

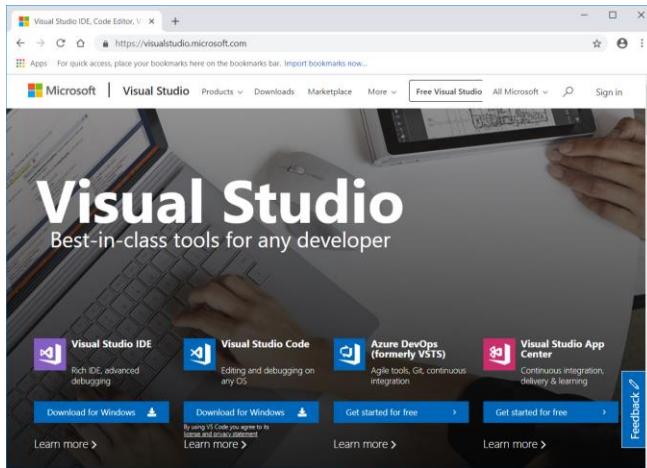
Objektno programiranje

Dr. Goran Aritonović
goran.aritonovic@bbs.edu.rs
kabinet 211

Softver

- Visual Studio 2017 (15.7 ili noviji)
- Visual Studio Code
- SQL Server 2017 Express
- SQL Server Management Studio Express

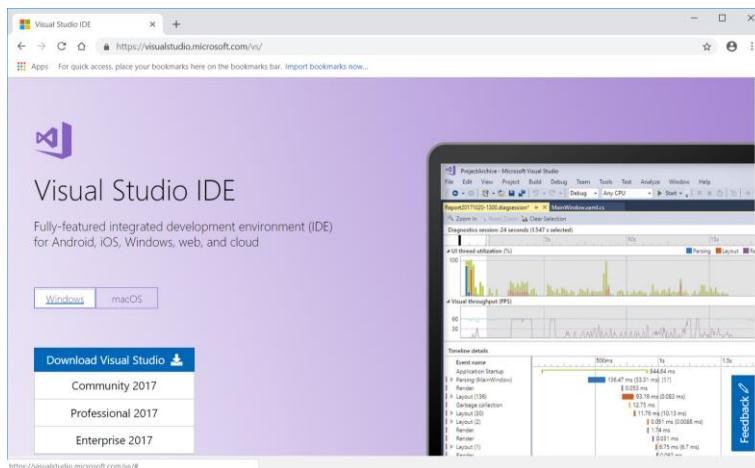
Visual Studio



<https://visualstudio.microsoft.com/>

3

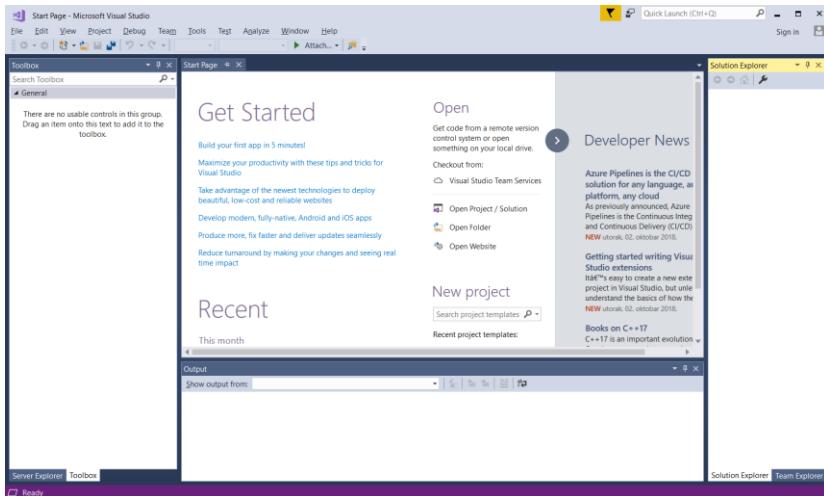
Web stranica Visual Studio IDE



<https://visualstudio.microsoft.com/vs/>

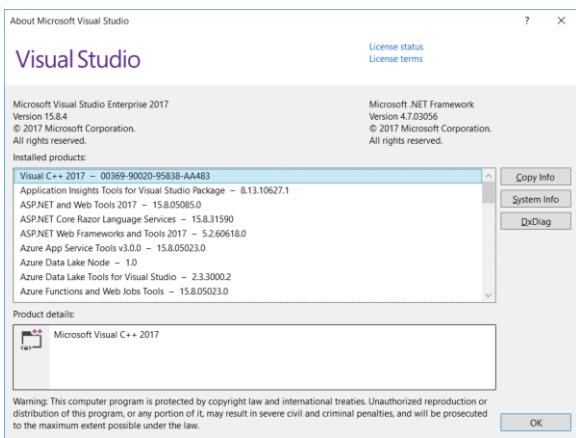
4

Visual Studio 2017



5

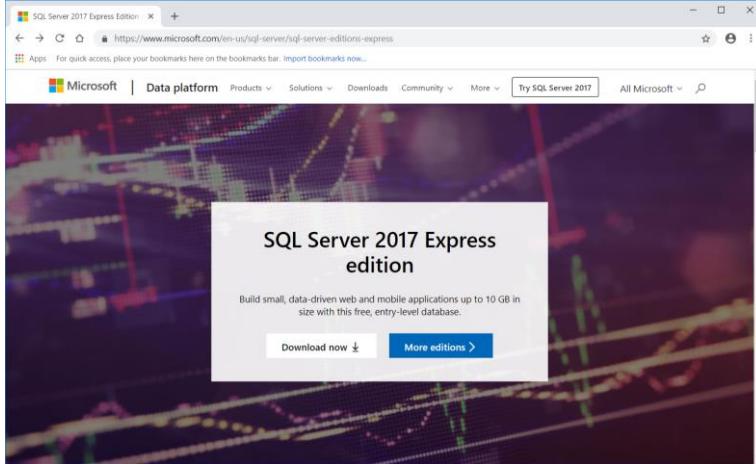
Verzija Visual Studija



6

3

SQL server

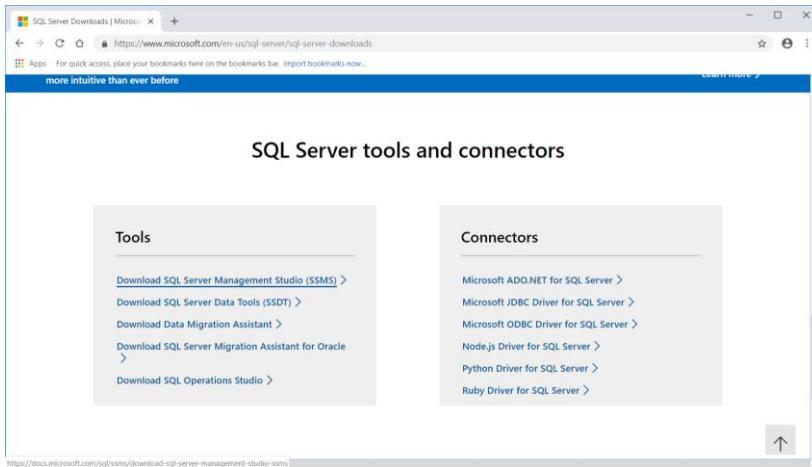


<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

7

SQL Server Management Studio (SSMS)

<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>



8

Literatura

- Materijali sa predavanja u elektronskom formatu
- MSDN -[Microsoft Solution Developer Network](#)

9

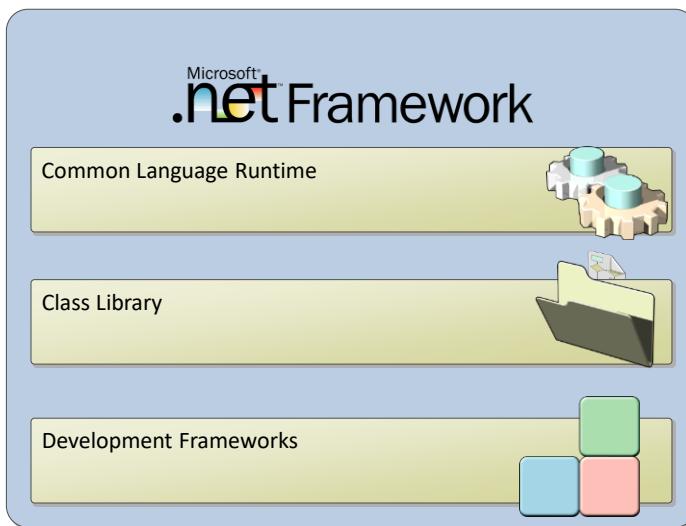
C# .NET Framework, .NET Core

.NET Framework

- .NET Framework je razvojno okruženje koje omogućava kreiranje .NET aplikacija
- .NET Framework je nastao 2000 godine
- Tekuća verzija je 4.7.2
- Predstavlja osnovu .NET platforme
- Sastoji se iz komponenti:
 - Common Language Runtime (CLR) – virtualna mašina odgovorna za izvršavanje koda
 - .NET framework biblioteka klase, biblioteka klasa koja omogućava kreiranje windows, web aplikacija i web servisa
 - kolekcija razvojnih frameworka (ASP.NET, WPF, WCF,...)

11

Struktura .NET Frameworka



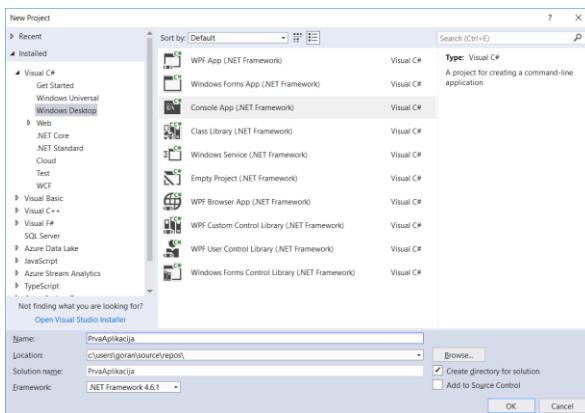
12

Princip rada .NET Frameworka

- Visual Studio .NET je integrisano razvojno okruženje (IDE) koje omogućava kreiranje .NET aplikacija
- Kada se kompajlira aplikacija u Visual Studio .NET-u source kod se translira u tzv. Microsoft Intermediate Language (MSIL)
- Posle kompajliranja runtime je komponenta koja upravlja izvršavanjem aplikacije
- Runtime konzumira izlaz koga proizvodi kompajler
- Runtime uključuje karakteristiku koja se naziva JIT – just in time compilation, transliranje MSIL koda u mašinski kod

13

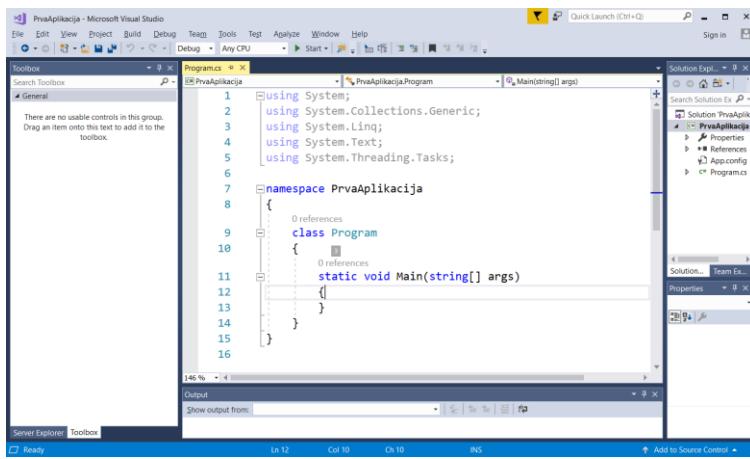
Kreiranje novog projekta



File → New Project, Console Application

14

Konzolna aplikacija



15

Osnove C# programa

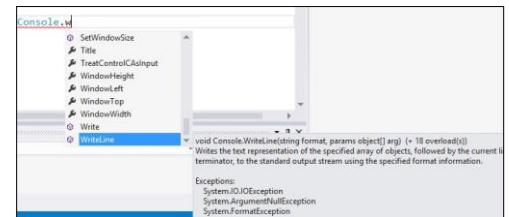
- Izvršavanje programa počinje od Main() metode
- U C# aplikaciji može postojati samo jedna Main() metoda
- Ključna reč using se odnosi na korišćenje .NET Framework biblioteke klase
- Klase u .NET Framework-u su organizovane u tzv. namespace – ove (prostore imena)
- Svaka aplikacija ima Main() metodu definisani u jednoj od njenih klasa
- Main() metoda je statička što znači da je globalna, i klasa se ne mora instancirati da bi se metoda pozvala
- C# razlikuje velika i mala slova
- Klasa Console se nalazi u namespace-u System

16

Prvi C# program

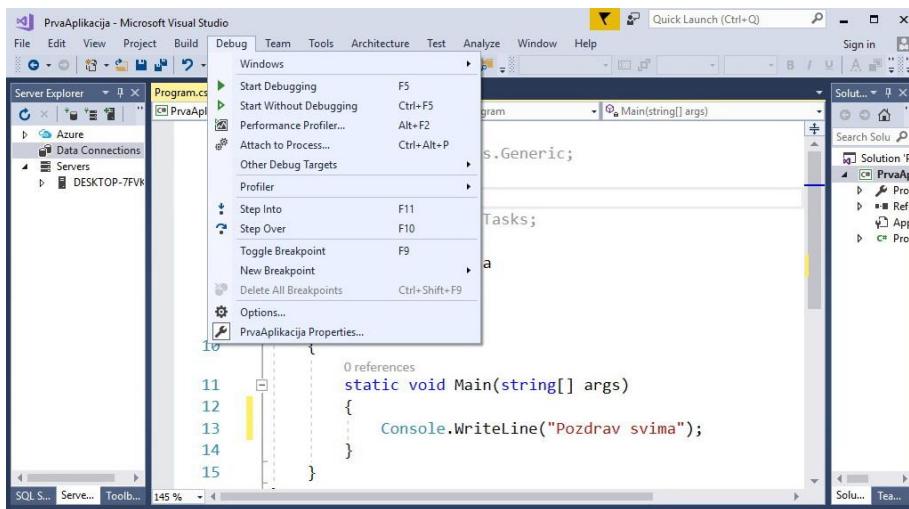
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PrvaAplikacija
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Dobar dan svima");
        }
    }
}
```



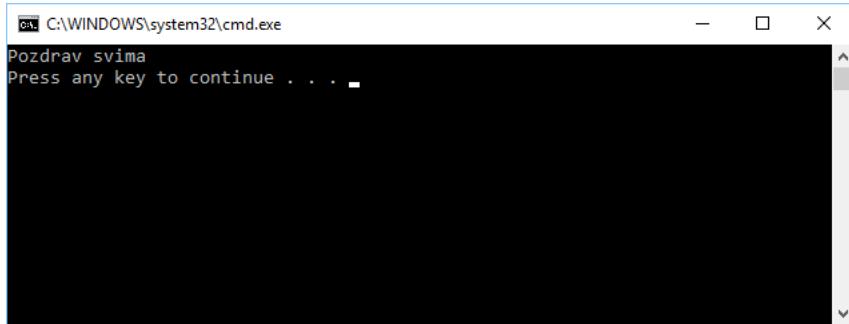
17

Pokretanje aplikacije



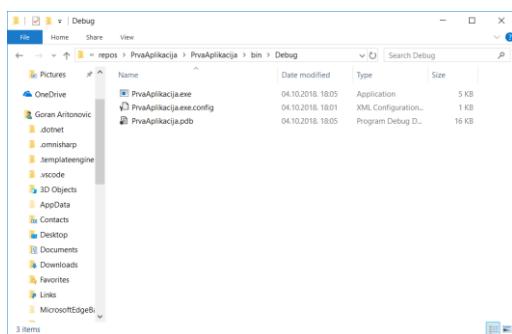
18

Rezultat prikazan u konzoli



19

Putanja do izvršne verzije programa



20

Metode WriteLine() i ReadLine()

- Pokretanje aplikacije F5 (debug mode) ili CTRL+F5
- Ulazni parametar WriteLine() metode je string koji treba prikazati u konzoli
- Posle ispisa teksta kursor prelazi u novi red
- Odmah nakon ispisa teksta konzola se zatvara
- Da bi tekst ostao na ekranu koristi se metoda ReadLine()
- Metoda ReadLine() vraća liniju teksta (string) koji je uneo korisnik
- Metoda ReadLine() čeka korisnički unos koji se prosleđuje metodi pritiskom na taster ENTER

```
static void Main(string[] args)
{
    Console.WriteLine("Dobar dan svima");
    Console.ReadLine();
}
```

21

Pisanje komentara

- Komentar je tekst koji se ignoriše od strane kompjajlera
- Komentar u jednoj liniji piše se korišćenjem znaka //
- Komentar u više linija /* */

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World !!!");

    // metoda ReadLine() ceka ENTER da bi se izvrsila

    /* metoda ReadLine() prihvata tekst koji korisnik unosi u konzoli i
       nakon pritiska na taster ENTER smesta ga u string promenljivu
       koju treba definisati */

    Console.ReadLine();
}
```

22

.NET Core

- .NET Core je razvojna platforma opšte namene
- .NET Core je međuplatformska verzija.NET Frameworka
- Podržava standardne .NET biblioteke
- To je platforma otvorenog koda
- Kompatibilna je sa .NET frameworkom
- Postoji na platformama Windows, macOS i Linux

23

.NET Core

- Trenutno ne podržava sve modele aplikacija za windows (npr. WPF aplikacije)
- Potrebno je instalirati .NET Core SDK 2.1 da bismo mogli da razvijamo .NET Core aplikacije

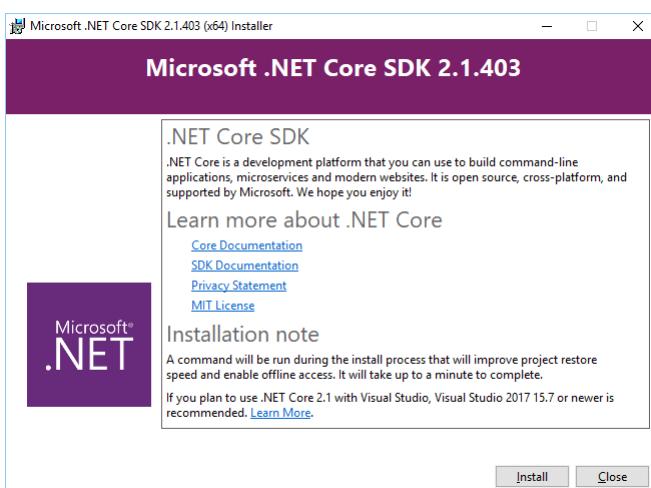
24

.NET Core distribucija

- Sadrži CoreCLR runtime, mašina koja pretvara .NET intermediate language IL u mašinski kod
- Sadrži pridružene biblioteke
- Sadrži **dotnet** app launcher (alat za pokretanje .NET Core aplikacija)

25

Instalirati Microsoft.NET Core SDK



<https://www.microsoft.com/net/download>

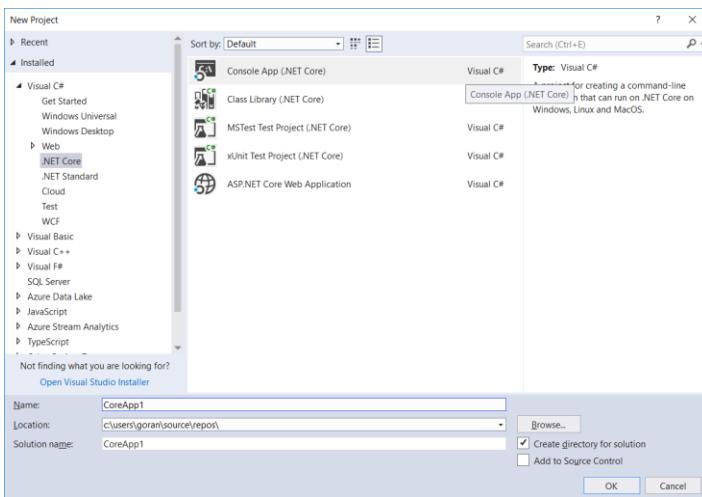
26

Informacije o instaliranom softveru



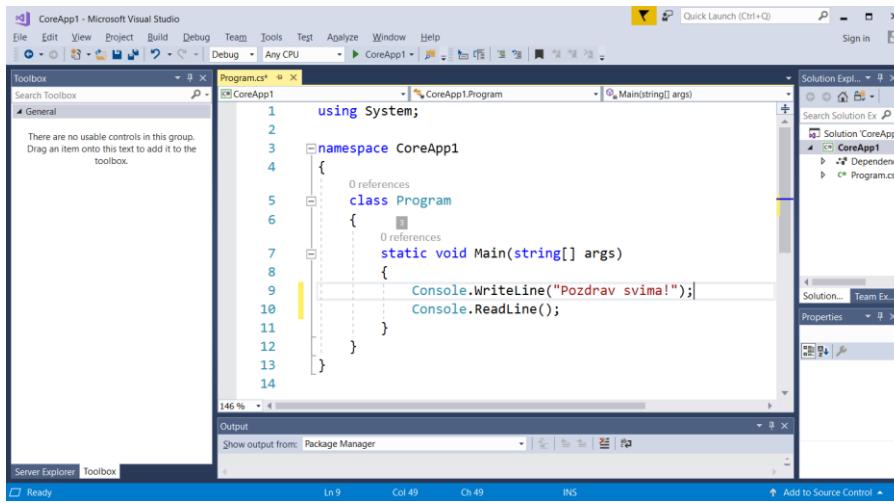
27

Kreiranje .NET Core konzolne aplikacije



28

.NET Core aplikacija



29

Folder .NET Core konzolne aplikacije

```

Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\goran>cd C:\Users\goran\source\repos\CoreApp1\CoreApp1
C:\Users\goran>dir
Volume in drive C has no label.
Volume Serial Number is 2G6A-7752

Directory of C:\Users\goran\source\repos\CoreApp1\CoreApp1

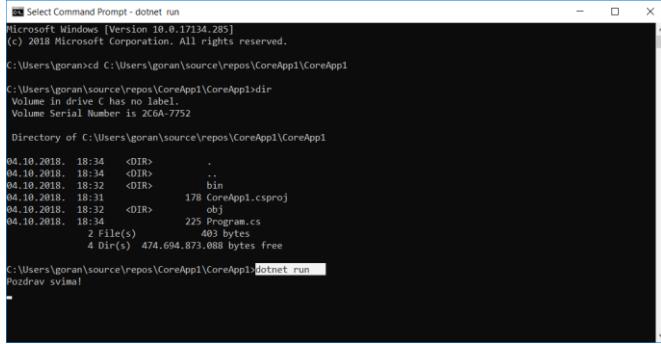
04.10.2018. 18:34 <DIR> .
04.10.2018. 18:34 <DIR> ..
04.10.2018. 18:32 <DIR> bin
04.10.2018. 18:31 178 CoreApp1.csproj
04.10.2018. 18:32 <DIR> obj
04.10.2018. 18:34 225 Program.cs
2 File(s) 403 bytes
4 Dir(s) 474.694.873.088 bytes free

C:\Users\goran>

```

30

Pokretanje aplikacije



```
Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\goran>cd C:\Users\goran\source\repos\CoreApp1\CoreApp1

C:\Users\goran\source\repos\CoreApp1\CoreApp1>dir
Volume in drive C has no label.
Volume Serial Number is 2C6A-7752

Directory of C:\Users\goran\source\repos\CoreApp1\CoreApp1

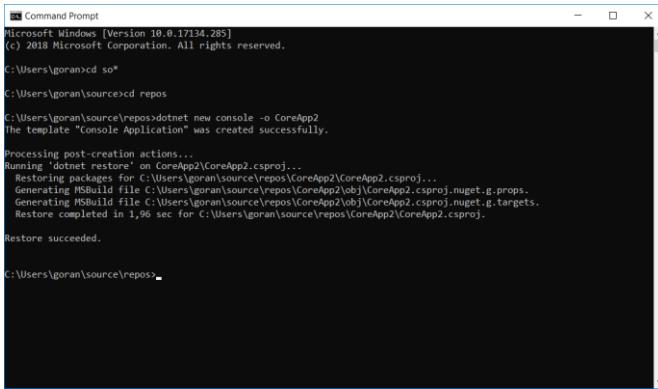
04.10.2018  18:34    <DIR>          .
04.10.2018  18:34    <DIR>          bin
04.10.2018  18:31            178 CoreApp1.csproj
04.10.2018  18:32    <DIR>          obj
04.10.2018  18:34            225 Program.cs
                           2 File(s)       403 bytes
                           4 Dir(s)   474.694.873.088 bytes free

C:\Users\goran\source\repos\CoreApp1\CoreApp1>dotnet run
Pozdrav svima!
```

31

Kreiranje aplikacije kroz konzolu

```
C:\Users\Goran\source\repos>dotnet new console -o CoreApp2
```



```
Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\goran>cd so*
C:\Users\goran\source>cd repos

C:\Users\goran\source\repos>dotnet new console -o CoreApp2
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on CoreApp2\CoreApp2.csproj...
Restoring packages for C:\Users\goran\source\repos\CoreApp2\CoreApp2.csproj...
Generating MSBuild file C:\Users\goran\source\repos\CoreApp2\obj\CoreApp2.csproj.nuget.g.props.
Generating MSBuild file C:\Users\goran\source\repos\CoreApp2\obj\CoreApp2.csproj.nuget.g.targets.
Restore completed in 1,96 sec for C:\Users\goran\source\repos\CoreApp2\CoreApp2.csproj.

Restore succeeded.

C:\Users\goran\source\repos>
```

32

Sadržaj kreiranog foldera

```

Command Prompt
C:\Users\goran\source\repos>dotnet new console -o CoreApp2
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on CoreApp2\CoreApp2.csproj...
  Restoring packages for C:\Users\goran\source\repos\CoreApp2\CoreApp2.csproj...
  Generating MSBuild file C:\Users\goran\source\repos\CoreApp2\obj\CoreApp2.csproj.nuget.g.props.
  Generating MSBuild file C:\Users\goran\source\repos\CoreApp2\obj\CoreApp2.csproj.nuget.g.targets.
  Restore completed for C:\Users\goran\source\repos\CoreApp2\CoreApp2.csproj.

Restore succeeded.

C:\Users\goran\source\repos>cd CoreApp2

C:\Users\goran\source\repos\CoreApp2>dir
Volume in drive C has no label.
Volume Serial Number is 2G6A-7752

Directory of C:\Users\goran\source\repos\CoreApp2

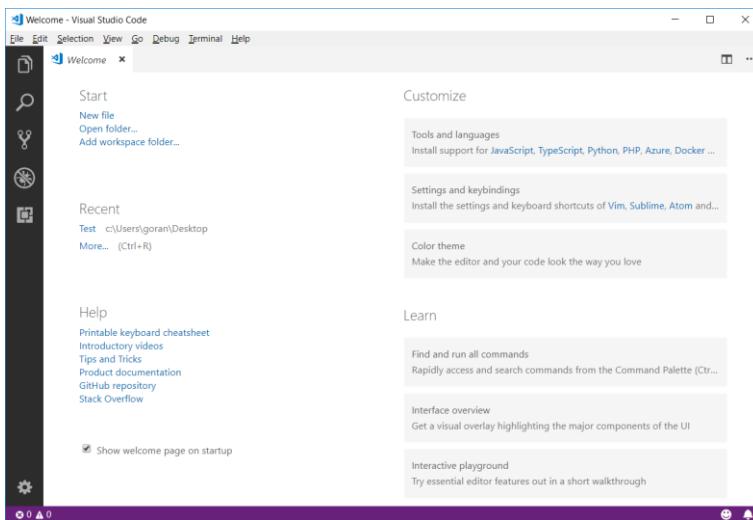
04.10.2018. 18:49    <DIR>      .
04.10.2018. 18:49    <DIR>      ..
04.10.2018. 18:49           178 CoreApp2.csproj
04.10.2018. 18:49    <DIR>      obj
04.10.2018. 18:49           198 Program.cs
               2 File(s)       368 bytes
               3 Dir(s)   469.809.614.848 bytes free

C:\Users\goran\source\repos\CoreApp2>

```

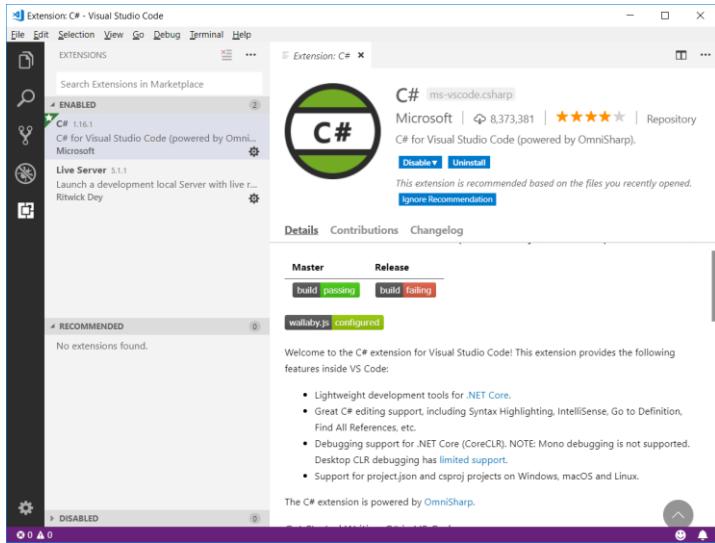
33

Okruženje VS Code



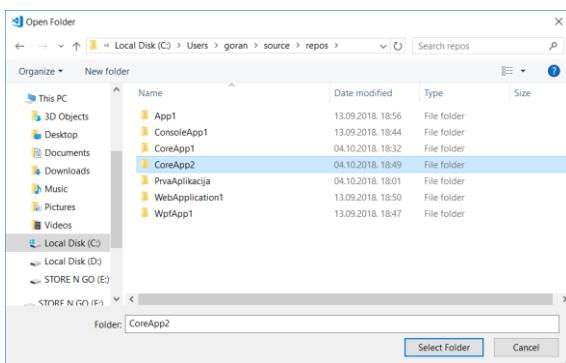
34

C# ekstenzija za VS Code



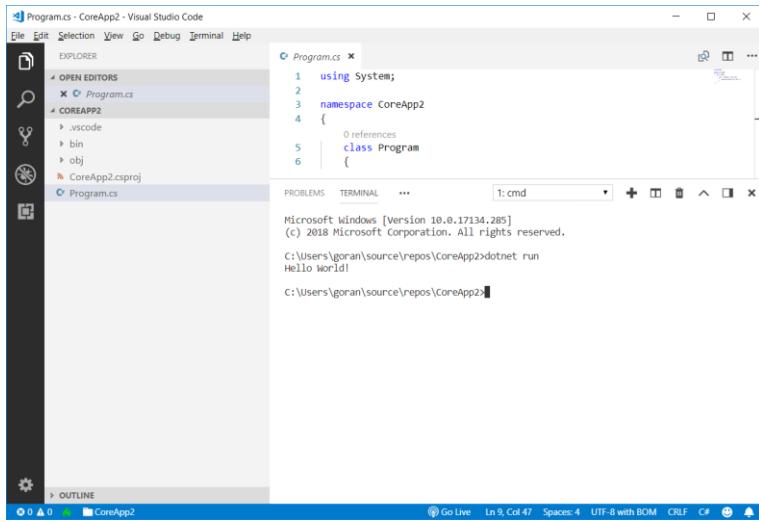
35

Izbor foldera



36

Pokretanje aplikacije



The screenshot shows the Visual Studio Code interface with a C# project named 'CoreApp2'. The 'Program.cs' file is open, displaying the following code:

```
1  using System;
2
3  namespace CoreApp2
4  {
5      class Program
6      {

```

The terminal window at the bottom shows the output of the 'dotnet run' command:

```
Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\goran\source\repos\CoreApp2>dotnet run
Hello world!

C:\Users\goran\source\repos\CoreApp2>
```

37

Pitanje 1

Kada se kompajlira C# source kod pisan u Visual Studio okruženju za .NET framework platformu dobija se:

- a. binarni kod
- b. MSIL kod
- c. JIT kod

Odgovor: b

38

Pitanje 2

- Za ispis jedne linije teksta na konzoli koristi se:
- a. ReadLine() metoda klase Console
 - b. WriteLine() metoda klase Main
 - c. WriteLine() metoda klase Console

Odgovor: c

39

Pitanje 3

- Izvršavanje C# konzolne aplikacije počinje izvršavanjem:
- a. go() metode
 - b. Main() metode
 - c. start() metode

Odgovor: b

40

Pitanje 4

Metoda ReadLine() klase Console, kada se izvrši, vraća:

- a. prvo slovo teksta koga je uneo korisnik
- b. ASCII kod karaktera ENTER
- c. liniju teksta koju je uneo korisnik

Odgovor: c

41

Pitanje 5

ReadLine() metoda klase Console izvršava se:

- a. kada korisnik pritisne taster ENTER
- b. kada korisnik pritisne taster TAB
- c. kada korisnik pritisne bilo koji taster na tastaturi

Odgovor: a

42

Enumeracije

Enumeracije

- Enumeracioni tip specificira skup imenovanih konstanti
- Prednosti korišćenja enumeracija:
 - Lakše je održavanje koda jer se promenljivama dodeljuju samo očekivane vrednosti
 - Kod je lakše čitati jer se vrednostima dodeljuju imena koja je lako identifikovati
 - Lakše je pisanje koda jer IntelliSense prikazuje listu mogućih vrednosti koju možemo koristiti

Enumercijski tip

```
class Program
{
    enum Dan
    {
        Ponedeljak,
        Utorak,
        Sreda,
        Cetvrtak,
        Petak,
        Subota,
        Nedelja
    }
    static void Main(string[] args)
    {
        Dan d = Dan.Petak;
        Console.WriteLine(d);

        int n = (int)d;
        Console.WriteLine(n);
        Console.ReadLine();
    }
}
```

Definiše se unutar klase ili unutar namespace-a.
Ne unutar metode.

