a>
<a href="#">MouseLeave</a>	<a href="#">MouseMove</a>	<a href="#">MouseUp</a>
<a href="#">MouseWheel</a>	<a href="#">Move</a>	<a href="#">PaddingChanged</a>
<a href="#">Paint</a>	<a href="#">ParentChanged</a>	<a href="#">PreviewKeyDown</a>
<a href="#">QueryAccessibilityHelp</a>	<a href="#">QueryContinueDrag</a>	<a href="#">RegionChanged</a>
<a href="#">Resize</a>	<a href="#">RightToLeftChanged</a>	<a href="#">Scroll</a>
<a href="#">SizeChanged</a>	<a href="#">StyleChanged</a>	<a href="#">SystemColorsChanged</a>
<a href="#">TabIndexChanged</a>	<a href="#">TabStopChanged</a>	<a href="#">TextChanged</a>
<a href="#">Validated</a>	<a href="#">Validating</a>	<a href="#">VisibleChanged</a>

### 6.9.12. Kontrola Timer

Tajmer ili [Timer](#) se koristi za događaje koji se dešavaju u nekom intervalu. Intervale trajanja nekog događaja definiše korisnik.

Kada se koristi komponenta tajmer, koristi se događaj *Tick* da bi se izvršio određeni događaj. Kad god je svojstvo *Enabled* postavljeno na true, a svojstvo *Interval* veće od nule, događaj *Tick* se postavlja u intervalima na osnovu postavke svojstva *Interval*.

Klasa *Timer* pruža metode za podešavanje intervala i pokretanje i zaustavljanje tajmera.

U tabeli 41 su data sva svojstva za kontrolu *Timer*.

**Tabela 41.** Tabela svojstava za kontrolu *Timer* [7]

<a href="#">CanRaiseEvents</a>	<a href="#">Container</a>	<a href="#">DesignMode</a>	<a href="#">Enabled</a>
<a href="#">Events</a>	<a href="#">Interval</a>	<a href="#">Site</a>	<a href="#">Tag</a>

U tabeli 42 su date sve metode za kontrolu *Timer*.

**Tabela 42.** Tabela metoda za kontrolu *Timer* [7]

<a href="#">CreateObjRef(Type)</a>	<a href="#">Dispose()</a>	<a href="#">Dispose(Boolean)</a>
<a href="#">Equals(Object)</a>	<a href="#">GetHashCode()</a>	<a href="#">GetLifetimeService()</a>
<a href="#">GetService(Type)</a>	<a href="#">GetType()</a>	<a href="#">InitializeLifetimeService()</a>
<a href="#">MemberwiseClone()</a>	<a href="#">MemberwiseClone(Boolean)</a>	<a href="#">OnTick(EventArgs)</a>
<a href="#">Start()</a>	<a href="#">Stop()</a>	<a href="#">ToString()</a>

Komponenta tajmer ima samo dva događaja, i to su: [Disposed](#) i [Tick](#).

## LITERATURA

- [1] C# Programske jezike, Materijali sa Univeziteta Metropolitan, Beograd sa sajta:  
<http://digis.edu.rs/course/view.php?id=86> (7.7.2020)
- [2] Nikola Bošković, Programiranje III, materijali sa nastave u elektronskoj formi, ITHS, 2019/2020. godina
- [3] Svetlana Andelić, VISUAL C#.NET PROGRAMIRANJE, Zbirka rešenih zadataka, Visoka škola strukovnih studija za informacione tehnologije ITS, Beograd, 2015.
- [4] Goran Agatonović, Objektno programiranje, Beogradska poslovna škola, Beograd, 2010.
- [5] Jesse Liberty, Programiranje na jeziku C#, prevod četvrtog izdanja, Mikro knjiga, Beograd, 2007.
- [6] Zoran Ćirović, Ivan Dunderski, Tehnike vizuelnog programiranja C#, Visoka elektrotehnička škola, Beograd, 2005.
- [7] <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms?view=netcore-3.1> (7.7.2020)