Operatori za pristup članu

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";
    }
}
```

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";
}
```

Struktura DateTime

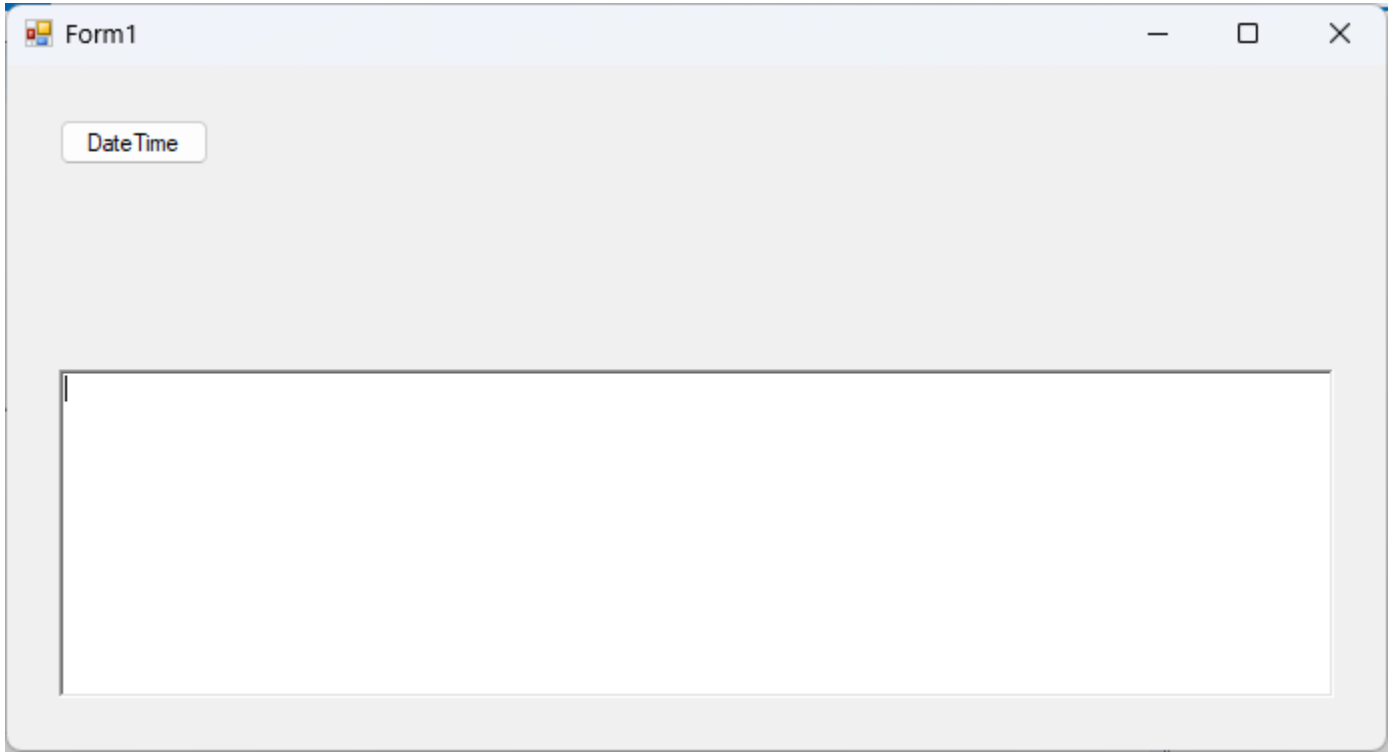
Konstruktori DateTime strukture

```
[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
);
```

Korisnički interfejs aplikacije




```
private void Stampaj(string s)
{
    richTextBox1.AppendText( s + "\n");
}
```

Format datuma

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

15.7.2005. 11:20:00
15.07.2005

```
private void button2_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Now;
    Stampaj(dt.ToLongDateString());
    Stampaj(dt.ToShortDateString());
    Stampaj(dt.ToString("D"));
    Stampaj(dt.ToString("d"));
}
```

sreda, 03. decembar 2025.
3.12.2025.
sreda, 03. decembar 2025.
3.12.2025

```
private void button1_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Today;
    Stampaj(dt.ToString());
    Stampaj(dt.ToString("d"));
}
```

3.12.2025. 00:00:00
3.12.2025.