45

Enumercijski tip

```
enum Dan
{
    Ponedeljak=1,
    Utorak,
    Sreda,
    Cetvrtak,
    Petak,
    Subota,
    Nedelja
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("Unesite redni broj dana u nedelji 1-7");
    int redniBroj =int.Parse(Console.ReadLine());
    Dan unetiDan = (Dan)redniBroj;
    Console.WriteLine(unetiDan);
    Console.ReadLine();
}
```

46

Pitanje 1

Enumeracioni tip podataka se ne može definisati unutar:

- a. klase
- b. namespace
- c. metode

Odgovor: c

47

Pitanje 2

Šta se ispisuje kao rezultat izvršavanja Main funkcije:

```
enum Doba
{
    Prolece,
    Leto,
    Jesen,
    Zima
}
static void Main(string[] args)
{
    Console.WriteLine((int)Doba.Leto);
    Console.ReadLine();
}

a. 0
b. 1
c. 2
```

Odgovor: b

48

Pitanje 3

Šta se ispisuje kao rezultat izvršavanja Main funkcije:

```
enum Doba
{
    Prolece,
    Leto,
    Jesen,
    Zima
}
static void Main(string[] args)
{
    Doba d = (Doba)1;
    Console.WriteLine(d);
    Console.ReadLine();
}

a. 0
b. Leto
c. 1
```

Odgovor: b

49

Definisanje klase i kreiranje objekata

Osnovni koncepti OOP-a

- Apstraktni tipovi podataka
- Enkapsulacija
- Nasleđivanje
- Polimorfizam

51

Apstraktni tipovi podataka

- Ugrađeni (osnovni, primitivni tipovi podataka) npr. float, int, double, object, ...
- Ravnopravno se definišu korisnički definisani tipovi – apstraktni tipovi podataka: TekuciRacun, Osoba, Student, ...
- Proizvoljan broj primeraka nekog tipa i mogu se vršiti operacije nad njima

52

Enkapsulacija

- Enkapsulacija je sposobnost objekta da skriva svoje unutrašnje podatke i implementacione detalje
- Grupisanje podataka i koda koji manipuliše podacima
- Enkapsulacija se ostvaruje korišćenjem klase kao novog tipa podataka
- Realizija nekog tipa podataka može i treba da se sakrije od ostatka sistema tj. onih koji ga koriste
- Korisnicima se definiše šta se sa tipom može uraditi a način na koji se to radi se skriva



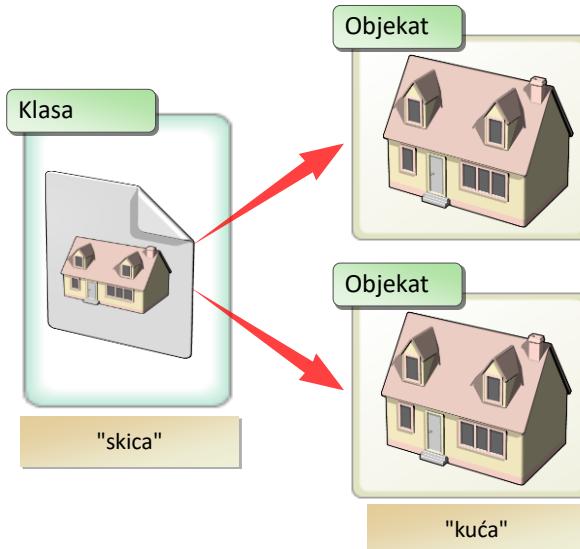
53

Odnos klase i objekata

- Klasa :
 - To je model koji opisuje kako kreirati objekat
 - je kao "šematski plan(skica)"
 - Sadrži podatke (polja) i metode
- Objekti:
 - Objekat je predstava nekog entiteta iz realnog sveta
 - Instance klase
 - Može biti više objekata (instanci) klase

54

Primer klase i objekata



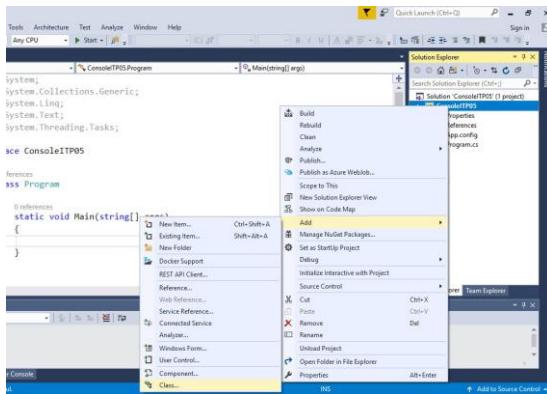
55

Funkcionalnosti enkapsulirane unutar klase



56

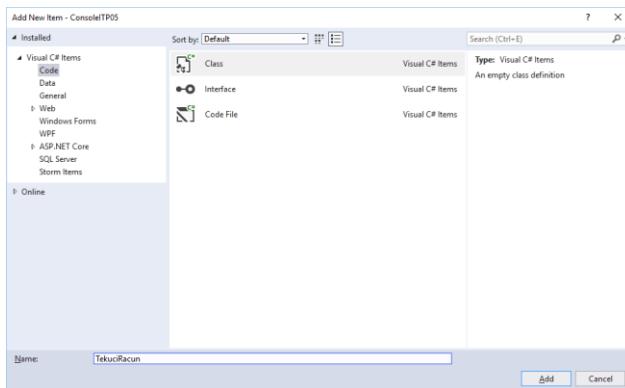
Dodavanje klase u Visual Studio 2017 okruženju



Desni klik na naziv projekta

57

Dodavanje klase u Visual Studio 2017 okruženju



58

Definisanje klase

```
class TekuciRacun
{
    public string ime;
    public string prezime;
    public double stanje;
}
```

59

Kreiranje objekata

- Objekti su inicijalno neoznačeni
- Podrazumevana vrednost objekta je null
- Da bi se koristila promenljiva tipa klase klasa se mora instancirati
- Nova instanca klase kreira se korišćenjem operatora **new**

60

Kreiranje objekata

- Operator new prouzrokuje da:
 - CLR alocira memoriju za objekat
 - poziva konstruktor da inicijalizuje objekat
- Da bi se pristupilo javnom članu klase koristi se ime instance iza koga sledi operator tačka

61

Instanciranje klase (kreiranje objekata)

```
static void Main(string[] args)
{
    TekuciRacun tr1 = new TekuciRacun();
    tr1.ime = "Pera";
    tr1.prezime = "Peric";
    tr1.stanje = 12345.34;
    Console.WriteLine($"Korisnik {tr1.ime} {tr1.prezime} ima
{tr1.stanje} dinara na racunu.");
    Console.ReadLine();
}
```

62

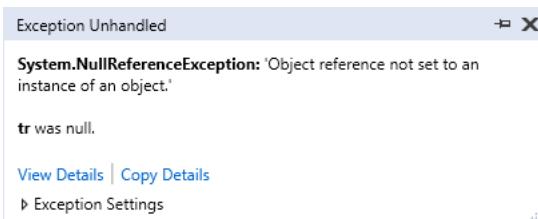
Dodeljivanje null reference

```
public static void Stampaj(TekuciRacun tr)
{
    Console.WriteLine($"Korisnik {tr.ime} {tr.prezime} ima {tr.stanje} dinara
na racunu.");
}

static void Main(string[] args)
{
    TekuciRacun tr1 = new TekuciRacun();
    tr1.ime = "Pera";
    tr1.prezime = "Peric";
    tr1.stanje = 12345.34;
    Stampaj(tr1);
    tr1 = null;
    Stampaj(tr1);
    Console.ReadLine();
}
```

63

Generisanje izuzetka od strane okruženja



64

Metode klase

- Metode klase predstavljaju funkcije – članice klase.
- Svaka metoda sadrži :
 - tip podataka koga metoda vraća (ili void ukoliko ne vraća podatke)
 - naziv(ime metode)
 - listu parametara
 - telo metode
- Ukoliko metoda vraća neku vrednost, onda se unutar tela metode mora pozvati naredba **return**.

```
public T nazivMetode(T1 param1, ..., TN paramN)
{
    // telo metode
    return rezultat;
}
```

65

Primeri metoda

```
public double Uplata(double iznos)
{
    stanje += iznos;
    return stanje;
}
```

```
public void Uplata(double iznos)
{
    stanje += iznos;
}
```

```
public void Isplata(double iznos)
{
    stanje -=iznos;
}
```

66

Klasa sa poljima i metodama

```
class TekuciRacun
{
    public string ime;
    public string prezime;
    public double stanje;

    public void Uplata(double iznos)
    {
        stanje += iznos;
    }

    public void Isplata(double iznos)
    {
        stanje -=iznos;
    }
}
```

67

Pozivanje metoda klase

```
static void Main(string[] args)
{
    TekuciRacun tr1 = new TekuciRacun();
    tr1.ime = "Pera";
    tr1.prezime = "Peric";
    tr1.stanje = 12345.34;
    Stampaj(tr1);

    tr1.Uplata(45678.12);
    Stampaj(tr1);

    tr1.Isplata(12345.45);
    Stampaj(tr1);
    Console.ReadLine();
}
```

68

Klasa sa podrazumevanim konstruktorom

```
class TekuciRacun
{
    public string ime;
    public string prezime;
    public double stanje;

    public void Uplata(double iznos)
    {
        stanje += iznos;
    }

    public void Isplata(double iznos)
    {
        stanje -= iznos;
    }
}
```

69

Definisanje parametarskog konstruktora

```
public TekuciRacun(string ime1, string prezime1, double stanje1)
{
    ime = ime1;
    prezime = prezime1;
    stanje = stanje1;
}
```

```
public TekuciRacun(string ime, string prezime, double stanje)
{
    this.ime = ime;
    this.prezime = prezime;
    this.stanje = stanje;
}
```

70

Klase sa parametarskim konstruktorom

```
class TekuciRacun
{
    public string ime;
    public string prezime;
    public double stanje;

    public TekuciRacun(string ime, string prezime, double stanje)
    {
        this.ime = ime;
        this.prezime = prezime;
        this.stanje = stanje;
    }

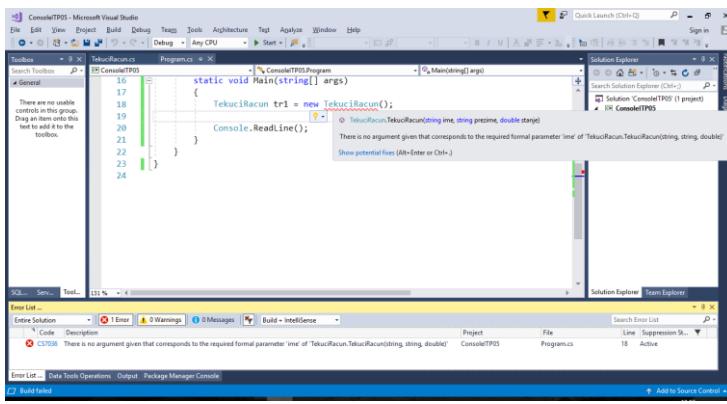
    public void Uplata(double iznos)
    {
        stanje += iznos;
    }

    public void Isplata(double iznos)
    {
        stanje -= iznos;
    }
}
```

71

Poziv podrazumevanog konstruktora

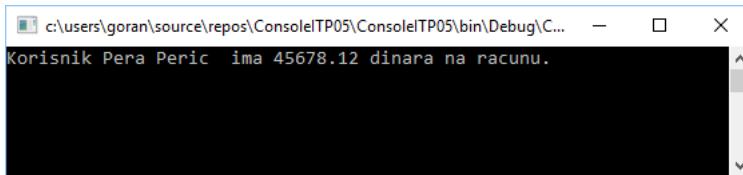
```
static void Main(string[] args)
{
    TekuciRacun tr1 = new TekuciRacun();
}
```



72

Poziv parametarskog konstruktora

```
static void Main(string[] args)
{
    TekuciRacun tr1 = new TekuciRacun("Pera", "Peric", 45678.12);
    Stampaj(tr1);
    Console.ReadLine();
}
```



73

Definisanje konstruktora bez parametara

```
public TekuciRacun()
{
}
```

74

Poziv konstruktora

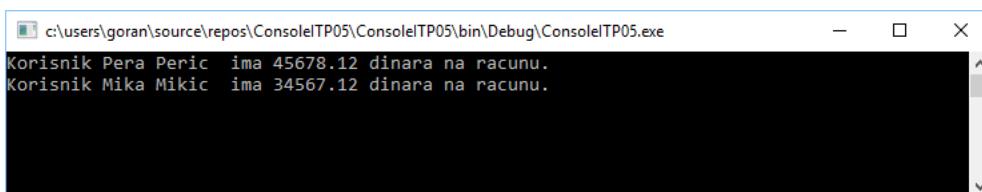
```
static void Main(string[] args)
{
    TekuciRacun tr1 = new TekuciRacun("Pera", "Peric", 45678.12);
    Stampaj(tr1);

    TekuciRacun tr2 = new TekuciRacun();
    tr2.ime = "Mika";
    tr2.prezime = "Mikic";
    tr2.stanje = 34567.12;

    Stampaj(tr2);
    Console.ReadLine();
}
```

75

Poziv različitih konstruktora - rezultat



76

Pitanje 1

Sposobnost objekta da skriva svoje podatke i implementacione detalje naziva se:

- a. abstrakcija
- b. enkapsulacija
- c. nasleđivanje

Odgovor: b

77

Pitanje 2

Klasa može sadržati:

- a. samo podatke ili polja klase
- b. samo metode ili funkcije klase
- c. i polja i metode

Odgovor: c

78

Pitanje 3

Ako metoda klase ne vraća vrednost onda je njen povratni tip:

- a. object
- b. void
- c. int

Odgovor: b

79

Pitanje 4

Neka je kreirana klasa Student i unutar nje konstruktor koji ima dva ulazna parametra.

Povratna vrednost konstruktora je tipa ?

- a. int
- b. objekat klase Student
- c. void
- d. konstruktor klase Student ne vraća nikakvu vrednost

Odgovor: d

80

Pitanje 5

- Kada se unutar klase definiše konstruktor tada nastupa sledeća situacija:
- a. klasa pored definisanog konstruktora ima i podrazumevani konstruktor
 - b. vrši se prebrisavanje podrazumevanog konstruktora definisanim konstruktorm
 - c. generiše se greška jer klasa već ima podrazumevani konstruktor

Odgovor: b

81

Pitanje 6

- Unutar klase moguće je definisati:
- a. samo jedan konstruktor
 - b. najviše dva konstruktora
 - c. neograničeni broj konstruktora

Odgovor: c

82

Prava pristupa članovima klase

Definisanje prava pristupa članovima klase

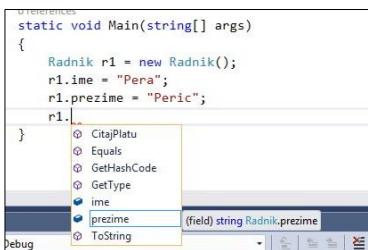
Deklaracija	Definicija
public	pristup nije ograničen
private	pristup je ograničen na klasu koja sadrži tog člana
internal	pristup je ograničen na aplikaciju
protected	Pristup je ograničen na članove klase i klase izvedenih iz te klase

Ilustracija private prava pristupa

```
class Radnik
{
    public string ime;
    public string prezime;
    private decimal plata;
}
```

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik();
    r1.ime = "Pera";
    r1.prezime = "Peric";
    r1.plata = 57898.56m;
}
```

'Radnik.plata' is inaccessible due to its protection level



85

Čitanje i setovanje privatnog člana unutar klase

```
class Radnik
{
    public string ime;
    public string prezime;
    private decimal plata;

    public decimal CitajPlatu()
    {
        return plata;
    }

    public void SetujPlatu(decimal plata)
    {
        this.plata = plata;
    }
    public void Stampaj()
    {
        Console.WriteLine("Radnik {0} {1} ima platu od {2} dinara",ime,prezime,plata);
    }
}
```

86

Indirektno setovanje privatnog člana klase

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik();
    r1.ime = "Pera";
    r1.prezime = "Peric";
    r1.SetujPlatu(345567.678m);
    r1.Stampaj();

    Console.ReadLine();
}
```

87

Svojstva (Properties)

```
class Radnik
{
    public string ime;
    public string prezime;

    private decimal plata;
    public decimal Plata
    {
        get { return plata; }
        set { plata = value; }
    }

    public void Stampaj()
    {
        Console.WriteLine("Radnik {0} {1} ima platu od {2} dinara", ime, prezime, plata);
    }
}
```

propfull Code Snippet

88

Upotreba svojstva

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik();
    r1.ime = "Pera";
    r1.prezime = "Peric";
    r1.Plata = 345567.678m; // set property

    decimal iznos = r1.Plata; // get property
    Console.WriteLine(iznos);
    Console.ReadLine();
}
```

89

Automatska svojstva

```
class Radnik
{
    public string ime;
    public string prezime;

    //automatsko svojstvo
    public decimal Plata { get; set; }

    public void Stampaj()
    {
        Console.WriteLine("Radnik {0} {1} ima platu od {2} dinara", ime, prezime, Plata);
    }
}
```

90

Definisanje klase korišćenjem automatskih svojstava

```
class Radnik
{
    public string Ime { get; set; }
    public string Prezime { get; set; }

    //automatsko svojstvo
    public decimal Plata { get; set; }

    public void Stampaj()
    {
        Console.WriteLine("Radnik {0} {1} ima platu od {2} dinara", Ime, Prezime, Plata);
    }
}
```

91

Inicijalizator objekta

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik { Ime = "Pera", Prezime = "Peric", Plata = 345678.56m };
    r1.Stampaj();
    Console.ReadLine();
}
```

92

Postavljanje privatnog set aksesora

```
class Osoba
{
    public Guid Id { get; private set; }
    public string Ime { get; set; }
    public string Prezime { get; set; }

    public Osoba()
    {
        this.Id = Guid.NewGuid();
    }
}
```

propg Code Snippet

b7ef3276-9cc6-4cef-a462-791965866b17

93

Pristup svojstvu koje može samo da se čita

```
static void Main(string[] args)
{
    Osoba os1 = new Osoba();
    os1.Ime = "Pera";
    os1.Prezime = "Peric";
    //os1.Id = 1;
    Console.WriteLine(os1.Id);
    Console.ReadLine();
}
```

94

Kreiranje anonymnog objekta

```
static void Main(string[] args)
{
    var v = new { Ime = "Pera", Prezime = "Peric" };
    Console.WriteLine(v.Ime + " " + v.Prezime);
    Console.ReadLine();
}
```

Ključna reč var ukazuje kompjleru da odredi tip na osnovu desne strane izraza za inicijalizaciju. To može biti ugrađeni tip, anonimni tip ili korisnički definisani tip podataka.

95

Pitanje 1

Svojstvo pridruženo privatnom članu klase mora imati i get i set aksesor:

- a. Da
- b. Ne

Odgovor: b

96

Pitanje 2

Definisanjem automatskog svojstvo zamenjuje se definicija:

- a. privatnog polja klase i propertija koji poseduje get i set aksesor
- b. privatnog polja i get aksesora
- c. get i set aksesora nekog privatnog polja klase

Odgovor: a

97

Vrednosni i referentni tipovi podataka

Vrednosni tipovi podataka

- Pri kopiranju vrednosnog tipa u memoriji se kreira nova promenljiva
- Promena vrednosti kod originala se ne odražava na kopiju i obratno
- Vrednosni tipovi se čuvaju na steku

```
static void Main(string[] args)
{
    int x = 10;
    int y = x;
    x++;
    Console.WriteLine("x={0}", x);
    Console.WriteLine("y={0}", y);
}
```

99

Referentni tipovi

- Kreiraju se u memoriji koja se naziva hip
- Kada promenljivoj pridružimo referencu, jednostavno joj pridružimo objekat u memoriji
- Ako dvema promenljivama pridružimo istu referencu, obe pokazuju na isti objekat
- Ako promenimo podatak u objektu, promene će se odnositi na sve promenljive koje referenciraju objekat

100

Demonstracija referentnih tipova

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }

    public void Stampaj()
    {
        Console.WriteLine(Ime + " " + Prezime);
    }
}
```

```
static void Main(string[] args)
{
    Osoba os1 = new Osoba {Ime = "Pera", Prezime="Peric" };
    os1.Stampaj();

    Osoba os2 = os1;
    os2.Ime = "Mika";

    os1.Stampaj();
    Console.ReadLine();
}
```

101

Nullable tip

- Null vrednost se koristi za inicijalizaciju referentnog tipa podataka i ne može se dodeliti vrednosnom tipu
- Kada promenljivoj dodelimo null referencu znači da ona nije inicijalizovana (tj. ne pokazuje ni na šta)
- Nije dozvoljeno `int x = null;`
- C# definiše modifikator koji se koristi da se promenljiva definiše kao nullable vrednost
- Koristi se modifikator ? koji naznačava da vrednosni tip može biti nullable
- Nullable tip se ponaša kao originalni vrednosni tip ali mu se može pridružiti null vrednost
- Dozvoljeno `int? x = null;`
- Nullable tipovi imaju HasValue i Value svojstva

102

Upotreba nullable tipa

```
static void Main(string[] args)
{
    int? x2 = null;
    Console.WriteLine("Unesite broj x1");

    if (int.TryParse(Console.ReadLine(), out int x1))
    {
        x2 = x1;
    }

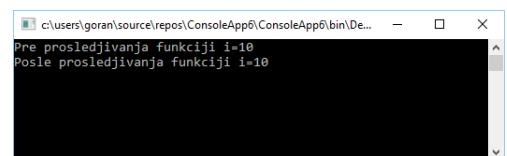
    if (x2.HasValue)
    {
        Console.WriteLine(x2.Value);
    }
    else
    {
        Console.WriteLine("X2 nije definisano");
    }

    Console.WriteLine("Pritisni ENTER za izlazak");
    Console.ReadLine();
}
```

103

Prenos parametara po vrednosti

```
public static void Promeni(int a)
{
    a++;
}
```



```
static void Main(string[] args)
{
    int i = 10;
    Console.WriteLine($"Pre prosledjivanja funkciji i={i}");
    Promeni(i);
    Console.WriteLine($"Posle prosledjivanja funkciji i={i}");
    Console.ReadLine();
}
```

104

Prenos vrednosnih tipova po referenci

```
public static void Promeni(ref int a)
{
    a++;
}
```

```
Pre prosledjivanja funkciji i=10
Posle prosledjivanja funkciji i=11
```

```
static void Main(string[] args)
{
    int i = 10;
    Console.WriteLine($"Pre prosledjivanja funkciji i={i}");
    Promeni(ref i);
    Console.WriteLine($"Posle prosledjivanja funkciji i={i}");
    Console.ReadLine();
}
```

105

Referenca kao ulazni parameter metode

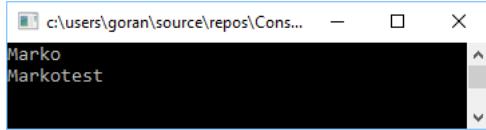
```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
}
```

106

Referenca kao ulazni parameter metode

```
static void PromeniIme(Osoba os)
{
    os.Ime += "test";
}
```

```
static void Main(string[] args)
{
    Osoba os1 = new Osoba { Ime = "Marko", Prezime = "Markovic" };
    Console.WriteLine(os1.Ime);
    PromeniIme(os1);
    Console.WriteLine(os1.Ime);
    Console.ReadLine();
}
```



107

Strukture

Strukture

- Strukture su vrednosni tipovi podataka
- Podaci iz strukture se čuvaju na steku
- Koriste se za modelovanje stavki koje sadrže manje količine podataka
- Strukture mogu da sadrže polja i metode kao i klase
- Ne mogu se definisati sopstvene hijerarhije nasleđivanja između struktura
- Ne mogu se definisati strukture koje proizilaze iz klasa ili drugih struktura

109

Sistemske strukture

System.Byte	System.Int64	System.Decimal
System.Int16	System.Single	System.Boolean
System.Int32	System.Double	System.Char

```
static void Main(string[] args)
{
    Console.WriteLine("Najveci ceo broj je {0}", Int32.MaxValue);
    Console.WriteLine("Najmanji ceo broj je {0}", Int32.MinValue);

    string s = "1234";
    int i = Int32.Parse(s);
    i++;
    Console.WriteLine(i);

    Console.ReadKey();
}
```

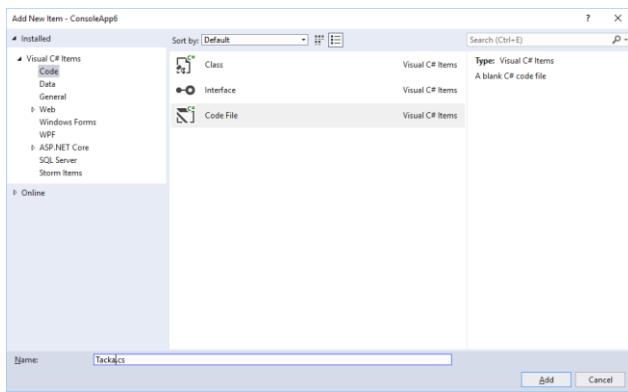
110

Osobine strukture

- Ne može se definisati konstruktor bez parametara za strukturu jer za razliku od klase kompjajler generiše sopstveni default konstruktor za strukturu
- Svaki konstruktor mora eksplisitno inicijalizovati svako polje u strukturi
- Nije dozvoljena inicijalizacija polja pri samoj deklaraciji
- Pri kreiranju promenljive tipa strukture nije neophodno je navoditi ključnu reč **new**
- Moguće je deklarisati nullable promenljivu tipa strukture

111

Dodavanje cs fajla



112

Primer strukture

```
struct Tacka
{
    public int x;
    public int y;

    public Tacka(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

113

Program.cs

```
static void Stampaj(Tacka t)
{
    Console.WriteLine($"Koordinate tacke su: ({t.x},{t.y})");
}
```

114

Inicijalizacija strukture

```
static void Main(string[] args)
{
    // Deklarisi objekat
    Tacka t1;
    // Inicijalizacija
    t1.x = 10;
    t1.y = 20;

    Stampaj(t1);

    Tacka t2 = new Tacka();
    t2.x = 15;
    t2.y = 20;
    Stampaj(t2);

    Tacka t3 = new Tacka(30, 40);
    Stampaj(t3);
    Console.ReadLine();
}
```

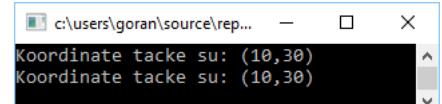
Ne mora se koristiti new operator da bi se kreirala instanca strukture

Operator new samo inicijalizuje polja u strukturi
Uvek koristiti konstruktor da bi se osigurala inicijalizacija polja u strukturi

115

Prosleđivanje struktturnog tipa metodi

```
public static void PromeniTacku(Tacka t)
{
    t.x += 1;
    t.y += 1;
}
```



```
static void Main(string[] args)
{
    Tacka t1 = new Tacka(10, 30);
    Stampaj(t1);
    PromeniTacku(t1);
    Stampaj(t1);
    Console.ReadLine();
}
```

116

Pitanje 1

Struktura predstavlja:

- a. Vrednosni tip podataka
- b. Referentni tip podataka
- c. I vrednosni i referentni tip podataka

Odgovor: a

117

Pitanje 2

Da li je moguće kreirati novi strukturni tip podataka na osnovu prethodno definisane strukture od strane korisnika

- a. da
- b. ne

Odgovor: b

118

Statički članovi klase

Statički članovi klase

- Pripadaju klasi a ne instanci klase
- Pristupa im se preko imena klase
- I metode i polja i svojstva klase mogu biti statički
- Pozivaju se bez kreiranja objekata klase
- Statičke metode i svojstva ne mogu pristupati ne - statičkim poljima u klasi u kojoj se definisu

Statički članovi klase

- Statičko polje se često koristi da čuva broj kreiranih objekata klase
- Statičko polje se koristi i za deljenje vrednosti između različitih instanci te klase
- Statički član klase se definiše korišćenjem ključne reči static pre povratnog tipa

121

Primer definisanja statičkog polja

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public int Starost { get; set; }

    public static int brojOsoba = 0;

    public Osoba()
    {
        brojOsoba++;
    }
}
```

122

Upotreba statičkog polja

```
static void Main(string[] args)
{
    Console.WriteLine($"Pocetni broj osoba je: {Osoba.brojOsoba}");
    Osoba os1 = new Osoba();
    Osoba os2 = new Osoba();
    Console.WriteLine($"Trenutni broj osoba je: {Osoba.brojOsoba}");
    Osoba.brojOsoba = 10;
    Console.WriteLine($"Trenutni broj osoba je: {Osoba.brojOsoba}");
    Console.ReadLine();
}
```

123

Definisanje statičke metode

```
class Pravougaonik
{
    private double sirina;
    private double visina;
    public Pravougaonik(double sirina, double visina)
    {
        this.sirina = sirina;
        this.visina = visina;
    }

    public double Povrsina()
    {
        return sirina * visina;
    }

    public static double RacunajPovrsinu(double a, double b)
    {
        return a * b;
    }
}
```

124

Poziv statičke metode

```
static void Main(string[] args)
{
    //Upotreba nestaticke metode
    Pravougaonik pr1 = new Pravougaonik(12.3, 45.6);
    double P1 = pr1.Povrsina();
    Console.WriteLine("P1= " + P1);

    //Upotreba staticke metode
    double P2 = Pravougaonik.RacunajPovrsinu(12.3, 45.6);
    Console.WriteLine("P2= " + P2);
    Console.ReadLine();
}
```

125

Statička klasa

- Ne može se instancirati
- Statičke klase sadrže samo statičke članove
- Koristi se kao kontejner za skup metoda koje rade sa ulaznim parametrima
- Klasa Math iz prostora imena sistem sadrži statičke metode za izvršavanje matematičkih operacija

126

Statička klasa Math

```
public static class Math
```

```
static void Main(string[] args)
{
    double x = Math.PI;
    double pi = Math.Round(x, 2);
    Console.WriteLine("pi= " + pi);
    Console.ReadLine();
}
```

127

Statički konstruktor

- Klasa može sadržati statički konstruktor koji se koristi za inicijalizaciju statičkih članova
- I statička i ne statička klasa mogu sadržati statički konstruktor
- Statički konstruktor nema modifikator pristupa i nema parametre
- Statički konstruktor se poziva automatski pre kreiranja instanci i pre pristupa bilo kom statičkom članu
- Statički konstruktor se poziva jednom pre poziva bilo kog instans konstruktora

128

Statičko svojstvo

```
class Radnik
{
    public static int brojRadnika;
    private static int brojac;

    public string Ime { get; set; }
    public string Prezime { get; set; }

    // staticki konstruktor
    static Radnik()
    {
        brojRadnika = 10;
        brojac = 0;
    }
    // staticko read only svojstvo
    public static int BrojRadnika
    {
        get { return brojac + brojRadnika; }
    }
    public Radnik()
    {

        brojac++;
    }
}
```

129

Poziv statičkog svojstva

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik {Ime="Marko", Prezime="Markovic" };
    Console.WriteLine(Radnik.BrojRadnika);
    Radnik r2 = new Radnik { Ime = "Milan", Prezime = "Petrovic" };
    Console.WriteLine(Radnik.BrojRadnika);
    Console.ReadLine();
}
```

130

Pitanje 1

Javnom statičkom polje klase A se u nekoj metodi klase B

- a. može pristupiti pre instanciranja klase A
- b. ne može pristupiti pre instanciranja klase A
- c. može pristupiti samo nakon instanciranja klase B

Odgovor: a

131

Pitanje 2

Statička klasa može da sadrži jedno nestatičko polje?

- a. Da
- b. Ne

Odgovor: b

132

Pitanje 3

Da li je moguće upotrebiti ključnu reč this unutar statičke metode:

- a. Da
- b. Ne

Odgovor: b

133

Pitanje 4

Kreirana je klasa Student koja modeluje Studente nekog fakulteta.

Da li polje Ime klase Student može biti statičko:

- a. Da
- b. Ne

Odgovor: b

134

Nasleđivanje i polimorfizam

Nasleđivanje

- Nasleđivanje omogućava da se kreiraju novi tipovi podataka na osnovu već postojećih tipova
- Nasleđivanje je vrsta relacije među osnovne(bazne) klase i izvedenih klasa
 - Izvedena klasa nasleđuje sva polja i metode osnovne klase
 - Izvedena klasa ima članove svojstvene samo izvedenoj klasi
 - Izvedena klasa postaje više specijalizovana

Nasleđivanje-pojmovi

- Osnovna klasa se naziva još i natklasa, bazna klasa, roditeljska klasa (parent class), super klasa
- Izvedena klasa se naziva još klasa potomak (child class) i podklasa
- Izvedena klasa može da nasledi samo jednu baznu klasu
- Nasleđivanje redukuje ponavljanje koda
- public class B : A { telo klase B }
//(klasa B izvedena iz klase A)
- public sealed class NekaKlasa {telo klase} – tada se iz ovakve klase ne može vršiti nasleđivanje
- Vrednosni tipovi su sealed implicitno tj. ne mogu se nasleđivati

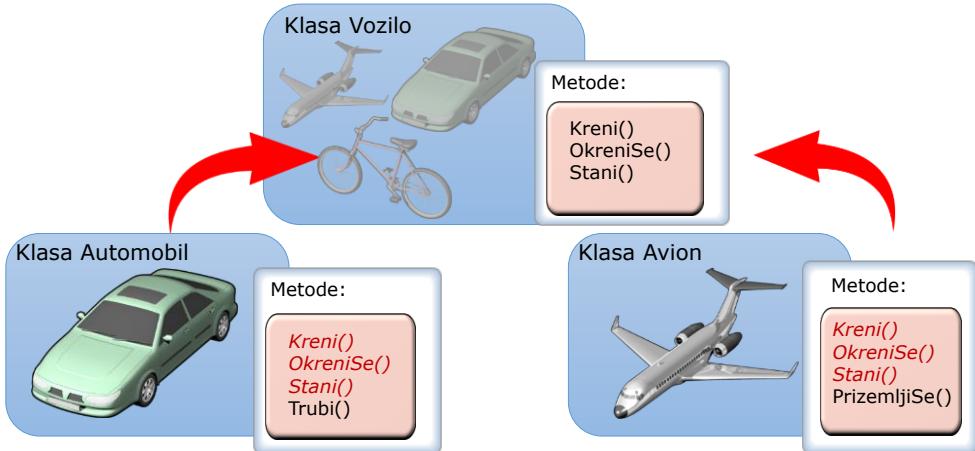
137

Vidljivost članova

- Privatni članovi bazne klase nisu vidljivi u izvedenoj klasi osim ako izvedena klasa nije ugnježđena u baznu klasu
- Protected članovi klase su vidljivi u izvedenoj klasi
- Public članovi klase su vidljivi u izvedenoj klasi

138

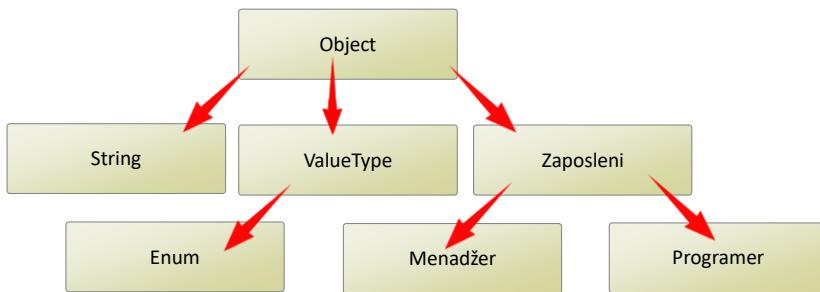
Ilustracija nasleđivanja



139

Nasleđivanje u .NET Frameworku

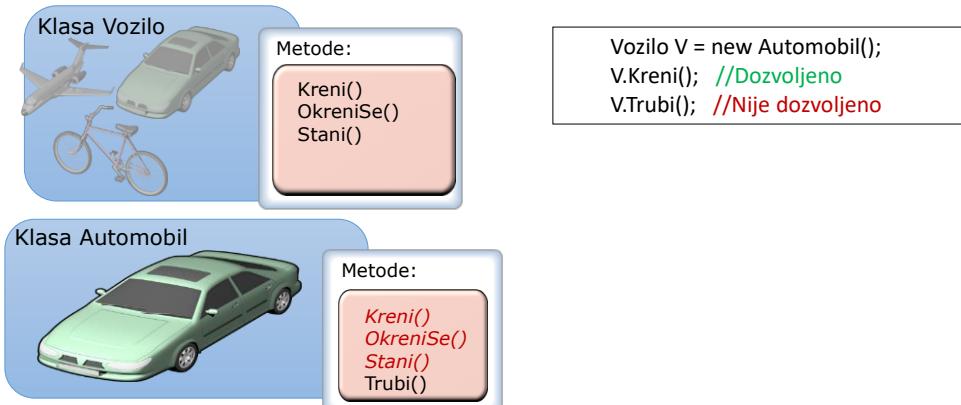
- Svi tipovi u .NET frameworku su nasleđeni direktno ili indirektno iz klase `System.Object`



140

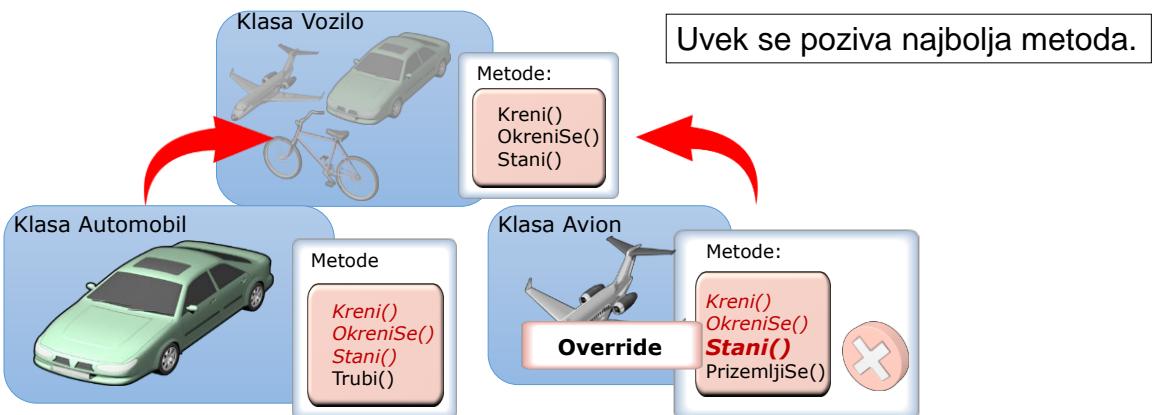
Prvo pravilo polimorfizma

Referenca tipa bazne klase može pokazivati na objekat izvedene klase



141

Drugo pravilo polimorfizma



142

Virtuelne metode

- Kada se kreira metoda u baznoj klasi za koju se očekuje da će biti promenjena u izvedenoj klasi, metoda se definiše kao virtuelna (**virtual**).
- Metoda u izvedenoj klasi koja ima isto ime kao i virtuelna metoda u baznoj klasi vrši "prebrisavanje" (**override**) metode iz bazne klase

143

Definisanje virtuelne metode u baznoj klasi

```
class Oblik
{
    public virtual void Crtanje()
    {
        Console.WriteLine("Genericka metoda za crtanje");
    }
}
```

144

Prebrisavanje virtuelne metode u izvedenoj klasi

```
class Krug : Oblik
{
    public override void Crtanje()
    {
        Console.WriteLine("Crtanje kruga");
    }
}
```

```
class Kvadrat : Oblik
{
    public override void Crtanje()
    {
        Console.WriteLine("Crtanje kvadrata");
    }
}
```

145

Ilustracija 1. i 2. pravila polimorfizma

```
static void Main(string[] args)
{
    Oblik[] oblici = new Oblik[4];
    // 1. pravilo polimorfizma
    oblici[0] = new Krug();
    oblici[1] = new Krug();
    oblici[2] = new Kvadrat();
    oblici[3] = new Kvadrat();

    foreach (Oblik obl in oblici)
    {
        // 2. pravilo polimorfizma
        obl.Crtanje();
    }
    Console.ReadLine();
}
```

146

Bazna klasa

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public Osoba(string Ime, string Prezime)
    {
        this.Ime = Ime;
        this.Prezime = Prezime;
    }

    public virtual void Stampaj()
    {
        Console.WriteLine($"Osoba: {Ime} {Prezime}");
    }
}
```

147

Izvedena klasa

```
class Student : Osoba
{
    public string Smer { get; set; }
    public Student(string Ime, string Prezime, string Smer):base(Ime,Prezime)
    {
        this.Smer = Smer;
    }

    public override void Stampaj()
    {
        base.Stampaj();
        Console.WriteLine("Smer: " + Smer);
    }
}
```

148

Primera 1 i 2 pravila polimorfizma

```
static void Main(string[] args)
{
    Student st1 = new Student("Pera", "Peric", "Informatika");
    st1.Stampaj();

    Osoba os1 = new Student("Mika", "Mikic", "Istorija");
    os1.Stampaj();

    Console.ReadLine();
}
```

149

Pitanje 1

Prvo pravilo polimorfizma glasi:

- a. Uvek se poziva najbolja metoda.
- b. Objektu bazne klase se uvek može pristupiti posredstvom reference tipa izvedene klase
- c. Objektu izvedene klase se uvek može pristupiti posredstvom reference tipa bazne klase

Odgovor: c

150

Pitanje 2

Ukoliko želimo da zabranimo nasleđivanje naše klase, onda to postižemo korišćenjem ključne reči ispred imena klase

- a. private
- b. noninheritable
- c. sealed

Odgovor c

151

Pitanje br 3

Drugo pravilo polimorfizma glasi:

- a. Uvek se poziva najbolja metoda.
- b. Objektu bazne klase se uvek može pristupiti posredstvom reference tipa izvedene klase
- c. Objektu izvedene klase se uvek može pristupiti posredstvom reference tipa bazne klase
- d. Uvek se poziva metoda bazne klase.

Odgovor: a

152

Apstraktne klase

Abstraktne klase

- Apstraktna metoda je prazna metoda tj. metoda koja nema implementaciju (nema ni prazno telo)
- Klasa koja ima bar jedan apstraktnu metodu je apstraktna klasa
- Ispred apstraktnih metoda i apstraktnih klasa piše se ključna reč **abstract**
- Abstraktna klasa ne može da se instancira
- Abstraktna klasa može imati i metode koje nisu apstraktne
- Klasa izvedena iz abstraktne klase mora da realizuje (implementira) sve abstraktne metoda
- Primer abstraktne klase je klasa `DbConnection`

Nasleđivanje abstraktne klase

```
abstract class A
{
    // apstraktna metoda
    public abstract void metoda1();
}
```

```
class B : A
{
    public override void metoda1()
    {
        throw new NotImplementedException();
    }
}
```

155

Abstraktna klasa Oblik

```
abstract class Oblik
{
    public string Ime { get; set; }

    public Oblik(string s)
    {
        Ime = s;
    }

    // apstraktna metoda
    public abstract double Povrsina();

    public override string ToString()
    {
        return $"{Ime} : Povrsina: {Povrsina()}";
    }
}
```

156

Klasa Kvadrat

```
class Kvadrat : Oblik
{
    private double a;
    public Kvadrat(string naziv, double a):base(naziv)
    {
        this.a = a;
    }
    public override double Povrsina()
    {
        return a * a;
    }
}
```

157

Klasa Pravougaonik

```
class Pravougaonik : Oblik
{
    private double a;
    private double b;
    public override double Povrsina()
    {
        return a * b;
    }
    public Pravougaonik(string naziv, double a, double b): base(naziv)
    {
        this.a = a;
        this.b = b;
    }
}
```

158

Ilustracija pravila polimorfizma

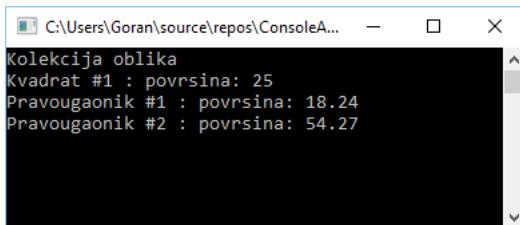
```
static void Main(string[] args)
{
    Oblik[] oblici ={
        new Kvadrat("Kvadrat #1",5),
        new Pravougaonik("Pravougaonik #1",3.2, 5.7),
        new Pravougaonik("Pravougaonik #2",6.7,8.1)
    };

    Console.WriteLine("Kolekcija oblika");
    foreach (Oblik s in oblici)
    {
        Console.WriteLine(s.ToString());
    }

    Console.ReadLine();
}
```

159

Rezultat izvršavanja koda



```
C:\Users\Goran\source\repos\ConsoleA...
Kolekcija oblika
Kvadrat #1 : povrsina: 25
Pravougaonik #1 : povrsina: 18.24
Pravougaonik #2 : povrsina: 54.27
```

160

Interfejsi

Pojam interfejsa

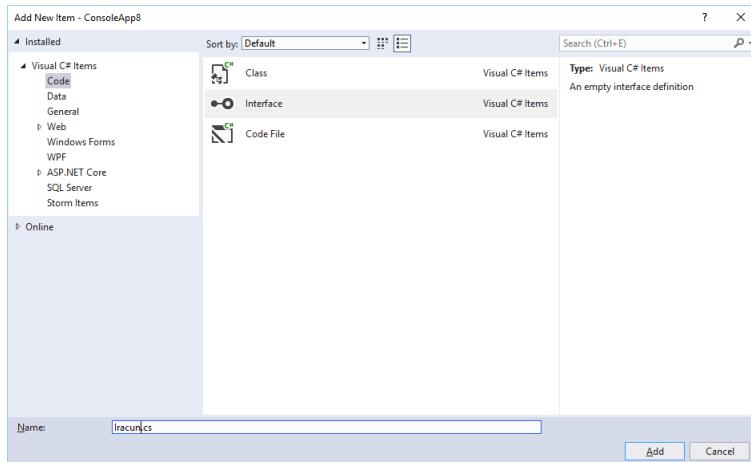
- Interfejs je koncept koji razdvaja specifikaciju metode od njene implementacije
- Interfejs je kao ugovor
 - Onaj ko objavljuje interfejs se obavezuje da neće da ga menja
 - Onaj ko implementira interfejs se obavezuje da implementira sve što je u njemu definisano
- Svaka metoda u interfejsu ima samo specifikaciju bez implementacije
- Interfejs se ne može instancirati

Karakteristike interfejsa

- Različite klase mogu implementirati interfejs na različite načine
- Koristi se ključna reč ***interface*** pri definiciji interfejsa
- Interfejs može da implementira više interfejsa
- Klasa može da implementira više interfejsa
- Metoda koja realizuje metodu iz interfejsa mora biti javna
- Upotrebom interfejsa se smanjuje velika zavisnost između klase i koda koji poziva klase
- Primer interfjesa IDbConnection

163

Dodavanje interfejsa u VisualStudio 2017 okruženju



164

Primer interfejsa

```
interface Iracun
{
    void Uplata(double iznos);
    bool Isplata(double iznos);
    double Stanje { get; }
}
```

165

Implementacija interfejsa -1

```
class Racun : Iracun
{
    private string ime;
    private string prezime;
    private double stanje;

    public Racun(string ime, string prezime, double stanje)
    {
        this.ime = ime;
        this.prezime = prezime;
        this.stanje = stanje;
    }
    public double Stanje
    {
        get
        {
            return stanje;
        }
    }
}
```

166

Implementacija interfejsa -2

```
public bool Isplata(double iznos)
{
    if (iznos <= stanje)
    {
        stanje -= iznos;
        return true;
    }
    return false;
}

public void Uplata(double iznos)
{
    stanje += iznos;
}

public void Stampaj()
{
    Console.WriteLine($"{ime} {prezime}, Stanje: {stanje}");
}
```

167

Pristup objektu klase posredstvom interfejsa

```
Racun r1 = new Racun("Marko", "Markovic", 34567);
r1.Uplata(23456);
Console.WriteLine(r1.Stanje);
bool b1 = r1.Isplata(56789);
if (!b1)
{
    Console.WriteLine("Nemate dovoljno novca na racunu");
}
Console.WriteLine(r1.Stanje);
Console.WriteLine("*****");
```

168

Pristup objektu klase posredstvom reference

```
Racun r2 = new Racun("Jovan", "Markovic", 12345);  
r2.Stampaj();  
r2.Uplata(13456.1);  
r2.Stampaj();
```

169

Eksplisitna implementacija interfejsa

```
double Iracun.Stanje  
{  
    get  
    {  
        return stanje;  
    }  
}  
  
bool Iracun.Isplata(double iznos)  
{  
    if (iznos <= stanje)  
    {  
        stanje -= iznos;  
        return true;  
    }  
    return false;  
}  
  
void Iracun.Uplata(double iznos)  
{  
    stanje += iznos;  
}
```

170

Poziv ekplicitno implementirane metode interfejsa

```
static void Main(string[] args)
{
    Racun1 r1 = new Racun1("Petar", "Markovic", 12345);
    //r1.Uplata(); // ne moze

    Ircun r2 = new Racun1("Petar", "Markovic", 12345);
    r2.Uplata(45678.1);
}
```

171

Sortiranje niza objekata

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
}

static void Main(string[] args)
{
    Osoba[] nizOsoba =
        new Osoba { Ime = "Pera", Prezime = "Peric" },
        new Osoba { Ime = "Mika", Prezime = "Mikic" },
        new Osoba { Ime = "Laza", Prezime = "Lazic" }
    };
    Array.Sort(nizOsoba);

    foreach (Osoba os in nizOsoba)
    {
        Console.WriteLine(os.Ime + " " + os.Prezime);
    }
    Console.ReadLine();
}
```

172

Implementacija IComparable interfejsa

```
class Osoba : IComparable
{
    public string Ime { get; set; }
    public string Prezime { get; set; }

    public int CompareTo(object obj)
    {
        Osoba os = (Osoba)obj;
        // -1 this prethodi objektu os
        // 1 this sledi iza objekta os
        // 0 this je na istom mestu kao i os

        if (Ime.CompareTo(os.Ime) == -1)
        {
            return -1;
        }
        else if (Ime.CompareTo(os.Ime) == 1)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
}
```

173

Pitanje 1

Abstraktna metoda klase :

- a. nema telo
- b. ima prazno telo
- c. ima telo u kome se može nalaziti kod

Odgovor: a

174

Pitanje 2

Klasa **Racun** ima konstruktor bez parametara i implementira interfejs **Iracun**. Šta je dozvoljeno pisati:

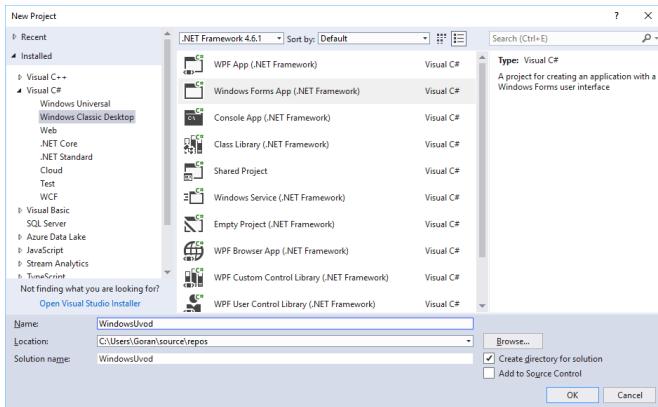
- a. Iracun i = new Iracun();
- b. Iracun i = new Racun();
- c. Racun r = new Iracun();

Odgovor: b

175

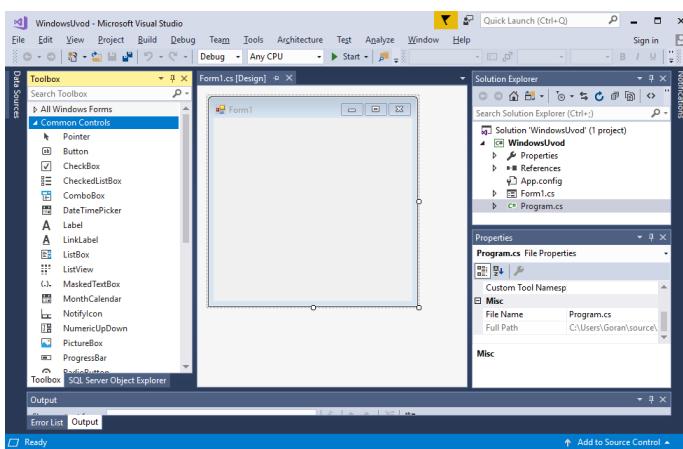
Windows forme

Kreiranje WindowsForms aplikacije



177

Dizajner WindowsForms aplikacije



178

Pojam windows forme

- Forma je osnovni elemenat korisničkog interfejsa
- Forma je kontrola izvedena iz klase **Form** a koja je izvedena iz klase **ContainerControl**
- Windows forme se kreiraju u slučajevima kreiranja desktop aplikacija i kada se очekuje da klijentski računari imaju dovoljno snage da izvrše odgovarajuću obradu podataka

```
public class Form : System.Windows.Forms.ContainerControl
```

179

Klasa Form1

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

180

Drugi deo parcijalne klase Form1

```

namespace WindowsUvod
{
    partial class Form1
    {
        /// <summary> Required designer variable.
        private System.ComponentModel.IContainer components = null;

        /// <summary> Clean up any resources being used.
        protected override void Dispose(bool disposing)...

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }
    }
}

```

181

Klasa program

```

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

```

182

Svojstva windows forme -1

- Name – ovo svojstvo postavlje ime forme odnosno ime odgovarajuće klase
public partial class Form1 : Form
- AcceptButton – pomoću ovog propertija se postavlja koje će dugme biti kliknuto kada korisnik pritisne taster ENTER
- CancelButton – ovaj properti određuje koje će dugme biti kliknuto kada se pritisne ESC taster
- ControlBox – određuje da li forma sadrži dugmad za minimizaciju, maksimizaciju i za zatvaranje prozora

183

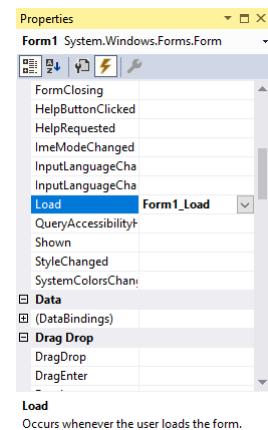
Svojstva windows forme -2

- MinimizeBox, MaximizeBox svojstva omogućavaju prikazivanje i sakrivanje dugmadi za maksimiziranje i minimiziranje
- Text property određuje tekst koji će biti prikazan na naslovnoj liniji forme
- StartPosition određuje početnu poziciju forme na ekranu
- WindowState svojstvo omogućava prikaz forme u normalnoj veličini, minimiziranu i maksimiziranu

184

Primer upotrebe Load događaja forme

```
private void Form1_Load(object sender, EventArgs e)
{
    Text = "Naslov forme";
    BackColor = Color.LightBlue;
}
```



185

Kontrola Label

- Koristi se za predstavljenje opisnog teksta korisniku
- Obično se koristi u kombinaciji sa kontrolama za unos i editovanje teksta
- Text property definiše tekst koji će se pojaviti na labeli
- Korisnik ne može menjati tekst na labeli
- Labela ne može da dobije fokus

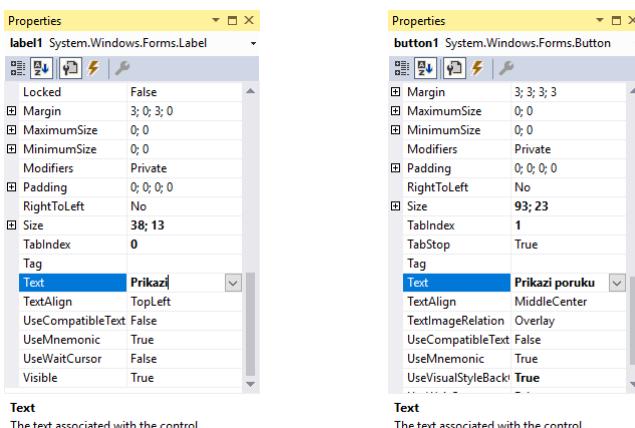
186

Kontrola Button

- Na dugme se može kliknuti mišem, pritiskom na taster **ENTER**, pritiskom na taster **ESC** ili pritiskom na **SPACE** taster (kada dugme ima fokus)
- Ako se kao svojstvo **AcceptButton** (CancelButton) forme postavi odgovarajuće dugme tada se pritiskom na taster **ENTER** (Esc) izvršava klik na to dugme
- Može se koristiti **DialogResult** property dugmeta da bi se specificirala povratna vrednost ShowDialog() metode

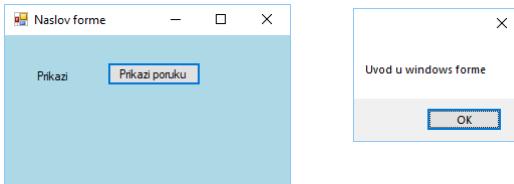
187

Podešavanje svojstava kontrola u Properties prozoru



188

Uvodna windows forms aplikacija



```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Uvod u windows forme");
}
```

189

TextBox kontrola -1

- Koristi se za prihvatanje ulaza od strane korisnika ili za prikaz vrednosti
- Svojstvo Text se koristi za čitanje sadržaja tekst boksa ili prikaz sadržaja u tekstu boksu
- Metoda Clear() briše sadržaj ove kontrole
- Metoda Focus() postavlja fokus u određenu TextBox kontrolu
- Svojstvo MaxLength – definisanje maksimalnog broja karaktera za TextBox

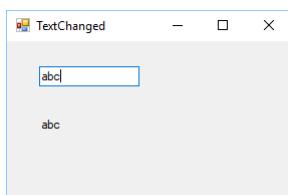
190

TextBox kontrola -2

- Svojstvo PasswordChar omogućava da se prikazuje određeni znak kada korisnik unosi tekst
- Svojstvo Enabled postavljeno na false omogućava da se kreira read-only kontrola
- Podrazumevani događaj ove kontrole je TextChanged koji se generiše svaki put kada se promeni tekst u TextBox kontroli
- Događaj KeyDown generiše se kada se pritisne neki taster na tastaturi pre nego što se otpusti