Format vremena

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

05:29:29

```
private void button5_Click(object sender, RoutedEventArgs e)
{
    DateTime dt = DateTime.Now;
    Stampaj(dt.ToLongTimeString());
    Stampaj(dt.ToShortTimeString());
    Stampaj(dt.ToString("T"));
    Stampaj(dt.ToString("t"));
}
```

17:30:06
17:30
17:30:06
17:30

```
private void button6_Click(object sender, EventArgs e)
{
    DateTime dt1 = DateTime.Now;
    string s = dt1.Hour.ToString() + " : " + dt1.Minute.ToString();
    Stampaj(s);
}
```

17 : 30

Sortiranje datumskih vrednosti

```
private void Button1_Click(object sender, RoutedEventArgs e)
{
    DateTime[] nizDatuma = {

    new DateTime(2018,4,21),
    new DateTime(2018,5,11),
    new DateTime(2018,10,21),
    new DateTime(2017,2,19),
    };
    Array.Sort(nizDatuma);

    foreach (DateTime dt in nizDatuma)
    {
        Stampaj(dt.ToString("d"));
    }
}
```

19.02.2017.
21.04.2018.
11.05.2018.
21.10.2018.

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 )
```

Upotreba TimeSpan strukture

```
private void button7_Click(object sender, EventArgs e)
{

    DateTime polazak = new DateTime(2025, 11, 25, 18, 30, 0);
    DateTime dolazak = new DateTime(2026, 2, 25, 19, 47, 0);
    TimeSpan putovanje = dolazak - polazak;

    Stampaj(putovanje.ToString());
    Stampaj(putovanje.Days.ToString());
    Stampaj(putovanje.Hours.ToString());
    Stampaj(putovanje.Minutes.ToString());

}
```

```
92.01:17:00
92
1
17
```

TimeSpan struktura-1

```
private void button8_Click(object sender, EventArgs e)
{
    DateTime dt1 = DateTime.Today;
    Stampaj(dt1.ToString());
    TimeSpan ts = new TimeSpan(23, 0, 0, 0);