191

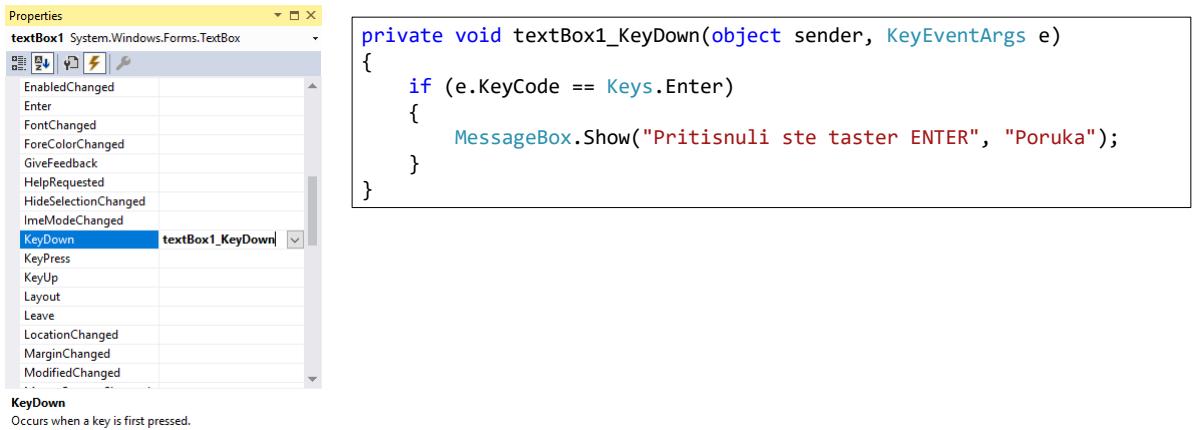
Obrada događaja TextChanged



```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label1.Text = textBox1.Text;
}
```

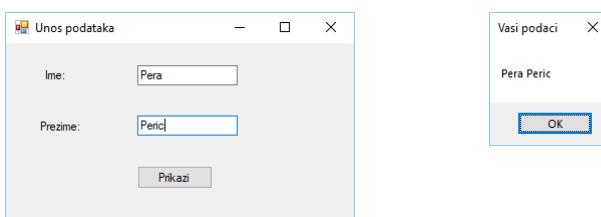
192

Obrada događaja KeyDown



193

Primer upotrebe TextBox kontrole - GUI



194

Primer upotrebe TextBox kontrole - kod

```
private void button1_Click(object sender, EventArgs e)
{
    string ime = textBox1.Text;
    string prezime = textBox2.Text;

    MessageBox.Show(ime + " " + prezime, "Vasi podaci");
    textBox1.Clear();
    textBox2.Clear();
    textBox1.Focus();
}
```

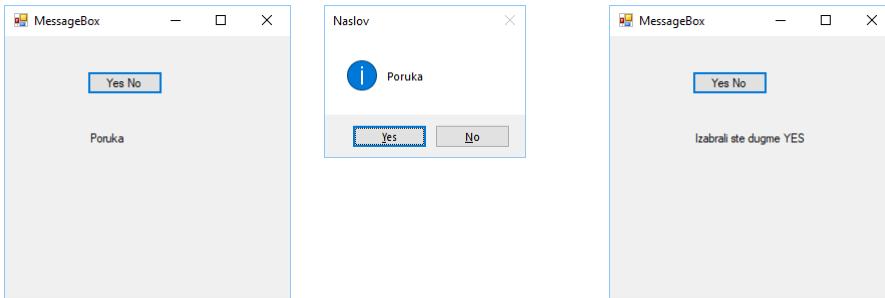
195

Show() metoda klase MessageBox

```
private void button1_Click(object sender, EventArgs e)
{
    DialogResult rez = MessageBox.Show("Poruka", "Naslov",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Information);
    switch (rez)
    {
        case DialogResult.Yes:
            label1.Text = "Izabrali ste dugme YES";
            break;
        case DialogResult.No:
            label1.Text = "Izabrali ste dugme NO";
            break;
    }
}
```

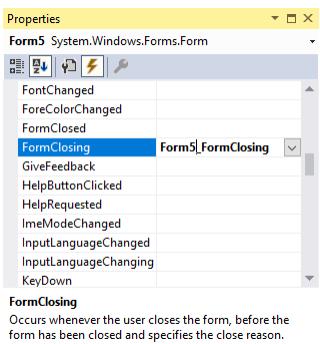
196

Show() metoda klase MessageBox



197

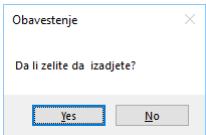
Generisanje handlera za Closing događaj forme



198

Obrada Closing događaja forme

```
private void Form5_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult rez = MessageBox.Show("Da li zelite da izadjete?",
        "Obavestenje",
        MessageBoxButtons.YesNo);
    if (rez == DialogResult.No)
    {
        e.Cancel = true;
    }
}
```



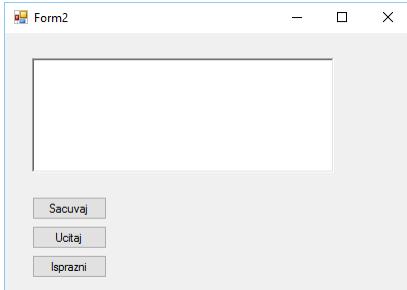
199

Osnovne karakteristike RichTextBox kontrole

- Svojstvo **Multiline** je podrazumevano postavljeno na vrednost `true`
- Svojstvo **AcceptsTab** određuje da li je moguće koristiti taster TAB unutar ove kontrole
- Metoda **AppendText(String)** dodaje tekst na veće postojeći tekst
- Svojstvo **Text** čita ili setuje sadržaj ove kontrole
- Metoda **Clear()** briše sadržaj kontrole
- Svojstvo **Lines** vraća ili setuje niz stringova koji se nalazi unutar ove kontrole

200

Korisnički interfejs



201

Metoda SaveFile()

Mora postojati folder Temp na fajl sistemu

```
private void button1_Click(object sender, EventArgs e)
{
    if (richTextBox1.Text.Length >0)
    {
        richTextBox1.SaveFile(@"C:\Temp\podaci.rtf");
        MessageBox.Show("Podaci sacuvani", "Poruka");
    }
    else
    {
        MessageBox.Show("Nije dozvoljeno cuvanje praznog sadrzaja", "Poruka");
    }
}
```

202

Metoda LoadFile()

```
using System.IO;
```

```
private void button2_Click(object sender, EventArgs e)
{
    string putanja = @"C:\Temp\podaci.rtf";
    if (File.Exists(putanja))
    {
        richTextBox1.LoadFile(putanja);
    }
}
```

203

Metoda Clear()

```
private void button3_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();
}
```

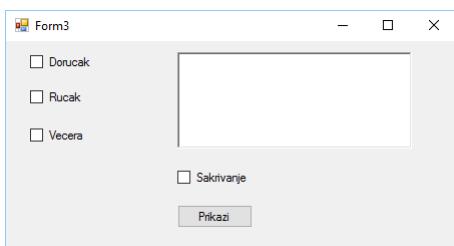
204

CheckBox kontrola

- U jednom trenutku može biti čekirano više od jednog ček dugmeta
- Svojstvo **Text** definiše tekst koji će se pojaviti pored polja za potvrdu
- Svojstvo Checked služi za čitanje ili setovanje stanja **CheckBox** kontrole
- Događaj **CheckedChanged** se okida kada CheckBox kontrola prelazi iz stanja čekirano u stanje dečekirano ili obrnuto

205

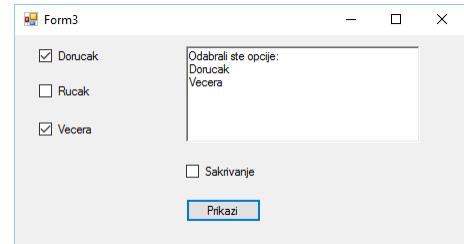
Upotreba CheckBox kontrole - GUI



206

Dugme "Prikazi"

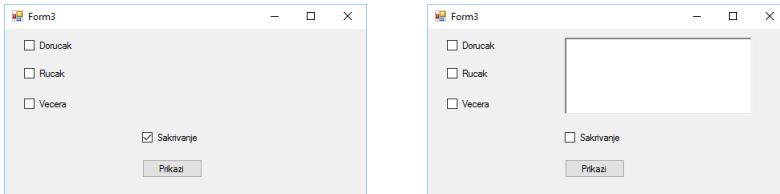
```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Text = "Odabrali ste opcije:\n";
    if (checkBox1.Checked)
    {
        richTextBox1.AppendText("Dorucak\n");
    }
    if (checkBox2.Checked)
    {
        richTextBox1.AppendText("Rucak\n");
    }
    if (checkBox3.Checked)
    {
        richTextBox1.AppendText("Vecera");
    }
}
```



207

Check dugme "Sakrivanje"

```
private void checkBox4_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox4.Checked)
    {
        richTextBox1.Visible = false;
    }
    else
    {
        richTextBox1.Visible = true;
    }
}
```



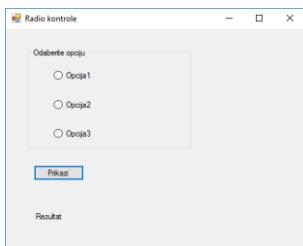
208

RadioButton kontrola

- Samo jedno radio dugme u grupi može biti čekirano u jednom trenutku
- Svojstvo **Text** definiše tekst koji će se pojaviti pored polja za potvrdu
- Svojstvo **Checked** služi za čitanje ili setovanje stanja CheckBox kontrole
- Događaj **CheckedChanged** se okida kada RadioButton kontrola prelazi iz stanja čekirano u stanje dečekirano ili obrnuto.

209

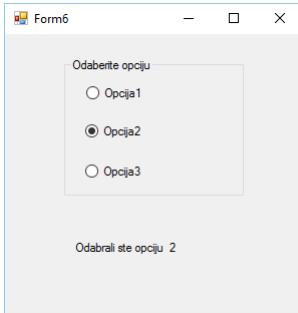
Iščitavanje stanja RadioButton kontrola



```
private void button1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked)
    {
        label1.Text = "Odabrali ste opciju 1";
    }
    else if (radioButton2.Checked)
    {
        label1.Text = "Odabrali ste opciju 2";
    }
    else if (radioButton3.Checked)
    {
        label1.Text = "Odabrali ste opciju 3";
    }
    else
    {
        label1.Text = "Morate odabrati bar jednu opciju";
    }
}
```

210

Obrada događaja CheckedChanged



```

private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton1.Checked)
    {
        label1.Text = "Odabrali ste opciju 1";
    }
}
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton2.Checked)
    {
        label1.Text = "Odabrali ste opciju 2";
    }
}
private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton3.Checked)
    {
        label1.Text = "Odabrali ste opciju 3";
    }
}

```

211

Pitanje 1

Koja od sledećih kontrola ne može da dobije fokus:

- a. TextBox
- b. Label
- c. Button

Odgovor: b

212

Pitanje 2

Load događaj forme izvršava se:

- a. Nakon što se forma prikaže korisniku
- b. Svaki put kada forma postane aktivna
- c. Pre nego što forma prvi put postane vidljiva korisniku

Odgovor c

213

Pitanje br 3

Metoda Show() klase MessageBox :

- a. vraća enumeraciju tipa DialogResult
- b. ne vraća ništa
- c. vraća true ako je rezultat pozitivan u protivnom vraća false

Odgovor: a

214

Pitanje 4

Da li se unutar FormClosing događaja forme može poništiti zahtev da se forma zatvori?

- a. Da
- b. Ne

Odgovor: a

215

Pitanje 5

Stanje CheckBox kontrole se isčitava korišćenjem :

- a. metode Selected()
- b. svojstva Checked
- c. metode Checked()
- d. svojstva IsSelected

Odgovor: b

216

Pitanje 6

Unutar GroupBox kontejnera se nalazi 5 RadioButton kontrola. U jednom trenutku može biti selektovano:

- a. samo jedno radio dugme
- b. najviše 5 radio dugmeta
- c. najviše 2 radio dugmeta

Odgovor: a

217

Pitanje 7

Sadržaj kontrole RichTextBox može se sačuvati u fajl korišćenjem njene metode:

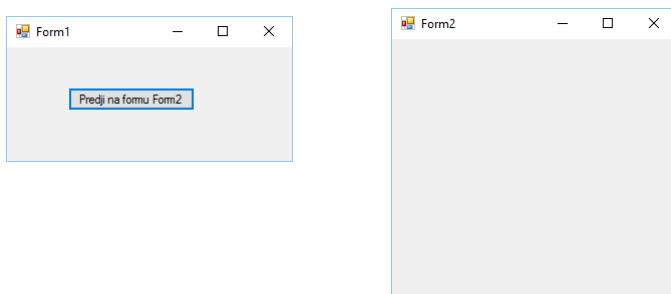
- a. SaveToFile()
- b. Save()
- c. SaveFile()

Odgovor: c

218

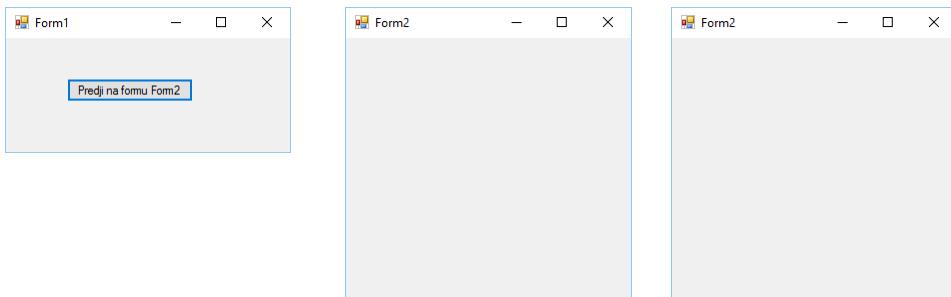
Modalne i nemodalne forme

Korisnički interfejs aplikacije



Kreiranje nemodalne forme

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.Show();
}
```



221

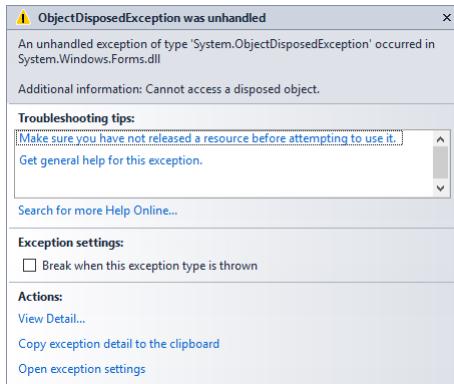
Objekat forme definisan kao polje klase Form1

```
public partial class Form1 : Form
{
    private Form2 f2 = new Form2();
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click (object sender, EventArgs e)
    {
        f2.Show();
    }
}
```

222

Zatvaranje nemodalne forme

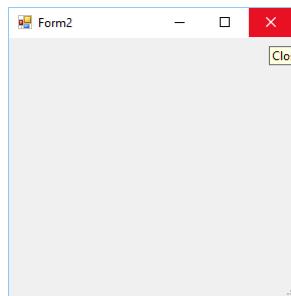
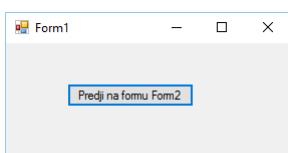


Zatvaranjem nemodalne forme uništava se odgovarajući objekat forme u memoriji.

223

Kreiranje modalne forme

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.ShowDialog();
    f2.Dispose();
}
```



224

Objekat modalne forme definisan ko polje klase Form1

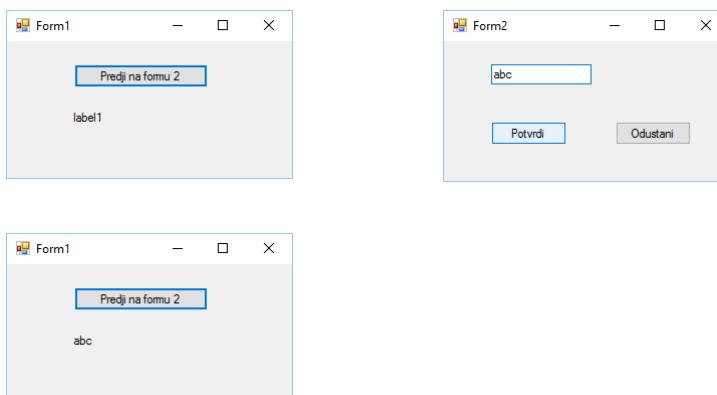
```
public partial class Form1 : Form
{
    private Form2 f2 = new Form2();
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click (object sender, EventArgs e)
    {
        f2.ShowDialog();
    }
}
```

Zatvaranjem modalne forme, objekat forme se ne uništava.

225

Slanje podataka sa child forme na parent formu



226

Forma Form2

```
public partial class Form2 : Form
{
    public string Podaci { get; set; }
    public Form2()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
    }
    private void button2_Click(object sender, EventArgs e)
    {
    }
}
```

227

DialogResult enumeracija

```
private void button1_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(textBox1.Text))
    {
        MessageBox.Show("Unesite podatke");
        textBox1.Focus();
    }
    else
    {
        Podaci = textBox1.Text;
        DialogResult = DialogResult.OK;
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.Cancel;
}
```

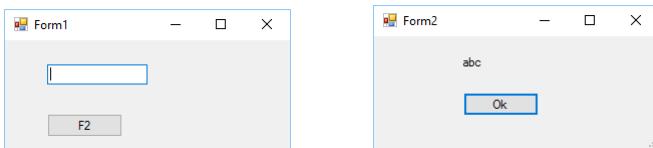
228

Forma Form1

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    if (f2.ShowDialog() == DialogResult.OK)
    {
        label1.Text = f2.Podaci;
    }
    else
    {
        label1.Text = "Cancel";
    }
    f2.Dispose();
}
```

229

Slanje podataka sa parent forme na child forme



```
public partial class Form2 : Form
{
    public string Podaci { get; set; }
    public Form2()
    {
        InitializeComponent();
    }
}
```

230

Load procedura forme Form2

```
private void Form2_Load(object sender, EventArgs e)
{
    label1.Text = Podaci;
}
```

231

Kod na formi Form1

```
private void button1_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(textBox1.Text))
    {
        Form2 f2 = new Form2();
        f2.Podaci = textBox1.Text;
        f2.ShowDialog();
        f2.Dispose();
    }
    else
    {
        MessageBox.Show("Unesite podatke");
        textBox1.Focus();
    }
}
```

232

Pitanje 1

Da li je moguće kreirati više modalnih formi (koristi se metoda ShowDialog()) jedne iste parent forme koje će egzistirati istovremeno

- a. zavisi od konteksta u kome radi aplikacija
- b. da
- c. ne

Odgovor: c

233

Pitanje 2

Unutar Click dodatnog za dugme na formi Form1 instancira se forma Form2 na sledeći nacin:

```
Form2 f2 = new Form2();
f2.Show();
```

Nakon što je korisnik kliknuo na dugme i prikazala mu se druga forma on želi da se vrati na početnu formu. Korisnik:

- a. mora da zatvori formu f2
- b. ne mora da zatvori formu f2
- c. zavisi od konteksta u kome radi aplikacija

Odgovor: b

234

Pitanje 3

Da li se zatvaranjem modalne forme, objekat forme uništava?

- a. Da
- b. Ne

Odgovor: b

235

Klasa Object

- Svaki objekat u C# je izведен iz bazne klase System.Object
- Nadimak za ovu klasu je object
- Tipu object se može pridružiti bilo koji tip
- **public virtual string ToString();** - metoda vraća string koji opisuje instancu klase
- Metoda GetType() daje tip instance objekta

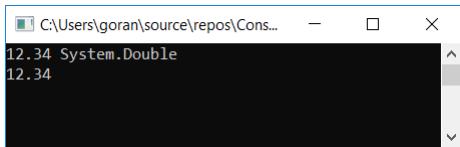
236

Boxing i unboxing

```
static void Main(string[] args)
{
    double a = 12.34;

    object o1 = a; // boxing
    Type t1 = o1.GetType();
    Console.WriteLine("{0} {1}", o1.ToString(), t1.ToString());

    double d = (double)o1; // unboxing
    Console.WriteLine(d);
    Console.ReadLine();
}
```



237

Rad sa stringovima

Stringovi

- String je nepromenjiva sekvenca unicode karaktera
- Metoda koja menja string u stvari vraća promenjenu kopiju dok originalni string ostaje nepromenjen sve dok se ne uništi od strane Garbage Collector-a
- Stringovi se mogu sortirati, kopirati, konvertovati u druge tipove i moguća je njihova enumeracija korišćenjem foreach petlje
- Operator + se koristi za nadovezivanje stringova (konkatenaciju)

```
public sealed class String : IComparable, ICloneable, IConvertible,
IEnumerable
```

239

Metode i svojstva klase String-1

Method or property	Explanation
Compare()	Overloaded public static method that compares two strings
Copy()	Public static method that creates a new string by copying another
Equals()	Overloaded public static and instance method that determines if two strings have the same value
Format()	Overloaded public static method that formats a string using a format specification
Length	Property that returns the number of characters in the instance
PadLeft()	Right-aligns the characters in the string, padding to the left with spaces or a specified character
PadRight()	Left-aligns the characters in the string, padding to the right with spaces or a specified character
Remove()	Deletes the specified number of characters
Split()	Divides a string, returning the substrings delimited by the specified characters
StartsWith()	Indicates if the string starts with the specified characters

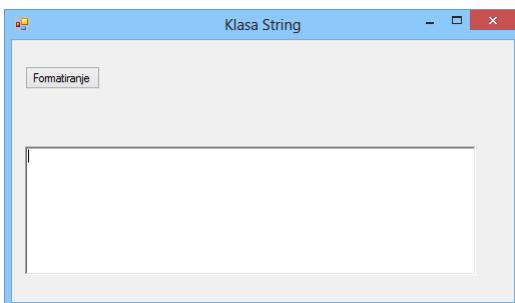
240

Metode i svojstva klase String-2

Method or property	Explanation
LastIndexOf	Reports the index position of the last occurrence of a specified Unicode character or String within this instance
IndexOf	Reports the index of the first occurrence of a String, or one or more characters, within this string.
Substring()	Retrieves a substring
ToCharArray()	Copies the characters from the string to a character array
ToLower()	Returns a copy of the string in lowercase
ToUpper()	Returns a copy of the string in uppercase
Trim()	Removes all occurrences of a set of specified characters from beginning and end of the string
TrimEnd()	Behaves like Trim(), but only at the end
TrimStart()	Behaves like trim(), but only at the start

241

Korisnički interfejs aplikacije



242

Property Length

```
private void Stampaj(object x)
{
    if (x!=null)
    {
        richTextBox1.AppendText(x.ToString() + "\n");
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    string s1 = "Ovo je neki string";

    Stampaj(s1.Length);
}
```

243

string.Format()

```
private void button1_Click(object sender, EventArgs e)
{
    int a = 55;
    int b = 45;
    string f1 = string.Format("Broj {0} je veci od broja {1}.", a, b);
    string f2 = $"Broj {a} je veci od broja {b}";
    string f3 = string.Format("{0:C}", 345.67);
    string f4 = string.Format("{0:D}", DateTime.Now);
    string f5 = string.Format("{0:t}", DateTime.Now);

    Stampaj(f1);
    Stampaj(f2);
    Stampaj(f3);
    Stampaj(f4);
    Stampaj(f5);
}
```

Broj 55 je veci od broja 45.
 Broj 55 je veci od broja 45
 346 RSD
 subota, 19. novembar 2016.
 22.44

244

Metode EndsWith() i StartsWith()

```
private void button3_Click(object sender, EventArgs e)
{
    string putanja = @"C:\Temp\proba.txt";

    Stampaj(putanja.EndsWith(".txt"));

    Stampaj(putanja.StartsWith(@"C:\"));
}
```

True
True

245

Metode ToUpper() i ToLower()

```
private void button4_Click(object sender, EventArgs e)
{
    string s1 = "Pera Peric";
    Stampaj(s1.ToUpper());
    Stampaj(s1.ToLower());
}
```

PERA PERIC
pera peric

246

Poređenje stringova

```
private void button5_Click(object sender, EventArgs e)
{
    string s1 = "Aca";
    string s2 = "Bilja";
    Stampaj(s1.CompareTo(s2));
    Stampaj(s1.CompareTo(s1));
    Stampaj(s2.CompareTo(s1));
    Stampaj(String.Compare(s1, s2));
}
```

-1
0
1
-1

247

Pronalaženje pozicije karaktera ili podstringa u stringu

```
private void button6_Click(object sender, EventArgs e)
{
    string s = "Pera Peric";
    Stampaj(s.IndexOf("P"));
    Stampaj(s.LastIndexOf("P"));

    Stampaj(s.IndexOf("Peric"));

    // pristup 4-tom karakteru u stringu
    Stampaj(s[3]);
}
```

0
5
5
a

248

Izdvajanje podstringa

```
private void button7_Click(object sender, EventArgs e)
{
    string s = "Pera Peric";
    Stampaj(s.Substring(5, 3));
    Stampaj(s.Substring(5));
}
```

Per
Peric

249

Metoda Split()

```
private void button8_Click(object sender, EventArgs e)
{
    string s1 = "Ovo je neki string";
    char separator = ' ';

    string[] niz = s1.Split(separator);

    foreach (string s in niz)
    {
        Stampaj(s);
    }
}
```

Ovo
je
neki
string

250

Metode PadLeft() i PadRight()

```
private void button9_Click(object sender, EventArgs e)
{
    string str1 = "Pera";
    string str2 = "Djordje";
    char padCh = '.';

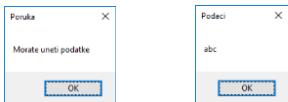
    Stampaj(str1.PadLeft(15, padCh));
    Stampaj(str1.PadRight(15, padCh));
    Stampaj(str2.PadLeft(15, padCh));
    Stampaj(str2.PadRight(15, padCh));
}
```

.....Pera
Pera.....
.....Djordje
Djordje.....

251

string.IsNullOrEmptyWhiteSpace()

```
private void button10_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmptyWhiteSpace(textBox1.Text))
    {
        MessageBox.Show("Morate uneti podatke", "Poruka");
    }
    else
    {
        MessageBox.Show(textBox1.Text.Trim(), "Podaci");
    }
}
```



252

Osobina nepromenljivosti stringova-1

```
private void button11_Click(object sender, EventArgs e)
{
    string s = "test";
    for (int i = 0; i < 100; i++)
    {
        s += i.ToString();
    }
    richTextBox1.Text = s.ToString();
}
```

253

Osobina nepromenljivosti stringova-2

- Stringovi su nepromenljivi (immutable) tj. nakon toga što se string sačuva u memoriji ta memorijska lokacija se ne može promeniti
- Linija koda: `s += i.ToString();` kreira novi string i onda rezultat čuva na novoj memorijskoj lokaciji
- I stara i nova verzija stringa se čuvaju privremeno u memoriji
- Stara verzija stringa će se obrisati u procesu “odnošenja smeća” (garbage collection).
- Ako aplikacija često manipuliše stringovima u memoriji se čuvaju nepotrebni podaci čekajući sledeći period “odnošenja smeća”
- Rešenje klasa `StringBuilder` koja se nalazi u namespace-u `System.Text`

254

Klasa StringBuilder

- String bilder alocira inicijalnu vrednost od 16 karaktera i kako string postaje veći alocirana memorija se proširila da bi se prilagodila dužini stringa (slično kao kod kolekcija)

```
private void button11_Click(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder("test");
    for (int i = 0; i < 100; i++)
    {
        sb.Append(i.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

255

Pitanje 1

Neka je Stampaj() metoda koja služi da prikaže tekst na korisničkom interfejsu windows aplikacije. Aplikacija izvršava sledeće linije koda:

```
string s = "Pera Peric";
Stampaj(s.Substring(5, 3));
```

Šta se ispisuje na korisničkom interfejsu?

- Pera Pera Pera
- Per
- Peric
- Peri

Odgovor: b

256

Pitanje 2

Dinamički string , odnosno string koji može menjati svoju dužinu se kreira kao instanca klase:

- a. DinamicString
- b. StringBuffer
- c. String
- d. StringBuilder

Odgovor: d

257

Pitanje 3

Da li je u C# moguće kreirati klasu koja je izvedena iz sistemske klase String

- a. zavisi od konteksta u kome radi aplikacija
- b. ne
- c. da

Odgovor: b

258

Rad sa datumsko-vremenskim vrednostima

Null-conditional operator?.

Koristi se za testiranje na null vrednost pre pristupa članu objekta

```
private void button1_Click(object sender, EventArgs e)
{
    Random rnd = new Random();
    int a = rnd.Next(2); // 0 ili 1
    string s = null;
    if (a == 1)
    {
        s = "Test";
    }

    int? duzina = s?.Length;

    if (duzina != null)
    {
        label1.Text = "String duzine: " + duzina;
    }
    else
    {
        label1.Text = "NULL string";
    }
}
```

260

Null coalescing operator ??

Vraća levi operand ako je različit od null u protivnom vraća desni operand

```
private void button2_Click(object sender, EventArgs e)
{
    Random rnd = new Random();
    int a = rnd.Next(2); // 0 ili 1

    string s = null;

    if (a == 1)
    {
        s = "Test";
    }

    label1.Text = s ?? "Null string";
}
```

261

Konstruktori DateTime strukture

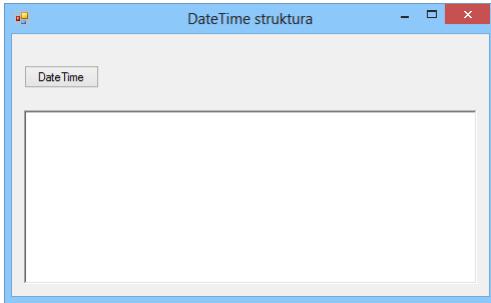
```
[SerializableAttribute]
public struct DateTime : IComparable, IFormattable,
IConvertible, ISerializable, IComparable<DateTime>, IEquatable<DateTime>
```

```
public DateTime(
int year,
int month,
int day)
```

```
public DateTime(
    int year,
    int month,
    int day,
    int hour,
    int minute,
    int second
);
```

262

Korisnički interfejs aplikacije



```
private void Stampaj(object x)
{
    richTextBox1.AppendText((x?.ToString() ?? "Null") + "\n");
}
```

263

Upotreba DateTime strukture -1

```
private void button1_Click(object sender, EventArgs e)
{
    DateTime dt = new DateTime(2005, 7, 15, 11, 20, 0);
    Stampaj(dt);
    string s = dt.ToString("dd.MM.yyyy");
    Stampaj(s);
}
private void button2_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Now;
    Stampaj(dt.ToString("dd.MM.yyyy"));
    Stampaj(dt.ToString("dd/MM/yyyy"));
}

private void button3_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Now;
    Stampaj(dt.ToString("yyyy-MM-dd"));
    Stampaj(dt.ToString("dd/MM/yyyy"));
}
```

15.07.2005. 11.20.00
15.07.2005

četvrtak, 29. mart
29.03.2018.

10.08.27
10.08

264

Upotreba DateTime strukture -2

```
private void button4_Click(object sender, EventArgs e)
{
    DateTime dt1 = DateTime.Now;
    string s1 = dt1.ToString("hh:mm:ss");
    Stampaj(s1);
}
```

10.09.17

```
private void button5_Click(object sender, EventArgs e)
{
    DateTime dt1 = DateTime.Now;
    string s = dt1.Hour.ToString() + " : " + dt1.Minute.ToString();
    Stampaj(s);
}
```

265

TimeSpan struktura

Koristi se da predstavi vremenski interval.

```
[SerializableAttribute][ComVisibleAttribute(true)]public struct TimeSpan :  
IComparable, IComparable<TimeSpan>, IEquatable<TimeSpan>, IFormattable
```

```
public TimeSpan(  
    int hours,  
    int minutes,  
    int seconds)
```

```
public TimeSpan(  
    int days,  
    int hours,  
    int minutes,  
    int seconds )
```

266

Upotreba TimeSpan strukture

```
private void button1_Click(object sender, EventArgs e)
{
    DateTime prijava = new DateTime(2018, 4, 25, 18, 32, 0);
    DateTime odjava = new DateTime(2018, 5, 4, 19, 47, 34);
    TimeSpan putovanje = odjava - prijava;
    string s = $"{prijava} - {odjava} = {putovanje}";
    Stampaj(s);
}
```

25.04.2018. 18.32.00 - 04.05.2018. 19.47.34 = 9.01:15:34

267

TimeSpan struktura-1

```
private void button1_Click(object sender, EventArgs e)
{
    DateTime dt1 = DateTime.Today;
    Stampaj(dt1);
    TimeSpan ts = new TimeSpan(23, 0, 0,0);

    DateTime dt2 = dt1 + ts;
    DateTime dt3 = dt1 - ts;

    Stampaj(dt2.ToString("dd/MM/yyyy"));
    Stampaj(dt3.ToString("dd/MM/yyyy"));
}
```

29.03.2018. 00.00.00
21.04.2018.
06.03.2018.

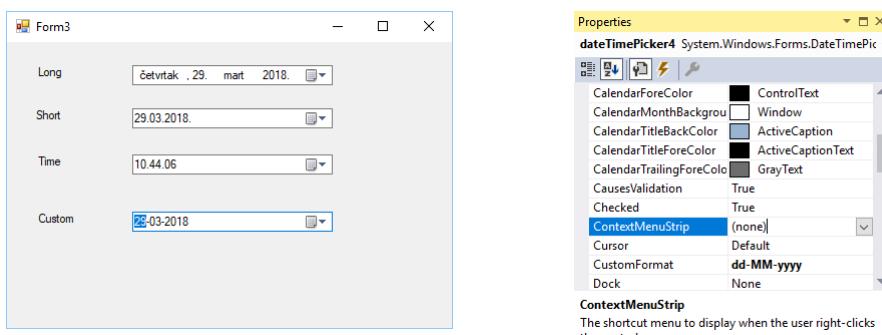
268

DatePicker Kontrola

- Omogućava setovanje datuma ili vremena korišćenjem jednostavnog grafičkog interfejsa
- Svojstvo Format omogućava setovanje formata prikaza u kontroli
- Svojstvo Value određuje trenutno setovani datume ili vreme
- Ukoliko se kao Format setuje Time, treba postaviti svojstvo ShowUpDown na vrednost True

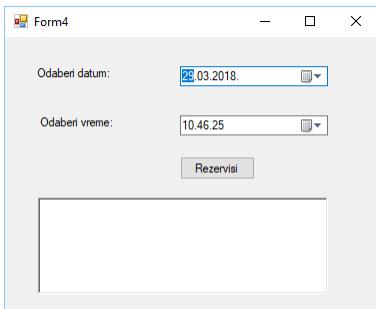
269

Definisanje formata prikaza DateTime Picker kontrole



270

Primer upotrebe DateTimePicker kontrole-GUI



271

Očitavanje vrednosti iz DateTimePicker kontrole

```
private void button1_Click(object sender, EventArgs e)
{
    DateTime dt1 = dateTimePicker1.Value;
    StringBuilder sb1 = new StringBuilder();
    sb1.Append("Vas datum polaska je :\n");
    sb1.Append(dt1.ToString() + "\n");

    DateTime dt2 = dateTimePicker2.Value;

    sb1.Append("Vase vreme polaska je:\n");
    sb1.Append(dt2.ToString());

    richTextBox1.AppendText(sb1.ToString());
}
```

272

Pitanje 1

Svojstvo Value kontrole DateTimePicker :

- a. vraća vrednost tipa DateTime
- b. vraća vrednost tipa string
- c. vraća promenljivu tipa int
- d. vraća objekat tipa DatePicker

Odgovor: a

273

Obrada izuzetaka

274

Pojam izuzetka

- Izuzetak je objekat koji sadrži informacije o grešci koja nastaje tokom izvršavanja koda
- Izuzetak je objekat izveden iz klase Exception ili klase koja je izvedena iz klase Exception
- Izuzetak se izbacuje od strane CLR ili eksplisitno od strane koda
- Izuzeci se obrađuju korišćenjem ključnih reči **try**, **catch** i **finally**

275

Obrada izuzetaka

- Blok try zahteva postojanje catch bloka i/ili finally bloka
- Kod unutar try bloka može da izbacuje različite tipove izuzetaka
- Naredbe unutar catch bloka se izvršavaju ukoliko je izuzetak izbačen unutar try bloka.
- Može se definisati više catch blokova od kojih svaki obrađuje specijalizovanu klasu izuzetaka.
- Blok finally omogućava da se oslobođe resursi i da se specificira kod koji će se uvek izvršiti nezavisno od toga da li je došlo do izuzetka ili ne
- Blok finally je opcioni blok

276

try-catch blok

```
try
{
    // deo koda u kome moze doci do izuzetka
}

catch (Exception ex)
{
    // obrada izuzetka
}
```

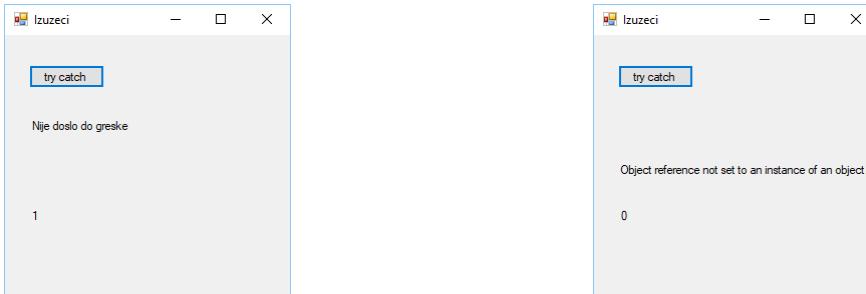
277

try-catch blok

```
private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "";
    label2.Text = "";
    label3.Text = "";
    object o = null;
    int i = 0;
    Random rnd = new Random();
    if (rnd.Next(3) == 1)
    {
        o = 1;
    }
    try
    {
        i = (int)o;
        label1.Text = "Nije doslo do greske";//izvrsava se samo ako ne dodje do greske
    }
    catch (Exception xcp)
    {
        label2.Text = xcp.Message;
    }
    //Stampam i
    label3.Text = i.ToString();
}
```

278

try-catch blok



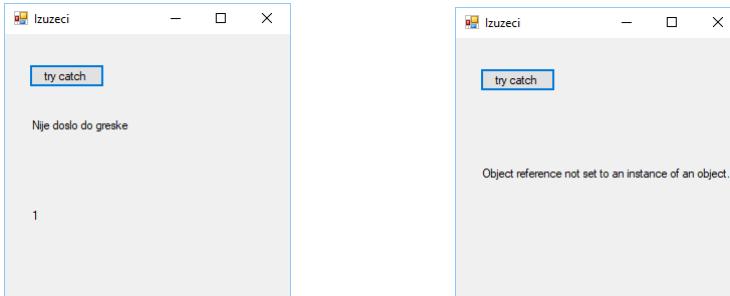
279

Modifikacija catch bloka -1

```
private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "";
    label2.Text = "";
    label3.Text = "";
    object o = null;
    int i = 0;
    Random rnd = new Random();
    if (rnd.Next(3) == 1)
    {
        o = 1;
    }
    try
    {
        i = (int)o;
        label1.Text = "Nije doslo do greske";//izvrsava se samo ako ne dodje do greske
    }
    catch (Exception xcp)
    {
        label2.Text = xcp.Message;
        return;
    }
    //Stampam i
    label3.Text = i.ToString();
}
```

280

Modifikacija catch bloka -2



281

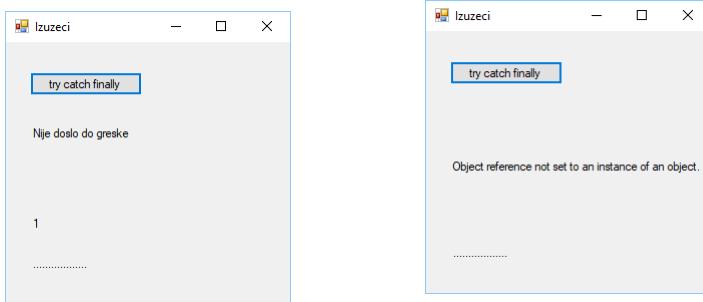
try, catch, finally blok

```
private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "";
    label2.Text = "";
    label3.Text = "";
    object o = null;
    int i = 0;
    Random rnd = new Random();
    if (rnd.Next(3) == 1)
    {
        o = 1;
    }
    try
    {
        i = (int)o;
        label1.Text = "Nije doslo do greske";
        //izvrsava se samo ako ne dodje do greske
    }
    catch (Exception xcp)
    {
        label2.Text = xcp.Message;
        return;
    }
}
```

```
finally
{
    // uvek izvrsavam
    label4.Text = ".....";
}
//Stampam samo ako nije doslo do greske
label3.Text = i.ToString();
}
```

282

try, catch, finally blok -2



283

Korišćenje više catch blokova

- Ukoliko postoji više catch blokova tada treba prvo hendlovati izuzetke koji su više specijalizovani, a zatim opštije.
- Npr. **DivideByZeroException** klasa je izvedena iz klase **ArithmetricException**.

```
try
{
    // kod u kome dolazi do izuzetka
}
catch (DivideByZeroException)
{
    // kod se izvrsava ukoliko dodje do pokusaja deljenja sa nulom
}
catch (ArithmetricException)
{
    // neki drugi aritmeticki izuzetak npr. OverflowException
}
```

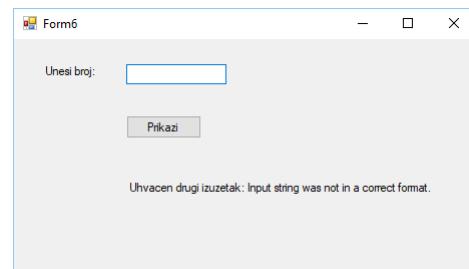
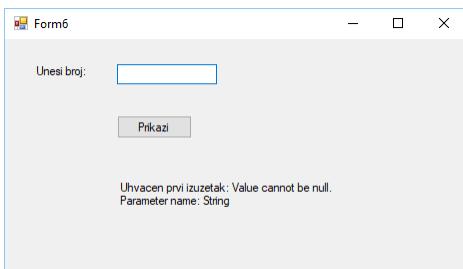
284

Upotreba dva catch bloka-1

```
private void button1_Click(object sender, EventArgs e)
{
    string s = null;
    try
    {
        if (!string.IsNullOrWhiteSpace(textBox1.Text))
        {
            // string nije prazan
            s = textBox1.Text.Trim();
        }
        int a = int.Parse(s);
        labelPoruka.Text = a.ToString();
    }
    catch (ArgumentNullException ex)
    {
        labelPoruka.Text = "Uhvacen prvi izuzetak: " + ex.Message;
    }
    catch (Exception ex)
    {
        labelPoruka.Text = "Uhvacen drugi izuzetak: " + ex.Message;
    }
    textBox1.Clear();
    textBox1.Focus();
}
```

285

Upotreba dva catch bloka-1



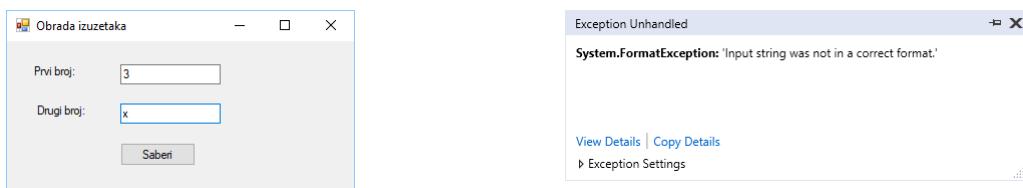
286

Primer bez upotrebe izuzetaka

```
private void button1_Click(object sender, EventArgs e)
{
    double a = double.Parse(textBox1.Text);
    double b = double.Parse(textBox2.Text);
    double zbir = a + b;
    MessageBox.Show("Zbir je: " + zbir, "Rezultat");
}
```

287

Aplikacija prekida sa radom zbog izuzetka



288

Kod sa obradom izuzetaka

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        double a = double.Parse(textBox1.Text);
        double b = double.Parse(textBox2.Text);
        double zbir = a + b;
        MessageBox.Show("Zbir je: " + zbir, "Rezultat");
    }
    catch (Exception xcp)
    {
        MessageBox.Show(xcp.Message);
    }
    textBox1.Clear();
    textBox2.Clear();
    textBox1.Focus();
}
```

289

Pitanje 1

Kod koji može da dovede do greške prilikom izvršavanja stavlja se unutar bloka:

- a. try
- b. catch
- c. finally

Odgovor: a

290

Pitanje 2

Ako zelimo da se neka sekvenca naredbi izvršava nezavisno od toga da li je do izuzetka došlo ili ne tada se ona stavlja unutar :

- a. try bloka
- b. catch bloka
- c. finally bloka

Odgovor: c

291

Pitanje 3

Generisani izuzetak od strane CLR komponente prosleđuje se bloku:

- a. try
- b. catch
- c. finally

Odgovor: b

292

Pitanje 4

Svaki izuzetak može se dodeliti referenci tipa Exception:

- a. Da
- b. Ne

Odgovor: a

293

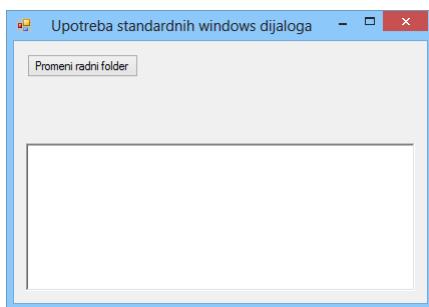
Standardni windows dijalozi

Standardni windows dijalozi

- Nalaze se u Dialogs sekciji Toolboxa
- Instanciraju se prevlačenjem na windows formu
 - OpenFileDialog
 - SaveFileDialog
 - FontDialog
 - ColorDialog
 - FolderBrowserDialog

295

Primer upotrebe standardnih dijaloga



296

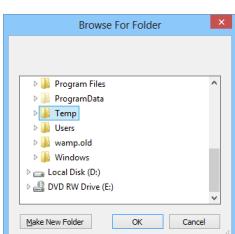
Load procedura forme

```
private void Form1_Load(object sender, EventArgs e)
{
    folderBrowserDialog1.RootFolder = Environment.SpecialFolder.MyComputer;
    openFileDialog1.InitialDirectory = @"C:\Temp";
    saveFileDialog1.InitialDirectory = @"C:\Temp";
    richTextBox1.Font = new Font("Calibri", 11);
}
```

297

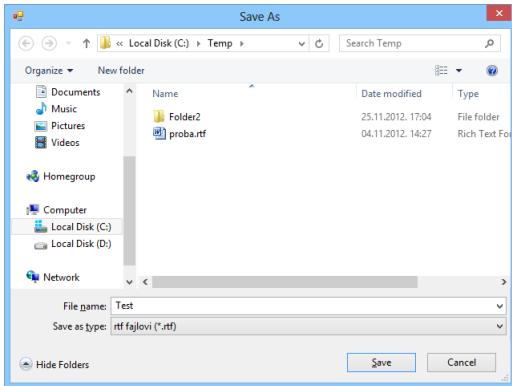
FolderBrowserDialog

```
private void button1_Click(object sender, EventArgs e)
{
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        string radniFolder = folderBrowserDialog1.SelectedPath;
        openFileDialog1.InitialDirectory = radniFolder;
        saveFileDialog1.InitialDirectory = radniFolder;
        MessageBox.Show("Promenjen radni folder aplikacije", "Obavestenje");
    }
}
```



298

SaveFileDialog



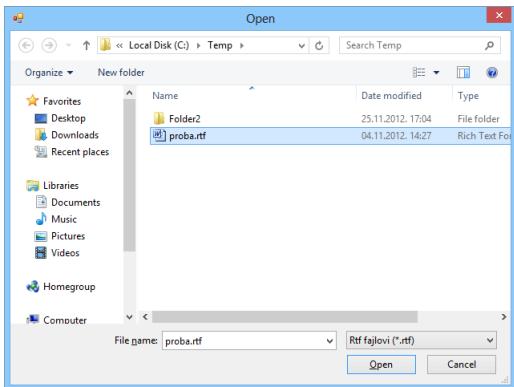
299

SaveFileDialog-upotreba

```
private void button2_Click(object sender, EventArgs e)
{
    saveFileDialog1.Filter = "rtf fajlovi|*.rtf";
    if (richTextBox1.Text.Trim().Length > 0)
    {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            richTextBox1.SaveFile(saveFileDialog1.FileName);
            MessageBox.Show("Podaci upisani u fajl", "Poruka");
            richTextBox1.Clear();
        }
    }
    else
    {
        MessageBox.Show("Nije dozvoljeno cuvanje praznog sadrzaja", "Poruka");
    }
}
```

300

OpenFileDialog



301

OpenFileDialog-upotreba

```
private void button3_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "Rtf fajlovi|*.rtf";

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        richTextBox1.LoadFile(openFileDialog1.FileName);
    }
}
```

302

ColorDialog

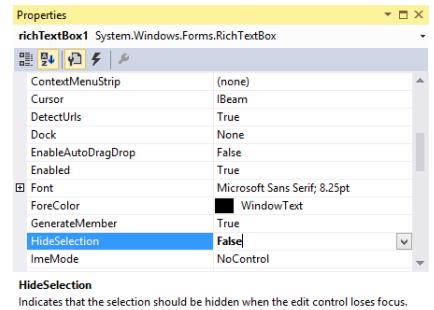


303

ColorDialog-upotreba

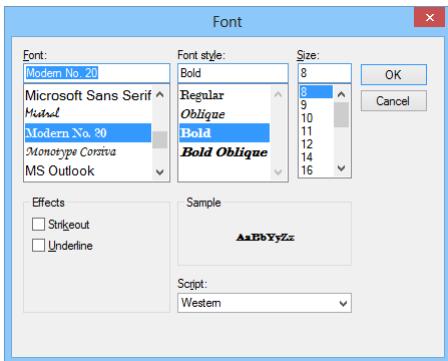
```
private void button1_Click(object sender, EventArgs e)
{
    colorDialog1.Color = richTextBox1.SelectionColor;

    if (colorDialog1.ShowDialog() == DialogResult.OK)
    {
        if (richTextBox1.SelectionLength > 0)
        {
            richTextBox1.SelectionColor = colorDialog1.Color;
            richTextBox1.DeselectAll();
        }
        else
        {
            richTextBox1.ForeColor = colorDialog1.Color;
        }
    }
}
```



304

FontDialog



305

FontDialog-upotreba

```
private void button2_Click(object sender, EventArgs e)
{
    fontDialog1.Font = richTextBox1.SelectionFont;
    if (fontDialog1.ShowDialog() == DialogResult.OK)
    {
        if (richTextBox1.SelectionLength > 0)
        {
            richTextBox1.SelectionFont = fontDialog1.Font;
        }
        else
        {
            richTextBox1.Font = fontDialog1.Font;
        }
    }
}
```

306

Pitanje 1

FolderBrowserDialog se prikazuje korišćenjem metode:

- a. Show()
- b. Display()
- c. ShowDialog()

Odgovor: c

307

Pitanje 2

Ograničenje ekstenzije fajla koji će biti prikazan u instanci OpenFileDialog klase vrši se pomoću svojstva:

- a. FileType
- b. FileExtension
- c. Filter

Odgovor: c

308

Pitanje 3

Boja selektovanog teksta u RichTextBox kontroli definiše se korišćenjem svojstva:

- a. SelectionColor
- b. ForeColor
- c. Color

Odgovor: a

309

Kolekcije

Pojam kolekcije

- Kolekcije se upotrebljavaju za upravljenje grupama objekata i imaju više funkcionalnosti nego nizovi
- Veličina niza se mora unapred definisati dok to nije slučaj sa kolekcijama
- Klase kolekcija nalaze se u prostoru imena System.Collections
- Klase kolekcija su definisane na bazi jasno definisanih interfejsa
- Negeneričke kolekcije rade sa tipom podataka object

311

Primeri negeneričkih kolekcija

- Klase
- ArrayList
- Queue
- Stack
- Hashtable
- Neophodno kastovanje članova kolekcije u njihov stvaran tip
- Ove kolekcije mogu miksovati različite tipove podataka
- Nisu type-safe

312

Kolekcija ArrayList

- Elementima liste pristupa se preko indeksa kao i kod niza
- Za razliku od niza nije neophodno unapred poznavati broj elemenata
- Metoda **Add(object)** dodaje objekat na kraj liste
- Metoda **Clear()** briše sve elemente iz liste
- Metoda **Insert(pozicija, vrednost)** ubacuje objekat **vrednost** na poziciju **pozicija**
- Metoda **RemoveAt(index)** briše elemenat sa indeksom index iz liste

```
public class ArrayList : IList, ICollection, IEnumerable, ICloneable
```

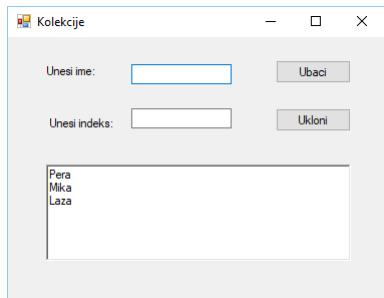
313

Kolekcija ArrayList

- Metoda **Remove(object)** uklanja prvo pojavljivanje specifičnog objekta iz kolekcije
- Svojstvo **Count** daje broj članova kolekcije
- Metoda **Contains(object)** određuje da li se određeni element nalazi u kolekciji
- Metoda **Sort()** sortira elemente kolekcije
- Metoda **Reverse()** prikazuje elemente kolekcije u inverznom redosledu
- Metoda **ToArray()** kopira elemente liste u jednodimenzionalan niz

314

Primer upotrebe kolekcija



315

Definisanje i štampanje kolekcije

```
public partial class Form1 : Form
{
    private ArrayList listaImena = new ArrayList();
    public void Stampaj(ArrayList al)
    {
        richTextBox1.Clear();
        if (al.Count > 0)
        {
            foreach (string clan in al)
            {
                richTextBox1.AppendText(clan + "\n");
            }
        }
    }
}
```

316

Load procedura forme

```
private void Form1_Load(object sender, EventArgs e)
{
    listaImena.Add("Pera");
    listaImena.Add("Mika");
    listaImena.Add("Laza");
    Stampaj(listaImena);
}
```

317

Dodavanje člana u kolekciju

```
private void button1_Click(object sender, EventArgs e)
{
    string ime = textBox1.Text.Trim();
    if (ime.Length > 1)
    {
        string b0 = ime.Substring(0, 1).ToUpper();
        string b1 = ime.Substring(1).ToLower();
        ime = b0 + b1;

        if (!listaImena.Contains(ime))
        {
            listaImena.Add(ime);
            Stampaj(listaImena);
        }
        else
        {
            MessageBox.Show("Clan se vec nalazi u kolekciji");
        }
    }
    else
    {
        MessageBox.Show("Ime mora sadrzati bar 2 slova");
    }
    textBox1.Clear();
    textBox1.Focus();
}
```

318

Uklanjanje člana iz kolekcije

```
private void button2_Click(object sender, EventArgs e)
{
    int indeks;
    if (!int.TryParse(textBox2.Text, out indeks))
    {
        MessageBox.Show("Unesite ceo broj");
        return;
    }

    if (indeks >= 0 && indeks < listaImena.Count)
    {
        listaImena.RemoveAt(indeks);
        Stampaj(listaImena);
    }
    else
    {
        MessageBox.Show("Indeks van opsega");
    }
}
```

319

Pitanje 1

Svi članovi kolekcije ArrayList su tipa:

- a. int
- b. object
- c. string

Odgovor: b

320

Pitanje 2

Broj članova neke negeneričke kolekcije dobija se korišćenjem:

- a. svojstva Count
- b. svojstva Length
- c. metode Count()
- d. metode Number()

Odgovor: a

321

Pitanje 3

Sve članove ArrayList kolekcije moguće je prikazati foreach petljom:

- a. Da
- b. Ne

Odgovor: a

322

Pitanje 4

Uklanjanje člana ArrayList kolekcije sa određene pozicije vrši se korišćenjem metode:

- a. Remove()
- b. RemoveAt()
- c. DefeteFrom()()

Odgovor: b

323

Generičke liste

Generičke liste

List<T>

using System.Collections.Generic;

```
private void button1_Click(object sender, EventArgs e)
{
    List<int> celobrojnaLista = new List<int>();
    celobrojnaLista.Add(1);
    celobrojnaLista.Add(2);
    celobrojnaLista.Add(55);

    for (int i = 0; i < celobrojnaLista.Count; i++)
    {
        richTextBox1.AppendText(celobrojnaLista[i] + "\n");
    }

    richTextBox1.AppendText("Novi nacin stampanja\n");

    foreach (int i in celobrojnaLista)
    {
        richTextBox1.AppendText(i + "\n");
    }
}
```

325

Klasa Trkac

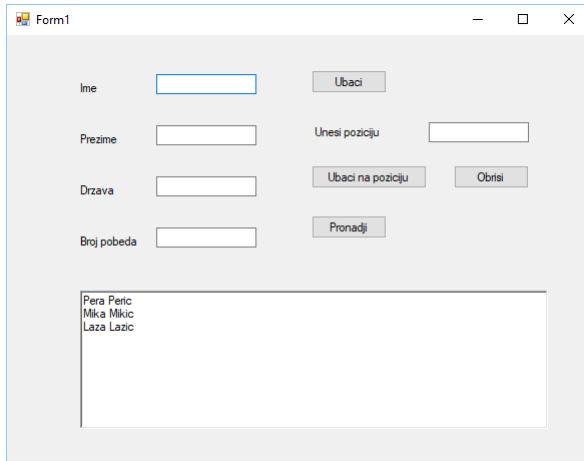
```
public class Trkac
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public string Drzava { get; set; }
    public int BrojPobeda { get; set; }

    public Trkac(string Ime, string Prezime, string Drzava, int BrojPobeda)
    {
        this.Ime = Ime;
        this.Prezime = Prezime;
        this.Drzava = Drzava;
        this.BrojPobeda = BrojPobeda;
    }

    public override string ToString()
    {
        return Ime + " " + Prezime + "\n";
    }
}
```

326

GUI aplikacije



327

Štampanje liste

```
private List<Trkac> lista= new List<Trkac>();
```

```
private void StampajListu(List<Trkac> trkaci)
{
    richTextBox1.Clear();
    foreach (Trkac t in trkaci)
    {
        richTextBox1.AppendText(t.ToString());
    }
}
```

328

Load događaj forme

```
private void Form1_Load(object sender, EventArgs e)
{
    Trkac t1 = new Trkac { Ime = "Pera", Prezime = "Peric", BrojPobeda = 12, Drzava = "Srbija" };
    lista.Add(t1);

    Trkac t2 = new Trkac { Ime = "Mika", Prezime = "Mikic", BrojPobeda = 10, Drzava = "Srbija" };
    lista.Add(t2);

    Trkac t3 = new Trkac { Ime = "Laza", Prezime = "Lazic", BrojPobeda = 8, Drzava = "Srbija" };
    lista.Add(t3);

    StampajListu(lista);
}
```

329

Resetovanje korisničkog interfejsa

```
private void Resetuj()
{
    textBoxIme.Clear();
    textBoxPrezime.Clear();
    textBoxDrzava.Clear();
    textBoxBrojPobeda.Clear();
}
```

330

Metoda za validaciju

```
public bool Validacija()
{
    if (textBoxIme.Text.Trim().Length < 2)
    {
        MessageBox.Show("Ime mora imati najmanje 2 karaktera");
        textBoxIme.Focus();
        return false;
    }
    if (textBoxPrezime.Text.Trim().Length < 2)
    {
        MessageBox.Show("Prezime mora imati najmanje 2 karaktera");
        textBoxPrezime.Focus();
        return false;
    }
    if (textBoxDrzava.Text.Trim().Length < 2)
    {
        MessageBox.Show("Drzava mora imati najmanje 2 karaktera");
        textBoxDrzava.Focus();
        return false;
    }
    int brojPobeda;

    if (! int.TryParse(textBoxBrojPobeda.Text.Trim(), out brojPobeda))
    {
        MessageBox.Show("Broj pobeda mora biti ceo broj");
        textBoxBrojPobeda.Clear();
        textBoxBrojPobeda.Focus();
        return false;
    }
    return true;
}
```

331

String metoda

```
public string VelikoPrvoSlovo(string rec)
{
    rec = rec.Trim().ToLower();
    if (rec.Length > 1)
    {
        return rec.Substring(0, 1).ToUpper() + rec.Substring(1);
    }
    return string.Empty;
}
```

332

Metoda KreirajTrkaca()

```
public Trkac KreirajTrkaca()
{
    if (Validacija())
    {
        string ime = VelikoPrvoSlovo(textBoxIme.Text);
        string prezime = VelikoPrvoSlovo(textBoxPrezime.Text);
        string drzava = VelikoPrvoSlovo(textBoxDrzava.Text);
        int brojPobeda = int.Parse(textBoxBrojPobeda.Text.Trim());

        Trkac t = new Trkac
        {
            Ime =ime, Prezime=prezime, Drzava=drzava, BrojPobeda = brojPobeda
        };
        return t;
    }
    else
    {
        return null;
    }
}
```

333

Ubacivanje u Listu-1

```
private void button1_Click(object sender, EventArgs e)
{
    Trkac t = KreirajTrkaca();
    if (t != null)
    {
        lista.Add(t);
        StampajListu(lista);
        resetuj();
    }
}
```

334

Definisanje jednakosti dva objekta

```
class Trkac : IEquatable<Trkac>
{
    ....
    public bool Equals(Trkac other)
    {
        if (Ime.ToLower() == other.Ime.ToLower()
            && Prezime.ToLower() == other.Prezime.ToLower())
        {
            return true;
        }
        return false;
    }
}
```

335

Ubacivanje u Listu-2

```
private void button1_Click(object sender, EventArgs e)
{
    Trkac t = KreirajTrkaca();
    if (t != null)
    {
        if (lista.Contains(t))
        {
            MessageBox.Show("Trkac se vec nalazi u listi", "Poruka");
        }
        else
        {
            lista.Add(t);
            StampajListu(lista);
        }
        resetuj();
    }
}
```

336

Ubacivanje elementa na poziciju