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

    Stampaj(dt2.ToString("d"));
    Stampaj(dt3.ToString("d"));
}
```

```
3.12.2025. 00:00:00
26.12.2025.
10.11.2025.
```

Obrada izuzetaka

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-a ili eksplicitno od strane programera (throw)
- Izuzeci se obrađuju korišćenjem ključnih reči ***try***, ***catch*** i ***finally***

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 oslobode 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

try-catch blok

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

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

Korišćenje više catch blokova

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

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

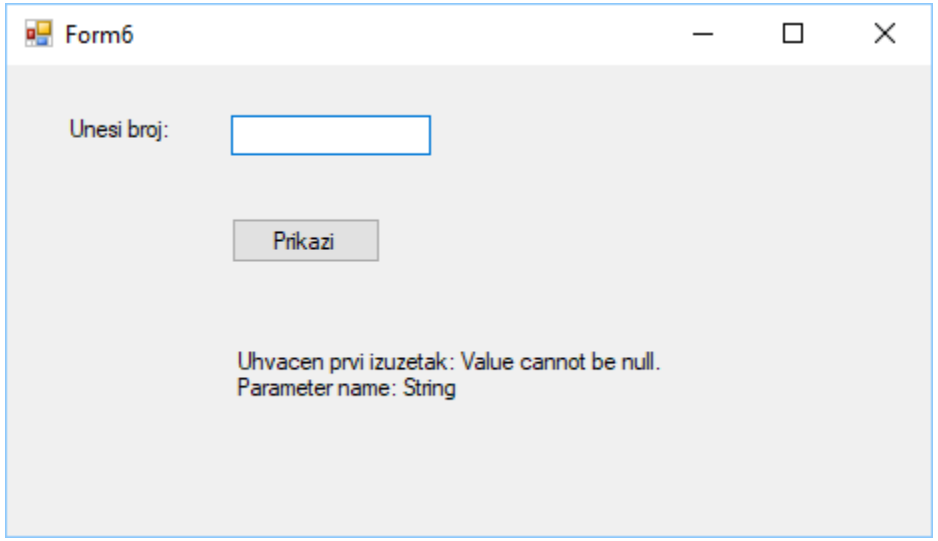
Upotreba dva catch bloka

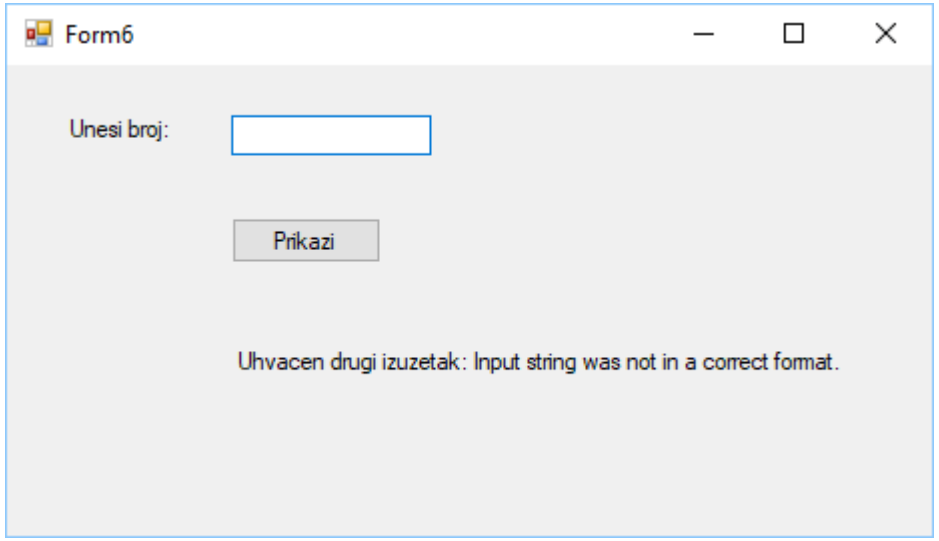
```
private void button1_Click(object sender, EventArgs e)
{
    string s = null;
    try
    {
        if (!string.IsNullOrWhiteSpace(textBox1.Text))
        {
            // string nije prazan
            s = textBox1.Text.Trim();
        }
        int a = int.Parse(s);
        labelPoruka.Text = a.ToString();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