```
private void button2_Click(object sender, EventArgs e)
{
    Trkac t = kreirajTrkaca();
    if (t != null)
    {
        int pozicija;

        if (!int.TryParse(textBoxPozicija.Text, out pozicija))
        {
            MessageBox.Show("Pozicija u listi mor biti ceo broj", "Poruka");
            return;
        }
        if (pozicija >= 0 && pozicija <= lista.Count)
        {
            lista.Insert(pozicija, t);
            StampajListu(lista);
        }
        else
        {
            MessageBox.Show("Prekoracili ste poziciju", "Poruka");
        }
        resetuj();
        textBoxPozicija.Clear();
    }
}
```

337

Uklanjanje elementa iz liste

```
private void button3_Click(object sender, EventArgs e)
{
    int pozicija;

    if (!int.TryParse(textBoxPozicija.Text, out pozicija))
    {
        MessageBox.Show("Pozicija u listi mora biti ceo broj", "Poruka");
        textBoxPozicija.Focus();
        return;
    }
    if (pozicija >= 0 && pozicija < lista.Count)
    {
        lista.RemoveAt(pozicija);
        StampajListu(lista);
    }
    else
    {
        MessageBox.Show("Ne postoji clan liste", "Poruka");
    }
}
```

338

Pristup elementima generičke liste

```
private void button4_Click(object sender, EventArgs e)
{
    int pozicija;
    if (!int.TryParse(textBoxPozicija.Text, out pozicija))
    {
        MessageBox.Show("Pozicija u listi mora biti ceo broj", "Poruka");
        textBoxPozicija.Focus();
        return;
    }
    if (pozicija > -1 && pozicija < listaTrkaca.Count)
    {
        Trkac t = listaTrkaca[pozicija];
        textBoxIme.Text = t.Ime;
        textBoxPrezime.Text = t.Prezime;
        textBoxDrzava.Text = t.Drzava;
        textBoxBrojPobeda.Text = t.BrojPobeda.ToString();
    }
    else
    {
        MessageBox.Show("Prekoracili ste poslednju poziciju", "Poruka");
    }
}
```

339

Pitanje 1

Generički celobrojna lista pod nazivom **intLista** se instancira na sledeći način:

- `int<List> intLista = new int<List>();`
- `List<int> intLista = new List<int>();`
- `List[int] intLista = new List[int];`

Odgovor: b

340

Pitanje 2

Da li je unutar generičke liste, koja nije tipa object, dozvoljeno čuvanje podataka različitog tipa:

- a. Da
- b. Ne

Odgovor: b

341

Pitanje 3

Drugom članu generičke liste tipa string pod nazivom **stringLista** pristupa se korišćenjem sledeće linije koda:

- a. string a= stringLista(1);
- b. string a= stringLista[1];
- c. string a= stringLista.indexOf(1);

Odgovor: b

342

Generički rečnici

Dictionary

- Dictionary je kolekcija koja pridružuje ključ (key) nekoj vrednosti (value)
- Dictionary< TKey, TValue >
- Pronalaženje vrednosti korišćenjem njemu pridruženog ključa je veoma brzo
- Elementi rečnika su članovi
KeyValuePair< TKey, TValue > strukture

Svojstva i metode kolekcije Dictionary<Tkey, Tvalue>

Method or property	Purpose
Count	Public property that gets the number of elements in the Dictionary.
Item()	The indexer for the Dictionary.
Keys	Public property that gets a collection containing the keys in the Dictionary. (See also Values, later in this table.)
Values	Public property that gets a collection containing the values in the Dictionary. (See also Keys, earlier in this table.)
Add()	Adds an entry with a specified Key and Value.
Clear()	Removes all objects from the Dictionary.
ContainsKey()	Determines whether the Dictionary has a specified key.
ContainsValue()	Determines whether the Dictionary has a specified value.
GetEnumerator()	Returns an enumerator for the Dictionary.
Remove()	Removes the entry with the specified Key.

345

Štampanje sadržaja rečnika

```
public void Stampaj<Tkey,Tvalue>(Dictionary<Tkey, Tvalue> dict)
{
    richTextBox1.Clear();

    foreach (KeyValuePair<Tkey, Tvalue> red in dict)
    {
        string s = string.Format("Ključ: {0}, Vrednost: {1}\n", red.Key, red.Value);

        richTextBox1.AppendText(s);
    }
}
```

346

Upotreba kolekcije Dictionary-2

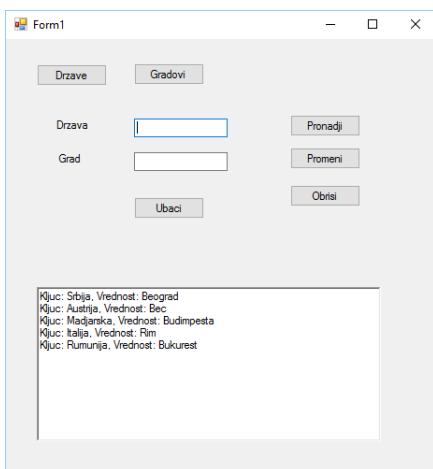
```
private Dictionary<string, string> recnik = new Dictionary<string, string>();
```

```
private void Form1_Load(object sender, EventArgs e)
{
    recnik.Add("Srbija", "Beograd");
    recnik.Add("Austrija", "Bec");
    recnik.Add("Madjarska", "Budimpesta");
    recnik.Add("Italija", "Rim");
    recnik.Add("Rumunija", "Bukurest");

    Stampaj(recnik);
}
```

347

Korisnički interfejs



348

Pristup ključevima rečnika

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();

    foreach (string s in recnik.Keys)
    {
        richTextBox1.AppendText(s + "\n");
    }
}
```

349

Pristup vrednostima rečnika

```
private void button2_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();

    foreach (string s in recnik.Values)
    {
        richTextBox1.AppendText(s + "\n");
    }
}
```

350

String funkcija

```
public string VelikoPrvoSlovo(string rec)
{
    rec = rec.Trim().ToLower();
    if (rec.Length > 1)
    {
        return rec.Substring(0, 1).ToUpper() + rec.Substring(1);
    }
    return string.Empty;
}
```

351

Unos reda u rečnik

```
private void buttonUbaci_Click(object sender, EventArgs e)
{
    string drzava = VelikoPrvoSlovo(textBoxDrzava.Text);
    string grad = VelikoPrvoSlovo(textBoxGrad.Text);

    if (!recnik.ContainsKey(drzava))
    {
        recnik.Add(drzava, grad);
        Stampaj(recnik);
    }
    else
    {
        MessageBox.Show("Država se već nalazi u recniku");
    }

    textBoxDrzava.Clear();
    textBoxGrad.Clear();
    textBoxDrzava.Focus();
}
```

352

Pristup vrednosti na osnovu ključa

```
private void buttonPronadji_Click(object sender, EventArgs e)
{
    string drzava = VelikoPrvoSlovo(textBoxDrzava.Text);

    if (recnik.ContainsKey(drzava))
    {
        textBoxGrad.Text = recnik[drzava];
    }
    else
    {
        MessageBox.Show("Drzava se ne nalazi u recniku");
        textBoxDrzava.Clear();
        textBoxDrzava.Focus();
    }
}
```

353

Brisanje člana rečnika

```
private void buttonObrisni_Click(object sender, EventArgs e)
{
    string drzava = VelikoPrvoSlovo(textBoxDrzava.Text.Trim());

    if (recnik.ContainsKey(drzava))
    {
        recnik.Remove(drzava);
        Stampaj(recnik);
    }
}
```

354

Pitanje 1

Red generičkog rečnika Dictionary<int, string> je :

- a. vrednost tipa KeyValuePair<int, string>
- b. vrednost tipa string
- c. vrednost tipa int

Odgovor: a

355

Pitanje 2

Za brisanje člana sa ključem **k** iz rečnika **dict** koristi se naredba:

- a. dict.Delete(k);
- b. dict[k].Delete();
- c. dict.Remove(k);

Odgovor: c

356

Pitanje 3

Ako je definisan sledeći kod:

```
Dictionary<int, string> dict = new Dictionary<int, string>();  
int a = 0;  
string b = "";  
if (dict.ContainsKey(a))  
{  
    //???  
}
```

koja linija koda se može napisati u bloku if

- a. b = dict[a];
- b. a = dict[b];
- c. a= dict(a);
- d. b=dict(a);

Odgovor: a

357

Lambda izrazi

Delegat Func<T,TResult>

- Delegat je tip koji predstavlja referencu na metodu
- Delegat **Func<TResult>** je pokazivač na metodu bez ulaznih parametar
- Delegat **Func<T,TResult>** je pokazivač na metodu koja ima jedan ulazni parametar tipa T i vraća rezultat tipa TResult
- Delegat **Func<T1,T2,TResult>** je pokazivač na metodu sa dva ulazna parametra tipa T1 i T2 koja ima povratnu vrednost tipa TResult

359

Lambda izrazi

- Anonimna metoda je metoda koja nema ime, može imati ulazne parametre i može vraćati vrednost
- Lambda izrazi omogućavaju kreiranje anonimnih metoda
- Lambda izrazi se definišu korišćenjem => operatora
- Lambda izrazi imaju prirodnu i preciznu sintaksu

360

Primer Lambda izraza

$x \Rightarrow x * x$

Za dato x, izračunaj $x * x$

Visual C# izvodi povratni tip na osnovu tela metode, konteksta

```
private void Button1_Click(object sender, EventArgs e)
{
    Func<double, double> f1 = x => x * x;

    double rezultat = f1(5);

    MessageBox.Show(rezultat.ToString());
}
```

361

Lambda izraz za metodu sa dva ulazna parametra

```
private void Button2_Click(object sender, EventArgs e)
{
    Func<int, int, int> f1 = (x, y) => x + y;
    int rezultat = f1(5, 6);

    MessageBox.Show(rezultat.ToString());
}
```

362

Lambda izraz za metodu bez parametara

```
private void Button3_Click(object sender, EventArgs e)
{
    Func<string> f1 = () => DateTime.Now.ToShortDateString();

    string rezultat = f1();

    MessageBox.Show(rezultat);
}
```

363

List<T>.ForEach(Action<T>) metoda

```
private void button1_Click(object sender, EventArgs e)
{
    List<string> imena = new List<string>
    {
        "Marko",
        "Ana",
        "Lazar",
        "Dejan"
    };

    imena.ForEach(i => richTextBox1.AppendText(i + "\n"));
}
```

Delegat Action<T> može da pokazuje na metodu sa ulaznim parametrom tipa T koja nema povratnu vrednost

364

Ekstenzione metode

- Ektenziona metoda omogućava da dodajemo metode u postojeću klasu bez potrebe da kreiramo izvedenu klasu
- To su statičke metode ali se pozivaju kao da su nestatičke metode klase koji proširujemo
- Definiše se statička klasa koja će sadržati ekstenzione metode
- Klasa mora biti vidljiva za kod koji će je pozivati

365

Kreiranje ekstenzione metode

- Ekstenziona metoda je statička metoda sa najmanje istom vidljivošću kao i klasa koja je sadrži
- Prvi parametar ekstenzione metode je tip sa kojim metoda radi ispred koga se nalazi reč this
- Ovakva metoda se poziva kao da je instance metoda tipa koga proširuje

366

Primer ekstenzione metode koja proširuje klasu string

```
static class EkstenzioneMetode
{
    public static string VelikoPrvoSlovo(this string s)
    {
        s = s.Trim().ToLower();
        if (s.Length > 1)
        {
            return s.Substring(0, 1).ToUpper() + s.Substring(1);
        }
        return string.Empty;
    }
}
```

367

Poziv ektenzione metode

```
private void button1_Click(object sender, EventArgs e)
{
    string s1 = "MARKO";
    string ime = s1.VelikoPrvoSlovo();
    richTextBox1.Text = ime;
}
```

Marko

368

Klasa Enumerable

```
public static class Enumerable
```

- Nalazi se u prostoru imena System.Linq
- TSource generički tip koga sadrži izvor podataka
- Statičke metode klase Enumerable su ekstenziona metode koje proširuju izvor podataka koji implementira `IEnumerable<TSource>` interfejs
- Statičke metode klase Enumerable omogućavaju da se pišu upiti nad bilo kojim izvorom podataka koji primenjuje interfejs `IEnumerable<T>`

369

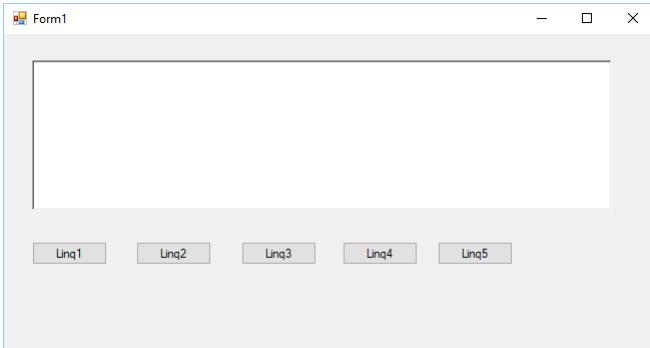
Ekstenziona metoda Select klase Enumerable

```
public static IEnumerable<TResult> Select<TSource, TResult>(
    this IEnumerable<TSource> source,
    Func<TSource, TResult> selector
)
```

- Ekstenziona metoda **Select** proširuje tip `IEnumerable<TSource>`
- **source** je izvor podataka koji imlementira interfejs `IEnumerable`
- **selector** je anonimna transformaciona funkcija koja se primenjuje nad svakim članom u izvoru podataka
- Povratna vrednost je sekvenca nastala primenom transformacione funkcije nad izvorom podataka

370

Korisnički interfejs



371

Primer upotrebe ekstenzije metode Select()

```
private void Button1_Click(object sender, EventArgs e)
{
    int[] brojevi = { 1, 3, 5, 15, 9, 24 };

    IEnumerable<int> kvadrati = brojevi.Select(x => x * x);
    StringBuilder sb = new StringBuilder();
    foreach (int i in kvadrati)
    {
        sb.AppendLine(i.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

372

Ekstenziona metoda Where klase Enumerable

```
public static IEnumerable<TSource> Where<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

- Where ekstenziona metoda proširuje tip **IEnumerable<TSource>**
- **source** je izvor podataka koji implementira interfejs **IEnumerable**
- Where ekstenziona metoda kao parametar ima instancu delegata **Func<TSource, bool>**, ova instanca je označena sa **predicate**
- Parametar **predicate** može biti bilo kakav lambda izraz (anonimna funkcija) sa ulazom tipa **TSource** i izlazom tipa **bool**

373

Primer upotrebe Where ekstenzione metode -1

```
private void Button2_Click(object sender, EventArgs e)
{
    int[] brojevi = { 0, 1, 2, 3, 4, 5, 6 };

    // definisanje upita
    IEnumerable<int> upit = brojevi.Where(n => n % 2 == 0);

    StringBuilder sb = new StringBuilder();

    foreach (int i in upit)
    {
        sb.AppendLine(i.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

374

Primer upotrebe Where ekstenzije metode -2

```
private void Button3_Click(object sender, EventArgs e)
{
    string[] imena = { "Marko", "Lazar", "Ivan", "Marija", "Jovana", "Jovan", "Desimir" };

    IEnumerable<string> upit = imena
        .Where(s => s.StartsWith("m", StringComparison.CurrentCultureIgnoreCase));

    StringBuilder sb = new StringBuilder();

    foreach (string n in upit)
    {
        sb.AppendLine(n);
    }

    richTextBox1.Text = sb.ToString();
}
```

375

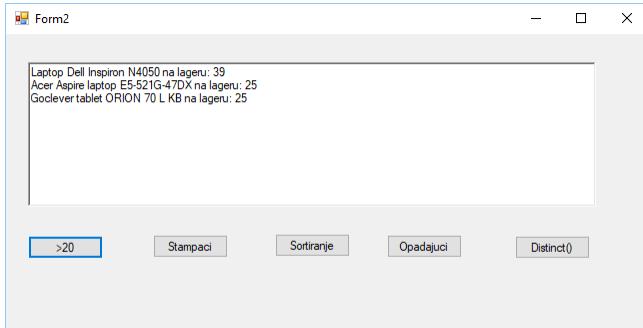
Kombinovanje metoda Where() i Select()

```
private void Button4_Click(object sender, EventArgs e)
{
    string[] imena = { "Aca", "Pera", "Mika", "Lazar", "Ivana", "Jovan", "Jovana", "Djordje" };
    IEnumerable<string> upit = imena
        .Where(s => s.Length == 5)
        .Select(s => s.ToUpper());
    StringBuilder sb = new StringBuilder();

    foreach (string clan in upit)
    {
        sb.AppendLine(clan);
    }
    richTextBox1.Text = sb.ToString();
}
```

376

Linq upiti nad kolekcijama objekata



377

Klasa Kategorija

```
class Kategorija
{
    public int KategorijaId { get; set; }
    public string NazivKategorije { get; set; }
    public string OpisKategorije { get; set; }
}
```

378

Klasa Proizvod

```
class Proizvod
{
    public int ProizvodId { get; set; }
    public int KategorijaId { get; set; }
    public string NazivProizvoda { get; set; }
    public decimal Cena { get; set; }
    public int KolicinaNaLageru { get; set; }
}
```

379

Metoda VratiKategorije(), statička klasa Podaci

```
public static List<Kategorija> VratiKategorije()
{
    List<Kategorija> listaKategorija = new List<Kategorija>() {
        new Kategorija{KategorijaId=1,NazivKategorije="Laptopovi", OpisKategorije="Laptopovi razlicitih proizvodjaca" },
        new Kategorija{KategorijaId=2,NazivKategorije="Stampaci", OpisKategorije="Stampaci razlicitih proizvodjaca" },
        new Kategorija{KategorijaId=3,NazivKategorije="Tableti", OpisKategorije="Tableti racunari razlicitih proizvodjaca" }
    };
    return listaKategorija;
}
```

380

Metoda VratiProizvode() statička klasa Podaci

```
public static List<Proizvod> VratiProizvode()
{
    List<Proizvod> listaProizvoda = new List<Proizvod>() {
        new Proizvod{ProizvodId=1, KategorijaId=1,NazivProizvoda="Laptop Dell Inspiron N4050",Cena=30999.25M,KolicinaNaLageru=39 },
        new Proizvod{ProizvodId=2, KategorijaId=1,NazivProizvoda="Laptop Asus X55U-SX009D",Cena=32990.12M,KolicinaNaLageru=17 },
        new Proizvod{ProizvodId=3, KategorijaId=1,NazivProizvoda="Acer Aspire laptop E5-521G-47DX",Cena=41989,KolicinaNaLageru=25 },

        new Proizvod{ProizvodId=4, KategorijaId=2,NazivProizvoda="Stampać Laser A4 Lexmark E260",Cena=8549.1M,KolicinaNaLageru=13 },
        new Proizvod{ProizvodId=5, KategorijaId=2,NazivProizvoda="Canon laserski stampac LBP-6670DN",Cena=31989.1M,KolicinaNaLageru=18 },
        new Proizvod{ProizvodId=6, KategorijaId=2,NazivProizvoda="Canon stampać imageCLASS LBP6030W",Cena=13589.12M,KolicinaNaLageru=11 },

        new Proizvod{ProizvodId=7, KategorijaId=3,NazivProizvoda="Acer tablet B1-730HD 8GB",Cena=11999.3M,KolicinaNaLageru=12 },
        new Proizvod{ProizvodId=8, KategorijaId=3,NazivProizvoda="Asus tablet MeMO Pad 7 ME70C-1A003A",Cena=12999.9M,KolicinaNaLageru=14 },
        new Proizvod{ProizvodId=9, KategorijaId=3,NazivProizvoda="Goclever tablet ORION 70 L KB",Cena=5699.45M,KolicinaNaLageru=25 }
    };
    return listaProizvoda;
}
```

381

Inicijalizacija izvora podataka

```
private List<Kategorija> listaKategorija = Podaci.VratiKategorije();
private List<Proizvod> listaProizvoda = Podaci.VratiProizvode();
```

382

Proizvodi kojih na lageru ima više od 20

```
private void Button1_Click(object sender, EventArgs e)
{
    IEnumerable<Proizvod> upit = listaProizvoda
        .Where(p => p.KolicinaNaLageru > 20);

    StringBuilder sb = new StringBuilder();

    foreach (Proizvod p1 in upit)
    {
        sb.AppendLine("Naziv: " + p1.NazivProizvoda);
        sb.AppendLine("Cena: " + p1.Cena);
        sb.AppendLine("Kolicina: " + p1.KolicinaNaLageru);
        sb.AppendLine("").PadRight(100, '.'));
    }

    richTextBox1.Text = sb.ToString();
}
```

383

Ekstenziona metoda SingleOrDefault() klase Enumerable

```
public static TSource SingleOrDefault<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

Vraća jedinstveni element sekvene koji zadovoljava uslov.
Vraća podrazumavanu vrednost za element u izvoru podataka
ukoliko ne postoji element.

384

Metoda SingleOrDefault() upotreba

```
private void Button2_Click(object sender, EventArgs e)
{
    Proizvod p1 = listaProizvoda.SingleOrDefault(p => p.ProizvodId == 1);

    if (p1 != null)
    {
        label1.Text = p1.NazivProizvoda;
    }
    else
    {
        MessageBox.Show("Ne postoji proizvod");
    }
}
```

385

Metoda FirstOrDefault()

```
public static TSource FirstOrDefault<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

Vraća prvi element sekvene koji zadovoljava uslov ili podrazumevanu vrednost ukoliko takav element ne postoji

386

Metoda FirstOrDefault upotreba

```
private void Button3_Click(object sender, EventArgs e)
{
    Proizvod p1 = listaProizvoda.FirstOrDefault(p => p.KategorijaId == 1);
    if (p1 != null)
    {
        label1.Text = p1.NazivProizvoda;
    }
    else
    {
        MessageBox.Show("Ne postoji proizvod");
    }
}
```

387

Metoda OrderBy

```
public static IOrderedEnumerable<TSource> OrderBy<TSource, TKey>(
    this IEnumerable<TSource> source,
    Func<TSource, TKey> keySelector
)
```

Sortira elemente sekvence u rastućem poretku prema ključu
keySelektor - anonimna funkcija koja izdvaja ključ iz svakog elementa sekvence

388

Sortiranje rezultata

```
private void Button4_Click(object sender, RoutedEventArgs e)
{
    Ienumerable<Proizvod> sortiraniProizvodi = listaProizvoda
        .Where(p => p.KategorijaId == 1)
        .OrderBy(p => p.Cena);

    StringBuilder sb = new StringBuilder();
    foreach (Proizvod p in sortiraniProizvodi)
    {
        sb.AppendLine(p.NazivProizvoda + " " + p.Cena);
    }
    richTextBox1.Text = sb.ToString();
}
```

389

Sortiranje u opadajućem poretku

```
private void Button5_Click(object sender, RoutedEventArgs e)
{
    Ienumerable<Proizvod> sortiraniProizvodi = listaProizvoda
        .Where(p => p.KategorijaId == 1)
        .OrderByDescending(p => p.Cena);

    StringBuilder sb = new StringBuilder();
    foreach (Proizvod p in sortiraniProizvodi)
    {
        sb.AppendLine(p.NazivProizvoda + " " + p.Cena);
    }
    richTextBox1.Text = sb.ToString();
}
```

390

Višestruko sortiranje

```
private void Button9_Click(object sender, EventArgs e)
{
    Ienumerable<Proizvod> upit = listaProizvoda
        .OrderBy(p => p.KategorijaId)
        .ThenByDescending(p => p.Cena);

    StringBuilder sb = new StringBuilder();
    foreach (Proizvod p in upit)
    {
        sb.AppendLine(p.KategorijaId + " " + p.NazivProizvoda + " " + p.Cena);
    }
    richTextBox1.Text = sb.ToString();
}
```

391

Metoda Distinct()

Vraća jedinstvene elemente sekvence korišćenjem podrazumevanog komparatora jednakosti.

```
private void Button1_Click(object sender, EventArgs e)
{
    List<int> brojevi = new List<int> { 21, 46, 46, 55, 17, 21, 55, 55 };
    Ienumerable<int> razlicitiBrojevi = brojevi.Distinct();

    StringBuilder sb = new StringBuilder();
    foreach (int b in razlicitiBrojevi)
    {
        sb.AppendLine(b.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

392

Metoda Count()

```
public static int Count<TSource>(
    this IEnumerable<TSource> source
)
```

```
public static int Count<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

393

Metoda Count()

```
private void Button2_Click(object sender, EventArgs e)
{
    List<int> brojevi = new List<int>{ 5, 4, 1, 3, 9, 8, 6, 7, 1, 0 };
    int ukupno = brojevi.Count();
    int neparnih = brojevi.Count(b => b % 2 == 1);
    int parnih = brojevi.Count(b => b % 2 == 0);

    StringBuilder sb = new StringBuilder();
    sb.AppendLine("Ukupno: " + ukupno.ToString());
    sb.AppendLine("Neparnih: " + neparnih.ToString());
    sb.AppendLine("Parnih: " + parnih.ToString());
    richTextBox1.Text = sb.ToString();
}
```

394

Metoda Min()

```
public static int Min(  
    this IEnumerable<int> source  
)
```

```
public static decimal Min<TSource>(  
    this IEnumerable<TSource> source,  
    Func<TSource, decimal> selector  
)
```

selector - transformaciona funkcija koja svaku vrednost sekvence pretvara u decimal

395

Metoda Max()

```
public static int Max(  
    this IEnumerable<int> source  
)
```

```
public static decimal Max<TSource>(  
    this IEnumerable<TSource> source,  
    Func<TSource, decimal> selector  
)
```

396

Primer Min()/Max()

```
private void Button3_Click(object sender, EventArgs e)
{
    int[] brojevi = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
    StringBuilder sb = new StringBuilder();

    int minBroj = brojevi.Min();
    sb.AppendLine(minBroj.ToString());

    decimal najJeftiniji = listaProizvoda.Min(p => p.Cena);
    sb.AppendLine(najJeftiniji.ToString());

    richTextBox1.Text = sb.ToString();
}
```

397

Metoda Average()

```
private void Button4_Click(object sender, EventArgs e)
{
    List<int> brojevi = new List<int> { 78, 92, 100, 37, 81 };

    StringBuilder sb = new StringBuilder();

    double prosek = brojevi.Average();
    sb.AppendLine(prosek.ToString());

    decimal prosecnaCena = listaProizvoda.Average(p => p.Cena);
    sb.AppendLine(prosecnaCena.ToString());

    richTextBox1.Text = sb.ToString();
}
```

398

Metoda Sum()

```
private void Button5_Click(object sender, RoutedEventArgs e)
{
    StringBuilder sb = new StringBuilder();

    List<int> brojevi = new List<int> { 78, 92, 100, 37, 81 };
    int suma = brojevi.Sum();
    sb.AppendLine(suma.ToString());

    int ukupnoLaptopova = listaProizvoda
        .Where(p => p.KategorijaId == 1)
        .Sum(p => p.KolicinaNaLageru);

    sb.AppendLine(ukupnoLaptopova.ToString());

    richTextBox1.Text = sb.ToString();
}
```

399

Metoda Take()

```
public static IEnumerable<TSource> Take<TSource>(
    this IEnumerable<TSource> source,
    int count
)
```

Vraća specificirani broj uzastopnih elemenata od početka sekvence.

400

Metoda Take() - primer

```
private void Button6_Click(object sender, EventArgs e)
{
    int[] poeni = { 59, 82, 70, 56, 92, 98, 85 };

    IEnumerable<int> najbolja3 =
        poeni.OrderByDescending(p => p).Take(3);

    StringBuilder sb = new StringBuilder();
    foreach (int p in najbolja3)
    {
        sb.AppendLine(p.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

401

Metoda Skip()

Preskače određeni broj elemenata sa početka sekvence.

```
private void Button7_Click(object sender, EventArgs e)
{
    int[] poeni = { 59, 82, 70, 56, 92, 98, 85 };

    IEnumerable<int> preskociPrvaTri =
        poeni.OrderByDescending(p => p).Skip(3);

    StringBuilder sb = new StringBuilder();
    foreach (int p in preskociPrvaTri)
    {
        sb.AppendLine(p.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

402

Pitanje 1

Delegat je tip koji:

- a. pokazuje na metodu
- b. pokazuje na klasu
- c. pokazuje na svojstvo

Odgovor: a

403

Pitanje 2

Pomoću lambda izraza definiše se:

- a. anonimna metoda
- b. sql upit
- c. metoda koja ima svoje ime

Odgovor: a

404

Pitanje 3

Za pisanje upita primenom lambda izraza koriste se

- a. Metode klase Linq
- b. Ekstenzije metode klase Enumerable
- c. Metode interfejsa IEnumerable

Odgovor: b

405

Pitanje 4

Ako je sa TSource označen tip podataka u izvoru podataka. Ekstenzije metode statičke klase Enumerable proširuju :

- a. Izvor podataka koji implementira IEnumerable<TSource> interfejs
- b. Izvor podataka koji implementira ISqlSource<TSource> interfejs
- c. Izvor podataka koji implementira IComparable<TSource> interfejs

Odgovor: a

406

Kontrole ComboBox i ListBox

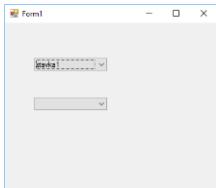
ComboBox kontrola

- Svojstvo **Items** daje kolekciju stavki ComboBox kontrole
- **SelectedIndex** svojstvo vraća indeks selektovane stavke kombo boksa
 - int selectedIndex = comboBox1.SelectedIndex;
- **SelectedItem** vraća selektovanu stavku kombo boksa koja je tipa object
 - object selectedItem = comboBox1.SelectedItem;
- Dodavanje stavki u kombo boks:
 - comboBox1.Items.Add(stavka1);
 - comboBox1.Items.AddRange(new object[] = {"stavka1", "stavka2", "stavka3"});

Dodavanje stavki iz koda

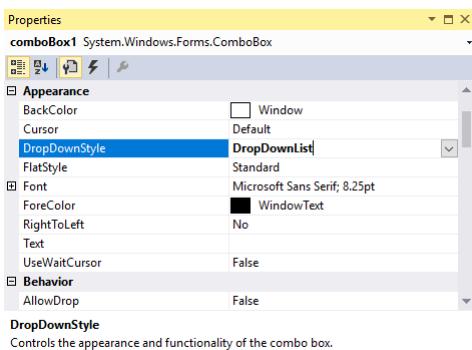
```
private void Form1_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add("stavka1");
    comboBox1.Items.Add("stavka2");
    comboBox1.Items.Add("stavka3");
    comboBox1.SelectedIndex = 0;

    comboBox2.Items.AddRange
        (new object[]{"stavka1", "stavka2", "stavka3", "stavka4"});
}
```



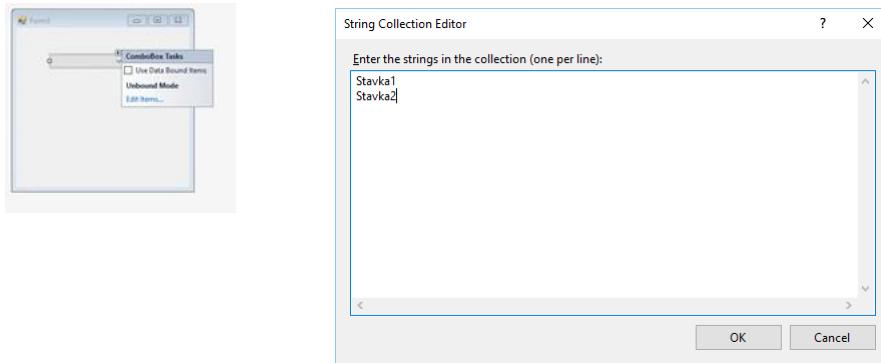
409

Svojstvo DropDownList



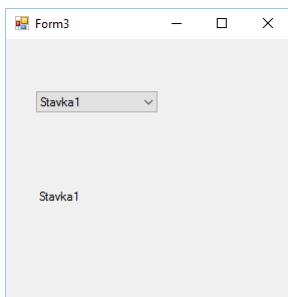
410

Dodavanje stavki u ComboBox u dizajn modu



411

SelectedIndexChanged događaj



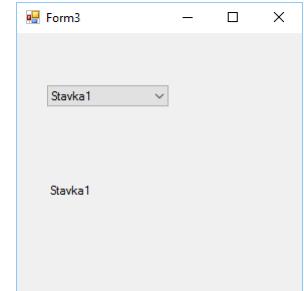
```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label1.Text = comboBox1.SelectedItem.ToString();
}
```

412

Povezivanje sa nizom