    }
    catch (Exception ex)
    {
        labelPoruka.Text = "Uhvacen drugi izuzetak: " + ex.Message;
    }
    textBox1.Clear();
    textBox1.Focus();
}
```

Analiza koda

- Ako je TextBox prazan, s ostaje null, int.Parse(null) baca **ArgumentNullException,** hvata ga prvi catch blok i ispisuje se poruka „Uhvacen prvi izuzetak”
- Kada se unese tekst koji nije broj (npr. „abc“) izbacuje se **FormatException,** hvata ga drugi, opšti catch(Exception) blok i prikazuje se poruka „Uhvacen drugi izuzetak…“
- Ako je sve u redu na kontroli Label se ispisuje uneta vrednost

Upotreba dva catch bloka

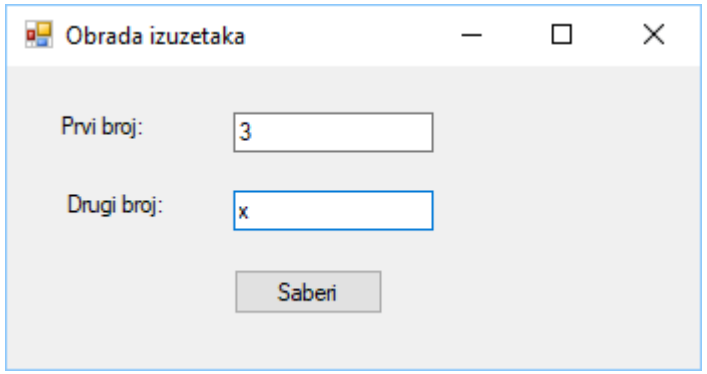


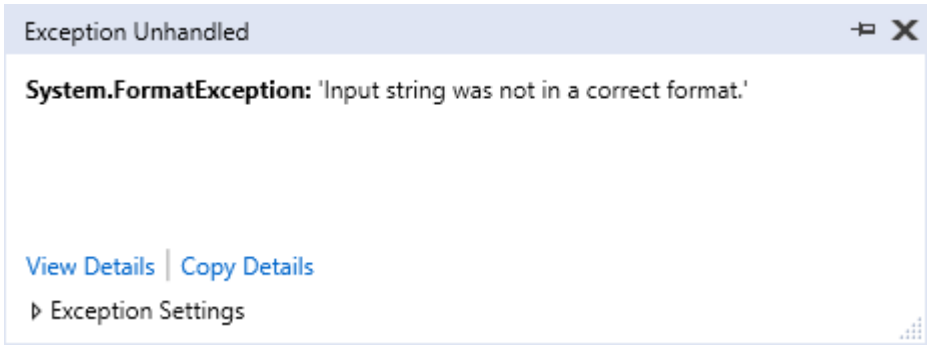


Primer bez upotrebe izuzetaka

```
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");
}
```

Aplikacija prekida sa radom zbog izuzetka





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();
}
```

Kolekcije

Pojam kolekcije

- Kolekcije se upotrebljavaju za upravljanje 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

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**

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 pozicuju ***pozicija***
- Metoda ***RemoveAt(index)*** briše element sa indeksom index iz liste

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

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 emente kolekcijeu inverznom redosledu
- Metoda **ToArray**() kopira elemente liste u jednodimenzionalan niz

Generičke liste

Generičke liste (List<T>)

- Generičke liste se nalaze u prostoru imena System.Collections.Generic
- Omogućavaju skladištenje elemenata istog tipa
- Ne koriste tip object, pa nije potrebno kastovanje
- Type-safe su i greške tipa se detektuju u vreme kompajliranja
- Automatski povećavaju kapacitet po potrebi
- Podržavaju indeksni pristup i jednostavnu iteraciju

```
List<int> brojevi = new List<int>();
```

Metode i svojstva klase List<T>

- **Add**(vrednost) – dodaje element na kraj liste
- **Insert**(indeks, vrednost) – ubacuje element na zadati indeks
- **Remove**(vrednost) – briše prvi element sa datom vrednošću
- **RemoveAt**(indeks) – briše element sa datog indeksa
- **Clear**() – uklanja sve elemente iz liste
- **Contains**(vrednost) – proverava da li element postoji
- **IndexOf**(vrednost) – vraća indeks prvog pojavljivanja
- **Sort**() – sortira elemente liste
- **Reverse**() – obrće redosled elemenata
- **ToArray**() – vraća elemente kao niz
- **Count** – svojstvo koje daje broj elemenata u listi