```
private void Form3_Load(object sender, EventArgs e)
{
    string[] stavke = {"Stavka1", "Stavka2", "Stavka3", "Stavka4" };
    comboBox1.DataSource = stavke;
    comboBox1.SelectedIndex = -1;
}
```

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex > -1)
    {
        label1.Text = comboBox1.SelectedItem.ToString();
    }
    else
    {
        label1.Text = "";
    }
}
```



413

Povezivanje sa rečnikom-1

```
private void Form4_Load(object sender, EventArgs e)
{
    Dictionary<int, string> recnik = new Dictionary<int, string>();
    recnik.Add(1, "Vrednost1");
    recnik.Add(2, "Vrednost2");
    recnik.Add(3, "Vrednost3");

    BindingSource bs = new BindingSource();
    bs.DataSource = recnik;

    comboBox1.DataSource = bs;
    comboBox1.DisplayMember = "Value";
    comboBox1.SelectedIndex = -1;
}
```

414

Povezivanje sa rečnikom-2

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex > -1)
    {
        KeyValuePair<int, string> selektovano =
            (KeyValuePair<int, string>)comboBox1.SelectedItem;

        label1.Text = selektovano.Key + " " + selektovano.Value;
    }
    else
    {
        label1.Text = "";
    }
}
```

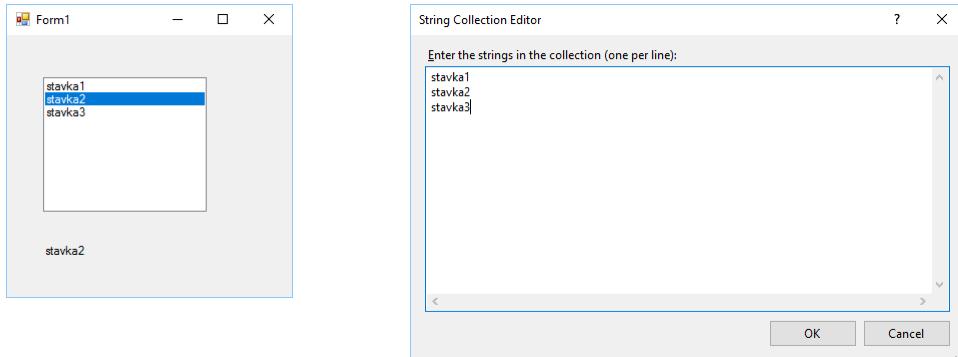
415

Svojstva ListBox kontrole

- **SelectedIndices** svojstvo vraća kolekciju selektovanih indeksa
- **SelectedIndex**
- **SelectionMode** svojstvo određuje koliko stavki može biti selektovanano u ListBox kontroli
 - MultiSimple omogućava selektovanje više stavki
 - MultiExtended omogućava korišćenje tastera SHIFT
- **SelectedItems** vraća kolekciju selektovanih stavki
- **SelectedItem**

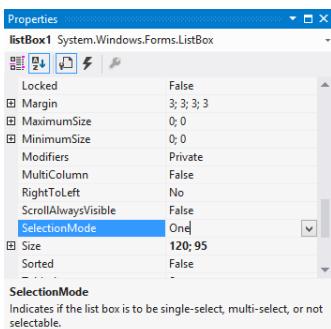
416

Dodavanje stavki



417

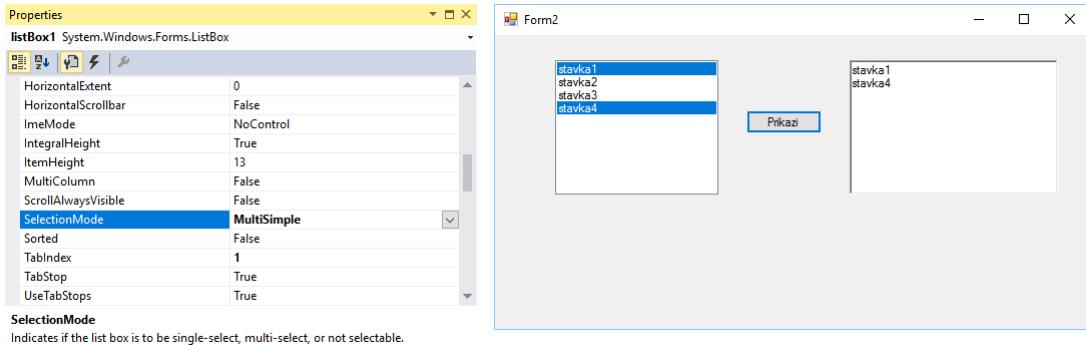
Događaj SelectedIndexChanged



```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label1.Text = listBox1.SelectedItem.ToString();
}
```

418

ListBox kontrola - MultiSimple



419

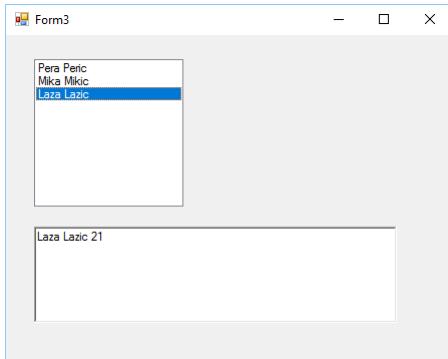
ListBox kontrola sa višestrukom selekcijom

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();

    foreach (string stavka in listBox1.SelectedItems)
    {
        richTextBox1.AppendText(stavka + "\n");
    }
}
```

420

Korisnički interfejs



421

Klasa Osoba

```
class Osoba
{
    public int OsobaId { get; set; }
    public string Ime { get; set; }

    public string Prezime { get; set; }
    public int Starost { get; set; }

    public string PunoIme
    {
        get
        {
            return Ime + " " + Prezime;
        }
    }
    //public override string ToString()
    //{
    //    return Ime + " " + Prezime;
    //}
}
```

422

Load procedura forme

```
private void Form3_Load(object sender, EventArgs e)
{
    Osoba os1 = new Osoba {OsobaId=1, Ime="Pera", Prezime="Peric", Starost=24 };
    Osoba os2 = new Osoba { OsobaId = 2, Ime = "Mika", Prezime = "Mikic", Starost = 22 };
    Osoba os3 = new Osoba { OsobaId = 3, Ime = "Laza", Prezime = "Lazic", Starost = 21 };

    listBox1.Items.Add(os1);
    listBox1.Items.Add(os2);
    listBox1.Items.Add(os3);

    listBox1.DisplayMember = "PunoIme";
    listBox1.ValueMember = "OsobaId";
    listBox1.SelectedIndex = -1;
}
```

423

Čitanje selektovane stavke

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listBox1.SelectedIndex >-1)
    {
        Osoba selektovano = (Osoba)listBox1.SelectedItem;
        richTextBox1.Text = selektovano.Ime + " " + selektovano.Prezime + " " + selektovano.Starost;
    }
    else
    {
        richTextBox1.Clear();
    }
}
```

424

Pitanje 1

Neka je comboBox1 instanca kontrole ComboBox. Nakon izvršavanja naredbe:
string selektovano = comboBox1.SelectedItem;

- a. promenljiva selektovano dobija vrednost koja je indeks selektovane stavke
- b. promenljiva selektovano dobija vrednost koja je vrednost selektovane stavke
- c. generiše se izuzetak jer nije izvršeno odgovarajuće kastovanje

Odgovor: c

425

Pitanje 2

Stavke ComboBox kontrole su tipa:

- a. string
- b. object
- c. double

Odgovor: b

426

Pitanje 3

Ako želimo da osiguramo da nijedna stavka ComboBox kontrole ne bude selektovana, onda pišemo:

- a. comboBox1.SelectedIndex = 0;
- b. comboBox1.SelectedIndex = -1;
- c. comboBox1.SelectedItem = -1;

Odgovor: b

427

Pitanje 4

Neka je instanca ListBox kontrole pod nazivom listBox1 povezana sa generičkom listom objekata klase Osoba pomoću svojstva DataSource. Ako je Punolme svojstvo klase Osoba, na koji način se vrednost ovog svojstva prikazuje na mestu stavke ListBox kontrole

- a. listBox1.BindingMember = "Punolme";
- b. listBox1.DisplayMember = "Punolme";
- c. listBox1.ValueMember = "Punolme";
- d. listBox1.ValueMember = Punolme;

Odgovor: b

428

Pitanje 5

Kreirana je WindowsForms aplikacija u kojoj se koristi ListBox kontrola u režimu jednostrukih selekcija. U Load procedure forme napisan je sledeći kod:

```
private void Form1_Load(object sender, EventArgs e)
{
    string[] stavke = { "stavka1", "stavka2", "stavka3", "stavka4" };
    listBox1.DataSource = stavke;
}
```

Pokretanjem forme:

- a. neće biti selektovana ni jedna stavka
- b. biće selektovana prva stavka
- c. biće selektovana poslednja stavka

Odgovor: b

429

Pitanje 6

Način selekcije stavki u ListBox kontroli definiše se korišćenjem:

- a. svojstva SelectionMode
- b. svojstva SelectionType
- c. svojstva Mode

Odgovor: a

430

Uvod u ADO.NET

ADO.NET

- ADO.NET je skup klasa za rad sa podacima
- .NET snabdevači podataka su klase koje obezbeđuju mogućnost konektovanja na izvor podataka
 - SQL Server snabdevač podataka
 - OLE DB snabdevač podataka
 - ostali snabdevači podataka
- Prostori imena za rad sa podacima su
 - System.Data
 - System.Data.SqlClient
 - System.Data.OleDb
 - System.Data.SqlTypes
 - System.Xml

Konektovani scenario

- Resursi se uzimaju sa servera sve dok se konekcija ne zatvori
- Korisnik je konstantno povezan na izvor podataka
- Podaci su ažurni
- Konkurentnost se lakše kontroliše
- Mora postojati konstantna mrežna konekcija

433

Korišćenje ADO.NET klasa u konektovanom scenariju



- Otvaranje konekcije
- Izvršavanje komande
- Zatvaranje konekcije

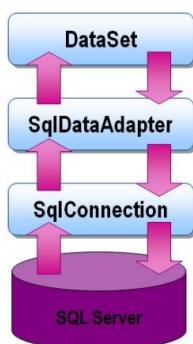
434

Diskonektovani scenario

- U diskonektovanom scenariju podskup podataka iz baze podataka se kopira lokalno
- Dok se korisnik nalazi u diskonektovanom radu ostali korisnici mogu da koriste konekciju
- Diskonektovani rad povećava skalabilnost aplikacije
- Podaci nisu uvek ažurni
- Kada se podacima iz lokalne kopije podataka ažurira baza može doći do konflikta

435

Korišćenje ADO.NET klasa u diskonektovanom scenariju



- Otvaranje konekcije
- Punjenje DataSet-a
- Zatvaranje konekcije
- Obrada podataka u DataSet-u
- Otvaranje konekcije
- Ažuriranje izvora podataka
- Zatvaranje konekcije

436

Konekcije

- Pre bilo kakvog rada sa bazom podataka potrebno je kreirati a zatim otvoriti konekciju
- U ADO.NET se kreira objekat klase XxxConnection
- System.Data.SqlClient.SqlConnection omogućava kreiranje konekcije na SQL Server bazu podataka
- System.Data.OleDb.OleDbConnection omogućava kreiranje konekcije na svaki izvor podataka sa pridruženim OLE DB provajderom

437

SqlConnection objekat

- Svaki SqlConnection objekat je izведен iz klase DbConnection koja se nalazi u prostoru imena System.Data.Common
- Klasa DbConnection predstavlja konekciju na bazu podataka

```
public abstract class DbConnection : Component, IDbConnection, IDisposable
```

438

Svojstva klase DbConnection

- **ConnectionString** svojstvo definiše string koji se koristi za otvaranje konekcije
- Svojstvo **DataSource** specificira ime database servera na koji se vrši konekcija
- Svojstvo **Database** specificira ime baze na koju se vrši konekcija
- Svojstvo **State** specificira stanje konekcije
- Svojstvo **ConnectionTimeout** specificira vreme dozvoljeno za uspostavljenje konekcije nakon čega se generiše poruka o grešci

439

Metode i događaji klase DbConnection

- Metoda **Open()** otvara konekciju na bazu podataka koristeći vrednosti specificirane u konekcionom stringu
- Metoda **Close()** zatvara konekciju sa bazom podataka i oslobađa sistemske resurse
- Metoda **BeginTransaction()** otpočinje transakciju nad bazom
- Događaj **StateChange** se trigeruje kada se stanje konekcije menja (npr. iz otvorenog u zatvoreno)

440

Baza Magacin

```

CREATE DATABASE Magacin
GO

USE Magacin
GO

CREATE TABLE Kategorija
(
    KategorijaId int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    NazivKategorije nvarchar(50) NOT NULL UNIQUE,
    OpisKategorije nvarchar(100) NOT NULL
)

CREATE TABLE Proizvod
(
    ProizvodId int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    KategorijaId int NOT NULL FOREIGN KEY REFERENCES Kategorija(KategorijaId),
    NazivProizvoda nvarchar(50) NOT NULL,
    Cena decimal(12,3) NOT NULL,
    KolicinaNaLageru int NOT NULL,
)

```

441

Konekcija kod Windows autentifikacije

```

using System.Data.SqlClient;

SqlConnection con = new SqlConnection();
con.ConnectionString = "konekciona string";

SqlConnection con = new SqlConnection("konekciona string");

```

Windows autentifikacija:

```

string konekcionString = @"Data Source=(local)\SqlExpress;
Initial Catalog=Magacin;Integrated Security=true";

```

Data Source - ime SQL server-a
 Initial Catalog – ime baze sa kojom se radi

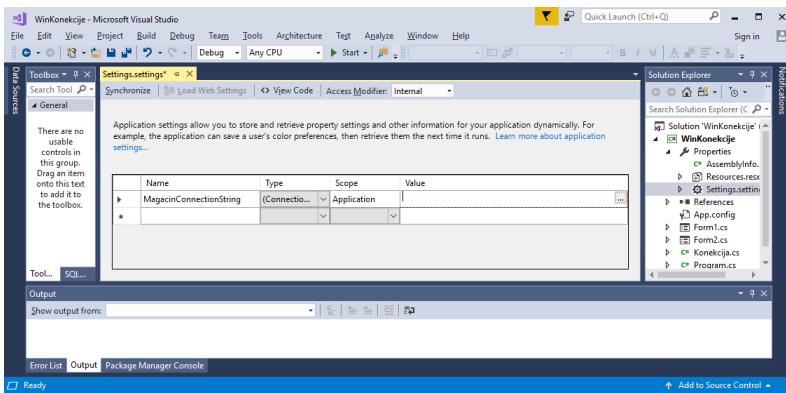
442

Konekcija kod SQLServer autentifikacije

```
//SQL Server autentifikacija
string konekcionistring = @"Data Source=(local)\SqlExpress;
Initial Catalog=Magacin;User ID=sa;Password=****";
```

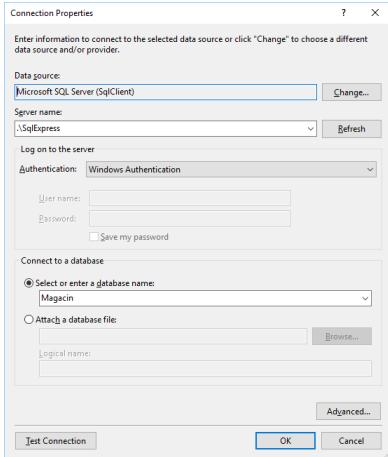
443

Upis konekcionog stringa u aplikaciona setovanja -1



444

Upis konekcionog stringa u aplikaciona setovanja -1



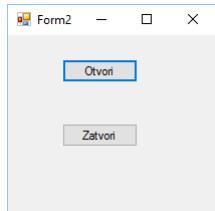
445

Kreiranje konekcije

```
static class Konekcija
{
    public static string cnnMagacin =
Properties.Settings.Default.MagacinConnectionString;
}
```

446

Primer uspostavljanja konekcije sa bazom



Napomena: primer ilustruje samo princip otvaranja i zatvaranja konekcije.
Vreme trajanja otvorene konekcije ne sme da zavisi od želje korisnika.

447

Otvaranje konekcije

```
private SqlConnection konekcija = new SqlConnection(Konekcija.cnnMagacin);
```

```
private void buttonOtvori_Click(object sender, EventArgs e)
{
    if (konekcija.State == ConnectionState.Open)
    {
        MessageBox.Show("Konekcija je vec otvorena", "Poruka");
    }
    else
    {
        konekcija.Open();
        MessageBox.Show(konekcija.State.ToString(), "Stanje konekcije");
    }
}
```

448

Zatvaranje konekcije

```
private void buttonZatvori_Click(object sender, EventArgs e)
{
    if (konekcija.State == ConnectionState.Closed)
    {
        MessageBox.Show("Konekcija je vec zatvorena");
    }
    else
    {
        konekcija.Close();
        MessageBox.Show(konekcija.State.ToString(), "Stanje konekcije");
    }
}
```

449

Pitanje 1

Pri koneksiom scenariju

- a. resursi se uzimaju a servera kada se konekcija zatvori
- b. resursi se uzimaju sa servera dok je konekcija otvorena
- c. kreira se lokalna kopija podataka iz baze

Odgovor: b

450

Pitanje 2

Za uspostavljanje konekcije na SQL Server bazu podataka koristi se objekat klase

- a. SqlConnection
- b. OleDbConnection
- c. SqlServerConnection

Odgovor: a

451

Pitanje 3

Atribut Data Source u konekcionom stringu definiše :

- a. bazu podataka sa kojom se uspostavlja konekcija
- b. tabelu u bazi sa kojom se uspostavlja konekcija
- c. database server sa kojim se uspostavlja konekcija

Odgovor: c

452

Pitanje 4

Pri diskonektovanom scenariju:

- a. kreira se lokalna kopija podataka iz baze
- b. korisnik je stalno povezan na izvor podataka

Odgovor: a

453

Pitanje 5

Da bi se koristio SQL Server.NET snabdevač podataka potrebno je ulkjučiti prostor imena:

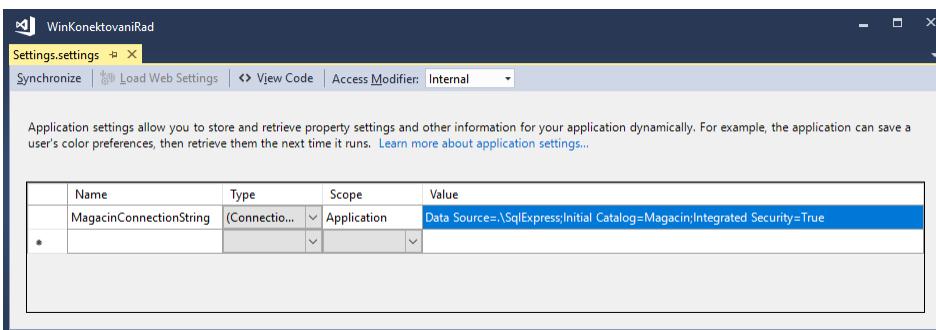
- a. System.Data.SqlTypes
- b. System.Data.SqlClient
- c. System.DataSqlServer

Odgovor: b

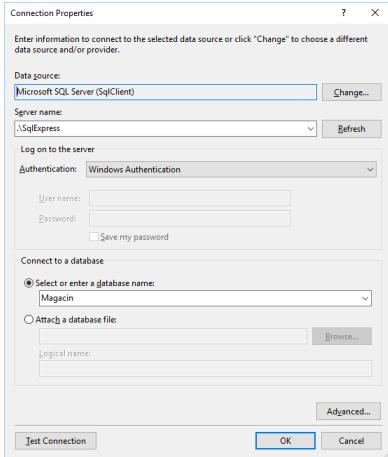
454

Rad u konektovanom okruženju

Upis konekcionog stringa u aplikaciona setovanja -1



Upis konekcionog stringa u aplikaciona setovanja -1



457

Klasa konekcija

```
static class Konekcija
{
    public static string cnnMagacin =
Properties.Settings.Default.MagacinConnectionString;
}
```

458

Objekat SqlCommand

- Koristi se za izvršavanje SQL komandi i uskladištenih procedura nad bazom podataka
- SqlCommand ili OleDbCommand
- Objekat SqlCommand omogućava direktni pristup podacima u bazi u konektovanom okruženju
- Pomoću njega moguće je izvršavati SELECT, INSERT, UPDATE i DELETE naredbe nad elementima baze podataka

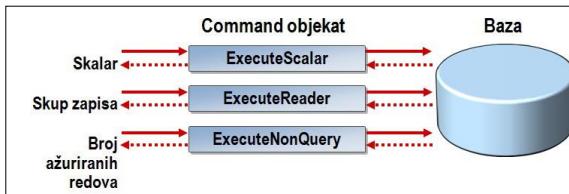
459

Svojstva SqlCommand objekta

- Svojstvo **CommandText** predstavlja tekst SQL komande ili naziv uskladištene procedure
- **CommandType** svojstvo se setuje na vrednost CommandType.Text ili na vrednost CommandType.StoredProcedure
- **Connection** svojstvo određuje objekat SqlConnection koji će SqlCommand objekat koristiti
- **Parameters** svojstvo daje kolekciju parametara SqlCommand objekta koji se koriste za izvršavanje parametarskih upita

460

Metode SqlCommand klase



461

Metode SqlCommand klase

- Metoda **ExecuteNonQuery()** izvršava izvršava SQL komande ili uskladištene procedure koje ne vraćaju vrednosti (INSERT, UPDATE, DELETE)
- Metoda **ExecuteReader()** izvršava komande koje vraćaju tabelarne podatke (SELECT)
- Metoda **ExecuteScalar()** se koristi za izvršavanje SQL komandi ili uskladištenih procedura koje vraćaju jednu skalarnu vrednost

462

Kreiranje i izvršavanje komande koja vraća sklarnu vrednost -1

```
private void button1_Click(object sender, EventArgs e)
{
    decimal cena = 0;
    string poruka = "";
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        SqlCommand komanda =
            new SqlCommand("SELECT AVG(Cena) FROM Proizvod", konekcija);
        try
        {
            konekcija.Open();
            cena = (decimal)komanda.ExecuteScalar();
        }
        catch (Exception xcp)
        {
            poruka = xcp.Message;
        }
    }
    ...
}
```

463

Kreiranje i izvršavanje komande koja vraća skalarnu vrednost -2

```
if (poruka != "")
{
    MessageBox.Show(poruka);
}
else
{
    MessageBox.Show("Prosecna cena je: " + cena);
}
```

464

Ažuriranje baze podataka -1

```
private void button2_Click(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder();
    sb.AppendLine("UPDATE Kategorija");
    sb.AppendLine("SET NazivKategorije = 'Racunari'");
    sb.AppendLine("WHERE KategorijaId =1");
    string greska = "";
    int promena = 0;

    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        try
        {
            SqlCommand komanda = new SqlCommand(sb.ToString(), konekcija);
            konekcija.Open();
            promena = komanda.ExecuteNonQuery();
        }
        catch (Exception xcp)
        {
            greska = xcp.Message;
        }
    }
    ...
}
```

465

Ažuriranje baze podataka -2

```
if (greska != "")
{
    MessageBox.Show(greska);
    return;
}
if (promena == 1)
{
    MessageBox.Show("Promenjena kategorija");
}
else
{
    MessageBox.Show("Nije promenjena kategorija");
}
```

466

SqlDataReader objekat

SqlDataReader

- SqlDataReader je brz (read-only,forward -only) kurzor koji se pomera kroz skup zapisa
- Pristup podacima korišćenjem objekta SqlDataReader sastoji se od sledećih koraka:
 - Kreira se objekat SqlConnection
 - Kreira se SqlCommand objekat sa odgovarajućim SELECT upitom
 - Otvara se konekcija
 - Izvršava se SqlCommand.ExecuteReader() metoda koja vraća objekat SqlDataReader
 - Koristeći Read() metodu objekta SqlDataReader prolazi se kroz sve vrste
 - Kada Read() metoda vrati false zatvara se konekcija

Svojstva i metode DataReader objekta

- Read() metoda
 - vrši se pozicioniranje na red
 - vraća true ako ima još redova za čitanje, false ukoliko se stiglo do poslednjeg reda
- Podrazumevana pozicija SqlDataReader-a je ispred prvog reda tabelekoju čitamo
- Pristup vrednosti u koloni određenog reda na koga SqlDataReader pokazuje
 - Vraćene vrednosti su u izvornom formatu i potrebno je izvršiti njihovo kastovanje da bi mogle da se koriste
 - dr["ImeKolone"] ili dr[PozicijaKolone].
- dr.GetDateTime(0), dr.GetDouble(0), dr.GetInt32(1) itd.

469

Entitetska klasa

```
class Kategorija
{
    public int KategorijaId { get; set; }
    public string NazivKategorije { get; set; }
    public string OpisKategorije { get; set; }

    public override string ToString()
    {
        return NazivKategorije;
    }
}
```

470

Prikaz podataka u ListBox kontroli

```

private void Form1_Load(object sender, EventArgs e)
{
    string greska = "";
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand("SELECT * FROM Kategorija", konekcija))
        {
            try
            {
                konekcija.Open();
                SqlDataReader dr = komanda.ExecuteReader();
                while (dr.Read())
                {
                    Kategorija k1 = new Kategorija
                    {
                        KategorijaId = dr.GetInt32(0),
                        NazivKategorije = dr.GetString(1),
                        OpisKategorije = dr.GetString(2)
                    };
                    listBox1.Items.Add(k1);
                }
            }
            catch (Exception xcp)
            {
                greska = xcp.Message;
            }
        }
    }
...
}

```

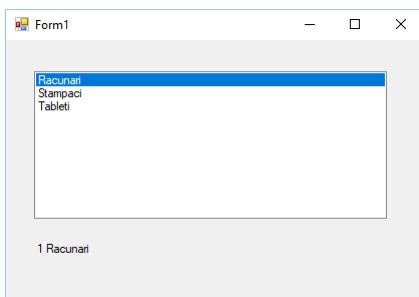
```

if (greska != "")
{
    MessageBox.Show(greska);
}
}

```

471

Korisnički interfejs



472

SelectedIndexChanged događaj

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listBox1.SelectedIndex > -1)
    {
        Kategorija k = listBox1.SelectedItem as Kategorija;
        label1.Text = k.KategorijaId + " " + k.NazivKategorije;
    }
}
```

473

Rad sa parametrizovanim upitima

Parametri u SQL komandama

- Parametri se mogu shvatiti kao promenljive koje se koriste za razmenu podataka između aplikacije i baze podataka
- Tip parametra se definiše korišćenjem standardne System.Data.SqlDbType enumeracije
- Tipičan primer korišćenja parametra je WHERE klauzula u SQL upitu

```
--parametarski SELECT upit
SELECT ProizvodId, NazivProizvoda, Cena
FROM Proizvod
WHERE KategorijaId = @KategorijaId
```

475

Izvršavanje parametarskog upita

```
DECLARE @KategorijaId INT
SET @KategorijaId = 1;

--parametarski SELECT upit
SELECT ProizvodId, NazivProizvoda,
Cena
FROM Proizvod
WHERE KategorijaId = @KategorijaId
```

476

Tipovi parametara

- Input – ulazni parametri se koriste u komandnim objektima za slanje podataka bazi
- Output – izlazni parametri se koriste za prihvatanje podataka od baze podataka nakon izvršavanja određenih upita
- InputOutput – ulazno/izlazni parametri opisuju i slanje i prihvatanje podataka

477

Primer parametarskog upita

```
DECLARE @NazivKategorije NVARCHAR(50)
SET @NazivKategorije = 'Stampaci';

--parametarski SELECT upit
SELECT ProizvodId, NazivProizvoda, Cena
FROM Proizvod INNER JOIN Kategorija
ON Kategorija.KategorijaId = Proizvod.KategorijaId
WHERE NazivKategorije = @NazivKategorije
```

478

Kreiranje ulaznog parametara

```
SqlCommand komanda = new SqlCommand(sb.ToString(), konekcija);
SqlParameter kategorijaParameter = new
SqlParameter("@NazivKategorije", SqlDbType.NVarChar, 50);
komanda.Parameters.Add(kategorijaParameter);
kategorijaParameter.Value = textBox1.Text;
```

```
komanda.Parameters.AddWithValue("@NazivKategorije",
textBox1.Text);
```

479

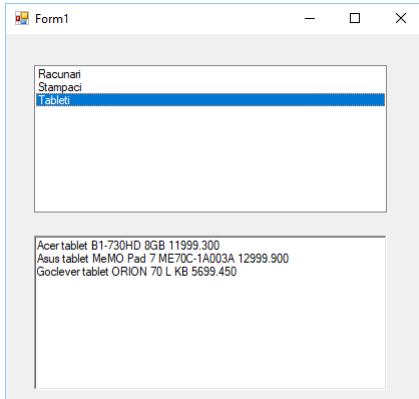
Kreiranje izlaznog parametra

```
SqlCommand cmd= new SqlCommand(tekstKomande, konekcija);

SqlParameter param = new SqlParameter("@KategorijaId", SqlDbType.Int);
param.Direction = ParameterDirection.Output;
cmd.Parameters.Add(param);
```

480

Korisnički interfejs



481

Definisanje parametarskog upita -1

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    Kategorija k = listBox1.SelectedItem as Kategorija;
    StringBuilder sb = new StringBuilder();
    sb.AppendLine("SELECT NazivProizvoda, Cena");
    sb.AppendLine("FROM Proizvod");
    sb.AppendLine("WHERE KategorijaId = @KategorijaId");
    string greska = "";
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand(sb.ToString(), konekcija))
        {
            ...
        }
    }

    if (greska != "")
    {

    }
    else
    {
        richTextBox1.Text = sb.ToString();
    }
}
```

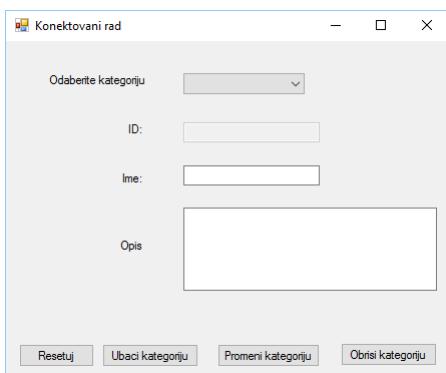
482

Definisanje parametarskog upita -2

```
SqlParameter idParametar = new SqlParameter("@KategorijaId", SqlDbType.Int);
idParametar.Value = k.KategorijaId;
komanda.Parameters.Add(idParametar);
sb.Clear();
try
{
    konekcija.Open();
    SqlDataReader dr = komanda.ExecuteReader();
    while (dr.Read())
    {
        sb.AppendLine(dr.GetString(0) + " " + dr.GetDecimal(1));
    }
}
catch (Exception xcp)
{
    greska = xcp.Message;
}
```

483

Korisnički interfejs



484

Resetovanje korisničkog interfejsa

```
private void Resetuj()
{
    textBoxId.Text = "";
    textBoxNaziv.Text = "";
    textBoxOpis.Text = "";
    comboBox1.SelectedIndex = -1;
}
```

485

Klasa Kategorija

```
public class Kategorija
{
    public int KategorijaId { get; set; }
    public string NazivKategorije { get; set; }
    public string OpisKategorije { get; set; }

    public override string ToString()
    {
        return NazivKategorije;
    }
}
```

486

Metoda VratiKategorije(), klasa KategorijaDal

```
public static List<Kategorija> VratiKategorije()
{
    List<Kategorija> lista = new List<Kategorija>();
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand("SELECT * FROM Kategorija ORDER BY NazivKategorije", konekcija))
        {

            try
            {
                konekcija.Open();
                using (SqlDataReader dr = komanda.ExecuteReader())
                {
                    while (dr.Read())
                    {
                        Kategorija k = new Kategorija();
                        k.KategorijaId = dr.GetInt32(0);
                        k.NazivKategorije = dr.GetString(1);
                        k.OpisKategorije = dr.GetString(2);
                        lista.Add(k);
                    }
                }
                return lista;
            }
            catch (Exception)
            {
                return null;
            }
        }
    }
}
```

487

Metoda PrikaziKategorije klasa Form1

```
private List<Kategorija> listaKategorija = new List<Kategorija>();
```

```
private void PrikaziKategorije()
{
    comboBox1.Items.Clear();
    listaKategorija = KategorijaDal.VratiKategorije();

    if (listaKategorija != null)
    {
        foreach (Kategorija k in listaKategorija)
        {
            comboBox1.Items.Add(k);
        }
    }
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    PrikaziKategorije();
}
```

488

Metoda Validacija() klasa Form1

```
private bool Validacija()
{
    if (string.IsNullOrWhiteSpace(textBoxNaziv.Text))
    {
        MessageBox.Show("Morate uneti naziv kategorije", "Poruka");
        textBoxNaziv.Focus();
        return false;
    }
    if (string.IsNullOrWhiteSpace(textBoxOpis.Text))
    {
        MessageBox.Show("Morate uneti opis kategorije", "Poruka");
        textBoxOpis.Focus();
        return false;
    }

    return true;
}
```

489

Metoda UbaciKategoriju, klasa KategorijaDal

```
public static int UbaciKategoriju(Kategorija k)
{
    StringBuiler sb = new StringBuiler();
    sb.AppendLine("INSERT INTO Kategorija VALUES (@NazivKategorije, @OpisKategorije)");
    sb.AppendLine("SELECT CAST(SCOPE_IDENTITY() AS int)");
    int ID = 0;
    using (SqlConnection konekcija = new SqlConnection(Konekcia.cnnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand(sb.ToString(), konekcija))
        {
            try
            {
                komanda.Parameters.AddWithValue("@NazivKategorije", k.NazivKategorije);
                komanda.Parameters.AddWithValue("@OpisKategorije", k.OpisKategorije);
                konekcija.Open();
                ID = (int)komanda.ExecuteScalar();
                return ID;
            }
            catch
            {
                return -1;
            }
        }
    }
}
```

490

Unos podataka u bazu, klasa Form1

```

private void button1_Click(object sender, EventArgs e)
{
    if (!Validacija())
    {
        return;
    }
    Kategorija k = new Kategorija();
    k.NazivKategorije = textBoxNaziv.Text;
    k.OpisKategorije = textBoxOpis.Text;

    int id = KategorijaDal.Ubacikategoriju(k);

    if (id > 0)
    {
        PrikaziKategorije();
        comboBox1.SelectedIndex = listaKategorija.FindIndex(k1 => k1.KategorijaId == id);
        MessageBox.Show($"Ubacena nova kategorija ciji je id {id}");
    }
    else
    {
        MessageBox.Show("Greska pri unosu kategorije");
        return;
    }
}

```

491

Metoda PromeniKategoriju, klasa KategorijaDal

```

public static int PromeniKategoriju(Kategorija k)
{
    StringBuilder sb = new StringBuilder();
    sb.AppendLine("UPDATE Kategorija");
    sb.AppendLine("SET NazivKategorije = @NazivKategorije,");
    sb.AppendLine("OpisKategorije = @OpisKategorije");
    sb.AppendLine("WHERE (KategorijaId = @KategorijaId)");
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand(sb.ToString(), konekcija))
        {
            try
            {
                komanda.Parameters.AddWithValue("@NazivKategorije", k.NazivKategorije);
                komanda.Parameters.AddWithValue("@OpisKategorije", k.OpisKategorije);
                komanda.Parameters.AddWithValue("@KategorijaId", k.KategorijaId);
                konekcija.Open();
                komanda.ExecuteNonQuery();
                return 0;
            }
            catch (Exception )
            {
                return -1;
            }
        }
    }
}

```

492

Promena podataka u bazi, klasa Form1

```
private void button2_Click(object sender, EventArgs e)
{
    int indeks = comboBox1.SelectedIndex;
    if (indeks < 0)
    {
        MessageBox.Show("Odaberite kategoriju");
        return;
    }
    Kategorija k = comboBox1.SelectedItem as Kategorija;
    if (Validacija())
    {
        k.NazivKategorije = textBoxNaziv.Text;
        k.OpisKategorije = textBoxOpis.Text;

        int rez = KategorijaDal.PromeniKategoriju(k);

        if (rez == 0)
        {
            PrikaziKategorije();
            comboBox1.SelectedIndex = listaKategorija.FindIndex(k1 => k1.KategorijaId == k.KategorijaId);
            MessageBox.Show("Promenjena kategorija");
        }
        else
        {
            MessageBox.Show("Greska pri promeni kategorije");
        }
    }
}
```

493

Metoda Obrisikategoriju, klasa KategorijaDal

```
public static int Obrisikategoriju(int id)
{
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand("DELETE FROM Kategorija WHERE KategorijaId=@KategorijaId", konekcija))
        {
            try
            {
                komanda.Parameters.AddWithValue("@KategorijaId", id);
                konekcija.Open();
                komanda.ExecuteNonQuery();
                return 0;
            }
            catch
            {
                return -1;
            }
        }
    }
}
```

494

Brisanje podataka u bazi, klasa Form1

```
private void button3_Click(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex < 0)
    {
        MessageBox.Show("Morate odabrat kategoriju");
        return;
    }

    Kategorija k = comboBox1.SelectedItem as Kategorija;

    int rez = KategorijaDal.Obrisikategoriju(k.KategorijaId);

    if (rez == 0)
    {
        Prikazikategorije();
        Resetuj();
        MessageBox.Show("Obrisana kategorija");
    }
    else
    {
        MessageBox.Show("Greska pri brisanju kategorije");
    }
}
```

495

Pitanje 1

SqlDataReader objekat se kreira:

- pozivom SqlDataReader konstruktora
- pozivom CreateReader() metode objekta SqlConnection
- pozivom metode ExecuteReader() objekta SqlConnection
- pozivom metode ExecuteReader() objekta SqlCommand

Odgovor: d

496

Pitanje 2

Za izvršavanje upita nad bazom podataka koji treba da vrati jedan realan broj koristićemo metodu SqlCommand objekta :

- a. ExecuteDataSet()
- b. ExecuteReader()
- c. ExecuteScalar()
- d. ExecuteNonQuery()

Odgovor: c

497

Pitanje 3

Ukoliko se drugacije ne naznači podrazumevana vrednost svojstva CommandType objekta SqlCommand je:

- a. Text
- b. CommandType.Text
- c. CommandType.StoredProcedure
- d. StoredProcedute

Odgovor: b

498

Pitanje 4

Izlaznom parametru SqlCommand objekta

- a. ne dodeljuje se vrednost pre izvršavanja odgovarajuće komande
- b. dodeljuje se vrednost pre izvršavanja odgovarajuće komande
- c. dodeljuje se vrednost pre izvršavanja odgovarajuće komande samo ako postoje i ulazni parametri

Odgovor: a

499

Pitanje 5

Ulaznom parametru SqlCommand objekta:

- a. ne dodeljuje se vrednost pre izvršavanja odgovarajuće komande
- b. mora se dodeliti vrednost pre izvršavanja odgovarajuće komande
- c. vrednost se može ali ne mora dodeliti

Odgovor: b

500

Pitanje 6

Dat je C# kod:

```
SqlParameter CityParameter = new SqlParameter("@City", SqlDbType.NVarChar, 15);
cmdKorisknikIzGrada.Parameters.Add(CityParameter);
CityParameter.Value = textBox1.Text;
```

Ovim kodom kreira se:

- a. ulazni parametar
- b. izlazni
- c. parametar koji može biti i ulazni i izlazni

Odgovor: a

501

Rad sa uskladištenim procedurama

Baza KorisnikDb

```

CREATE DATABASE KorisnikDB
GO

USE KorisnikDB
GO

CREATE TABLE Korisnik
(
KorisnikId int IDENTITY(1,1) NOT NULL PRIMARY KEY,
Ime nvarchar(30) NOT NULL,
Prezime nvarchar(30) NOT NULL,
Email varchar(50) NOT NULL
CONSTRAINT UQ_Email UNIQUE
CONSTRAINT CHK_Email CHECK (Email LIKE '%@%.%'),
Lozinka varchar(15) NOT NULL
)

```

503

Procedura UbaciKorisnika

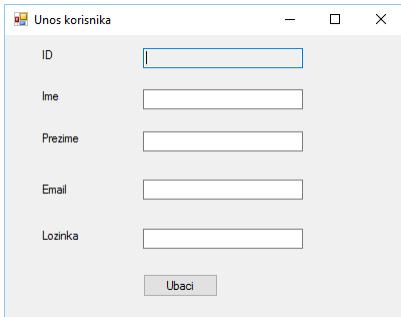
```

GO
CREATE PROC UbaciKorisnika
(
@Ime nvarchar(30),
@Prezime nvarchar(50),
@email nvarchar(50),
@Lozinka nvarchar(50),
@KorisnikId int OUTPUT
)
AS
INSERT INTO Korisnik ( Ime, Prezime, Email, Lozinka )
VALUES ( @Ime, @Prezime, @Email, @Lozinka );
SELECT @KorisnikID = @@IDENTITY;
GO

```

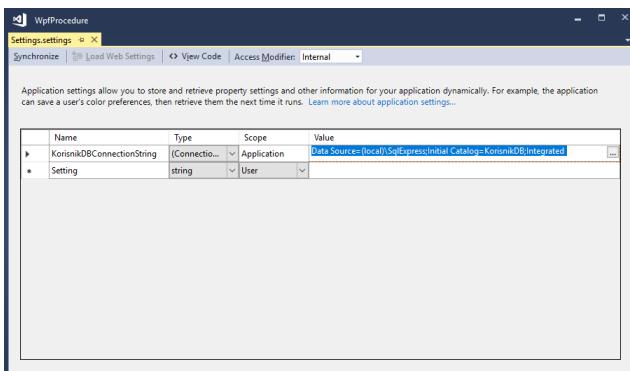
504

Korisnički interfejs



505

Upis konekcionog stringa u aplikaciona setovanja



506

Klasa Konekcija

```
static class Konekcija
{
    public static string cnnKorisnikDB =
Properties.Settings.Default.KorisnikDBConnectionString;
}
```

507

Izvršavanje uskladištene procedure -1

```
private void ButtonUbaci_Click(object sender, EventArgs e)
{
    int ID = 0;
    string poruka = "";
    using (SqlConnection konekcija = new SqlConnection(Konekcija.cnnKorisnikDB))
    {
        using (SqlCommand komanda = new SqlCommand("UbaciKorisnika", konekcija))
        {
            ...
        }
    }
}
```

508

Izvršavanje uskladištene procedure -2

```

komanda.CommandType = CommandType.StoredProcedure;
SqlParameter IdParametar = new SqlParameter("@KorisnikId ", SqlDbType.Int);
IdParametar.Direction = ParameterDirection.Output;
komanda.Parameters.Add(IdParametar);

try
{
    komanda.Parameters.AddWithValue("@Ime", textBoxIme.Text);
    komanda.Parameters.AddWithValue("@Prezime", textBoxPrezime.Text);
    komanda.Parameters.AddWithValue("@Email", textBoxEmail.Text);
    konekcija.Open();
    komanda.ExecuteNonQuery();
    ID = (int)IdParametar.Value;
}
catch (Exception xcp)
{
    poruka = xcp.Message;
}

```

509

Izvršavanje uskladištene procedure -3

```

if (poruka != "")
{
    MessageBox.Show(poruka);
}
else
{
    MessageBox.Show("Ubacen Korisnik ciji je ID : " + ID.ToString());
}

```

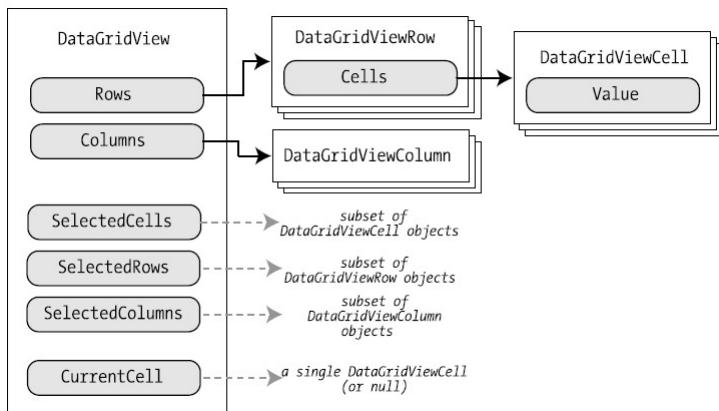
510

Kontrole DataGridView i ListView

Uvod u DataGridView

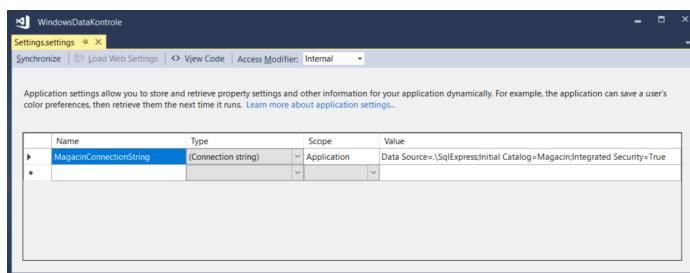
- DataGridView je moćna, fleksibilna kontrola, jednostavna za korišćenje, koja se koristi za prikaz tabelarnih podataka
- Može da se poveže sa kolekcijom objekata korišćenjem svojstva DataSource
- DataGridView kreira jednu kolonu za svako svojstvo iz izvora podataka
- Svojstvo Columns daje kolekciju DataGridViewColumn objekata
- Svojstvo Rows daje kolekciju DataGridViewRow objekata
- DataGridViewRow objekat ima svojstvo Cells kojim se prikazuje kolekcija DataGridViewCell objekata

Šematski prikaz DataGridView objekata



513

Setovanja aplikacije



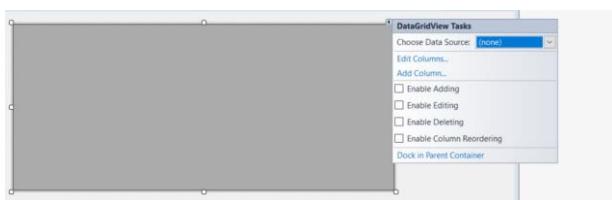
514

Klasa Konekcija

```
static class Konekcija
{
    public static string cnnMagacin =
Properties.Settings.Default.MagacinConnectionString;
}
```

515

Podešavanje DataGridView kontrole



516

Podešavanje DataGridView kontrole

Properties

dataGridView1 System.Windows.Forms.DataGridView

Dock None
EditMode EditOnKeystrokeOrF2
Enabled True
EnableHeadersVisualStyles True
GenerateMember True
GridColor ControlDark
ImeMode NoControl
Location 40; 23
Locked False
Margin 3; 3; 3; 3
MaximumSize 0; 0
MinimumSize 0; 0
Modifiers Private
MultiSelect **False**
ReadOnly True
RightToLeft No
RowHeadersBorderStyle Raised
Edit Columns... Add Column...

Properties

dataGridView1 System.Windows.Forms.DataGridView

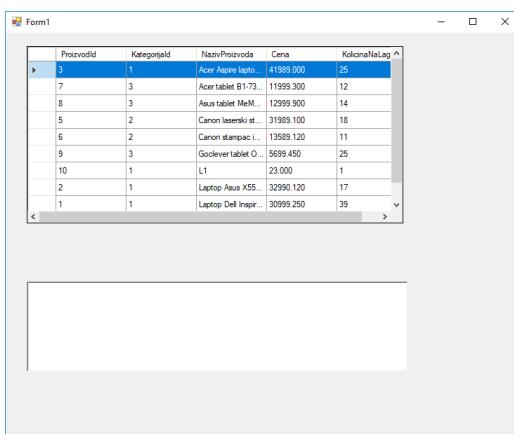
MaximumSize 0; 0
MinimumSize 0; 0
Modifiers Private
MultiSelect **False**
ReadOnly True
RightToLeft No
RowHeadersBorderStyle DataGridViewCellStyle { BackColor=Color
RowHeadersVisible True
RowHeadersWidth 41
RowHeadersWidthSizeMode EnableResizing
RowsDefaultCellStyle DataGridViewCellStyle { }
RowTemplate DataGridViewRow { Index=-1 }
ScrollBars Both
SelectionMode FullRowSelect
ShowCellErrors True
ShowCellToolTips True
Edit Columns... Add Column...

MultiSelect
Indicates whether the user is allowed to select more than one cell, row, or column of th...

SelectionMode
Indicates how the cells of the DataGridView can be selected.

517

Korisnički interfejs



518

Klasa Proizvod

```
class Proizvod
{
    public int ProizvodId { get; set; }
    public int KategorijaId { get; set; }
    public string NazivProizvoda { get; set; }
    public decimal Cena { get; set; }
    public int KolicinaNaLageru { get; set; }

    public override string ToString()
    {
        return NazivProizvoda;
    }
}
```

519

Metoda VratiProizvode() statička klasa ProizvodDal

```
public static List<Proizvod> VratiProizvode()
{
    List<Proizvod> listaProizvoda = new List<Proizvod>();
    using (SqlConnection konekcija = new SqlConnection(konekcija.cnMagacin))
    {
        using (SqlCommand komanda = new SqlCommand("SELECT * FROM Proizvod ORDER BY NazivProizvoda", konekcija))
        {
            try
            {
                konekcija.Open();
                using (SqlDataReader dr = komanda.ExecuteReader())
                {
                    while (dr.Read())
                    {
                        Proizvod p1 = new Proizvod
                        {
                            ProizvodId = dr.GetInt32(0),
                            KategorijaId = dr.GetInt32(1),
                            NazivProizvoda = dr.GetString(2),
                            Cena = dr.GetDecimal(3),
                            KolicinaNaLageru = dr.GetInt32(4)
                        };
                        listaProizvoda.Add(p1);
                    }
                }
                return listaProizvoda;
            }
            catch (Exception)
            {
                return null;
            }
        }
    }
}
```

520

Povezivanje sa kolekcijom objekata

```
private void PrikaziProizvode()
{
    dataGridView1.DataSource = null;
    dataGridView1.DataSource = ProizvodDal.VratiProizvode();
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    PrikaziProizvode();
}
```

521

Iščitavanje selektovanog reda DataGridView kontrole

```
private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        DataGridViewRow selektovaniProizvod = dataGridView1.SelectedRows[0];

        StringBuilder sb = new StringBuilder();
        sb.AppendLine(selektovaniProizvod.Cells[2].Value.ToString());
        sb.AppendLine(selektovaniProizvod.Cells[3].Value.ToString());
        sb.AppendLine(selektovaniProizvod.Cells[4].Value.ToString());

        richTextBox1.Text = sb.ToString();
    }
}
```

522

Klasa ProizvodView

```
class ProizvodView
{
    public int Id { get; set; }
    public string Naziv { get; set; }
    public decimal Cena { get; set; }
}
```

523

Modifikacija metode PrikaziProizvode()

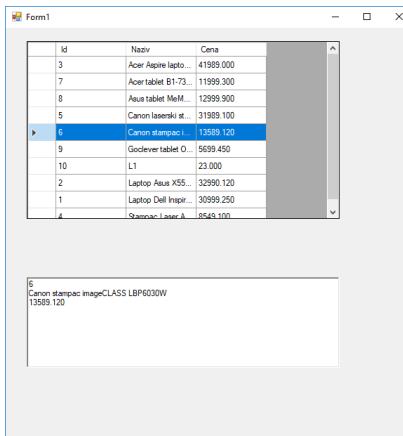
```
private void PrikaziProizvode()
{
    dataGridView1.DataSource = null;
    // dataGridView1.DataSource = ProizvodDal.VratiProizvode();

    List<Proizvod> listaProizvoda = ProizvodDal.VratiProizvode();

    IEnumerable<ProizvodView> lista1 = listaProizvoda.Select(p => new ProizvodView
    {
        Id = p.ProizvodId,
        Naziv = p.NazivProizvoda,
        Cena = p.Cena
    });
    dataGridView1.DataSource = lista1.ToList();
}
```

524

Prikaz određenih kolona tabele



525

Indeks selektovanog reda

```
private void button1_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int indeks = dataGridView1.SelectedRows[0].Index;
        MessageBox.Show(indeks.ToString());
    }
    else
    {
        MessageBox.Show("Nije selektovan red");
    }
}
```

526

ListView kontrola

- ListView omogućava da se prikaže lista stavki kojima se mogu pridružiti slike kao u File eksploreru
- Omogućava da se stavke prikazuju na jedan od četiri različita načina:
 - prikaz teksta sa velikim ikonama
 - prikaz teksta sa malim ikonama
 - prikaz vertikalne liste
 - prikaz detalja
- Stavke ListView kontrole su objekti klase ListViewItem

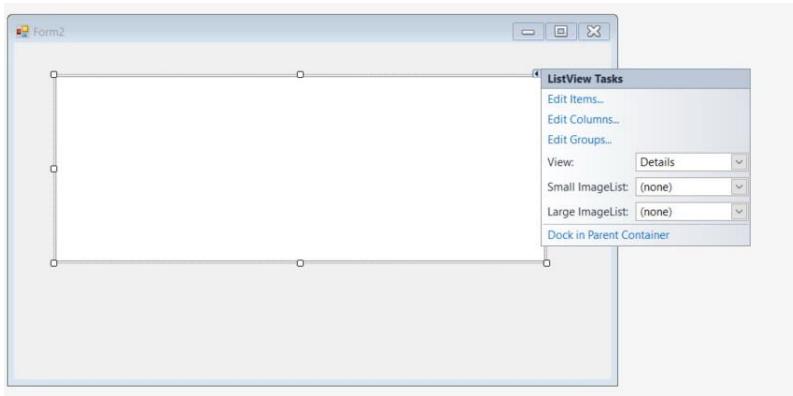
527

Svojstva ListView kontrole

- Svojstvo **Columns** sadrži kolekciju kolona koje će biti prikazani u režimu detalja
- Svojstvo **Items** daje kolekciju **ListViewItem** objekata koji će biti prikazani u ListView kontroli
- Svojstvo **LargeImageList** ukazuje na ImageList komponentu iz koje će biti prikazane slike u ListView kontroli kada je njeno View svojstvo postavljeno na vrednost LargeIcon
- Svojstvo **SmallImageList** ukazuje na ImageList komponentu iz koje se prikazuju slike kada je View svojstvo ListView kontrole postavljeno na vrednost SmallIcon
- Svojstvo **View** pokazuje način na koji se kontrola ListView prikazuje

528

ListView kontrola u režimu detalja



529

Podešavanje ListView kontrole

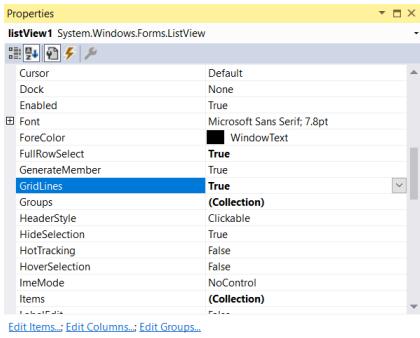
Property	ListView1	ListView2
Enabled	True	True
Font	Microsoft Sans Serif; 7.8pt	Microsoft Sans Serif; 7.8pt
ForeColor	WindowText	WindowText
GenerateMember	True	True
GridLines	False	False
Groups	(Collection)	(Collection)
HeaderStyle	Clickable	Clickable
HideSelection	True	True
HotTracking	False	False
HoverSelection	False	False
ImeMode	NoControl	NoControl
Items	(Collection)	(Collection)
LabelEdit	False	False
LabelWrap	True	True
LargeImageList	(none)	(none)
Location	56, 45	56, 45
Locked	False	False
Margin	3, 3, 3, 3	3, 3, 3, 3
MaximumSize	0, 0	0, 0
MinimumSize	0, 0	0, 0
Modifiers	Private	Private
MultiSelect	False	False
OwnerDraw	False	False
RightToLeft	No	No

MultiSelect
Allows multiple items to be selected.

FullRowSelect
Indicates whether all SubItems are highlighted along with the item when selected.

530

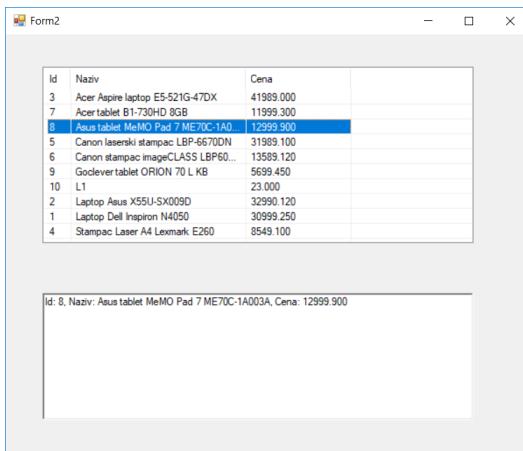
Prikaz linija grida kod ListView kontrole



GridLines
Displays grid lines around items and SubItems. Only shown when in Details view.

531

Korisnički interfejs



532

Definisanje kolona ListView kontrole

```
private void Form2_Load(object sender, EventArgs e)
{
    listView1.View = View.Details;

    listView1.Columns.Add("Id", 30);
    listView1.Columns.Add("Naziv", 200);
    listView1.Columns.Add("Cena", 120);

    // PrikaziProizvode();
}
```

533

Prikaz podataka iz baze u ListView kontroli

```
private void PrikaziProizvode()
{
    listView1.Items.Clear();

    foreach (Proizvod p in ProizvodDal.VratiProizvode())
    {
        ListViewItem lvi = new ListViewItem(p.ProizvodId.ToString());
        lvi.SubItems.Add(p.NazivProizvoda);
        lvi.SubItems.Add(p.Cena.ToString());
        listView1.Items.Add(lvi);
    }
}
```

534

Load procedura forme

```
private void Form2_Load(object sender, EventArgs e)
{
    listView1.View = View.Details;

    listView1.Columns.Add("Id", 30);
    listView1.Columns.Add("Naziv", 200);
    listView1.Columns.Add("Cena", 120);

    PrikaziProizvode();
}
```

535

Iščitavanje selektovane stavke

```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listView1.SelectedItems.Count > 0)
    {
        ListViewItem sel = listView1.SelectedItems[0];

        string id = sel.SubItems[0].Text;
        string naziv = sel.SubItems[1].Text;
        string cena = sel.SubItems[2].Text;

        richTextBox1.Text = $"Id: {id}, Naziv: {naziv}, Cena: {cena}";
    }
}
```

536

Pitanje 1

Svojstvo Columns objekta klase DataGridView daje:

- a. Kolekciju objekata klase DataColumn
- b. Kolekciju objekata klase DataGridViewColumn
- c. Kolekciju objekata klase DataGridViewColumn

Odgovor: c

537

Pitanje 2

Tip selekcije u DataGridView kontroli definiše se njenim svojstvom:

- a. SelectionMode
- b. SelectionType
- c. SelectionUnit

Odgovor: a

538

Pitanje 3

DataGridView kontrola čije je svojstvo Name dataDataGridView1, ima svojstvo

MultiSelect postavljeno na vrednost false

i svojstvo SelectionMode na vrednost FullRowSelect.

Na koji način se pristupa selektovanom redu:

- a. DataGridViewRow selVrsta = dataDataGridView1.SelectedRows;
- b. DataGridViewRow selVrsta = dataDataGridView1.SelectedRows[0];
- c. DataGridViewRow selVrsta = dataDataGridView1.SelectedItem;

Odgovor: b

539

Pitanje 4

U kontroli ListView

- a. stavke se uvek prikazuju na isti način
- b. način prikazivanja stavki definise se svojstvom View
- c. način prikazivanja stavki definise se svojstvom Icon
- d. način prikazivanja stavki definise se svojstvom ImageMode

Odgovor: b

540

Pitanje 5

Stavke ListView kontrole su objekti klase:

- a. ListViewItem
- b. ListItem
- c. ListNode

Odgovor: a

541

Pitanje 6

Da bismo zabranili selektovanje više od jedne stavke unutar ListView kontrole :

- a. svojstvo Multiple postavimo na False
- b. svojstvo MultiSelect postavimo na False
- c. svojstvo SingleSelect postavimo na vrednost True

Odgovor: b

542