Generičke liste

List<T>

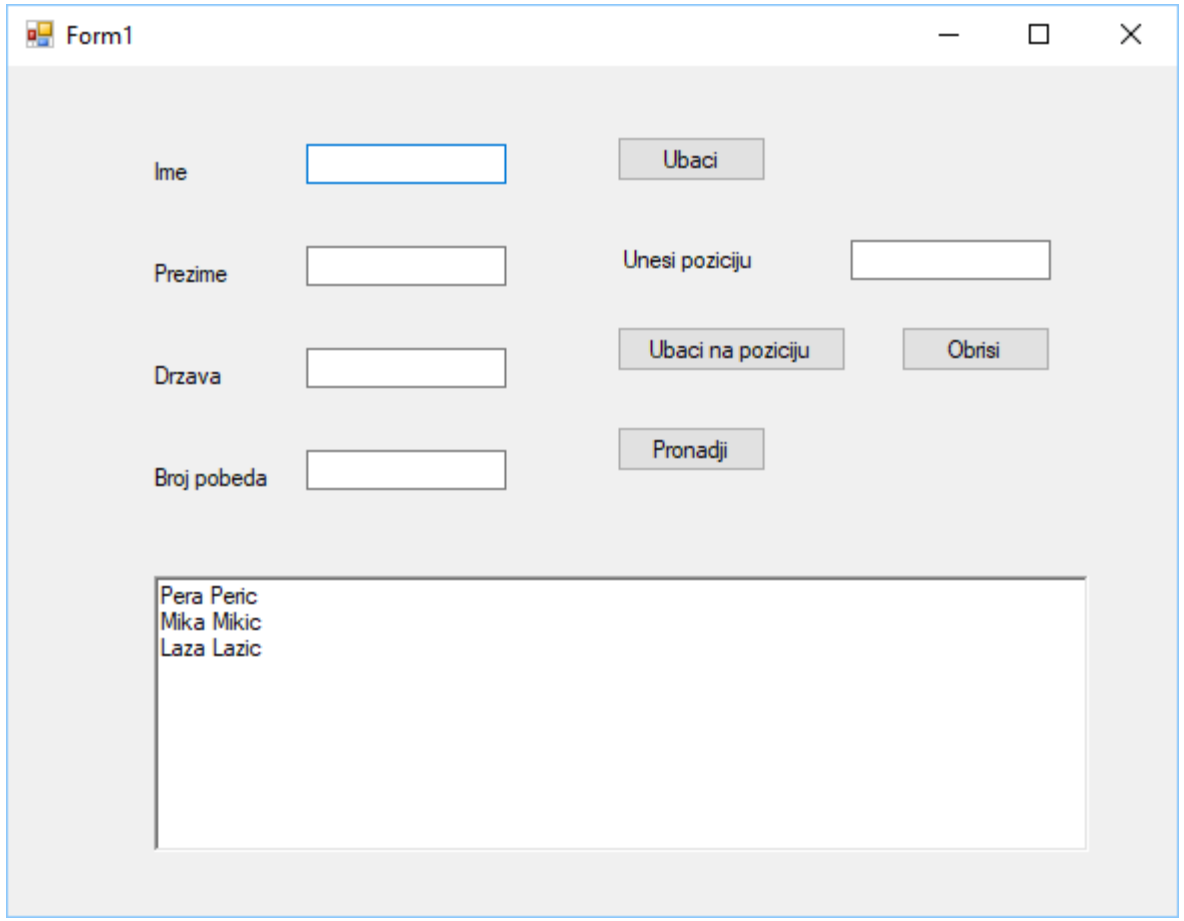
```
using System.Collections.Generic;
```

```
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");
    }

}
```

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 override string ToString()
        {
            return Ime + " " + Prezime;
        }

    }
```

GUI aplikacije



Štampanje liste

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

```
private void StampajListu()
{
    richTextBox1.Clear();
    foreach (Trkac t in lista)
    {
        richTextBox1.AppendText(t.ToString() + "\n");
    }
}
```

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();
}
```

Resetovanje korisničkog interfejsa

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

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;
    }

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

String metoda

```
public string VelikoPrvoSlovo(string rec)
{
    rec = rec.Trim().ToLower();

    if (rec.Length == 0)
        return string.Empty;

    if (rec.Length == 1)
        return rec.ToUpper();

    return rec.Substring(0, 1).ToUpper() + rec.Substring(1);
}
```

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;
    }
}
```

Ubacivanje elementa u listu

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

Definisanje jednakosti dva objekta

```
class Trkac : IEquatable<Trkac>
{
    ....
 public bool Equals(Trkac other)
 {
    if (other is null)
     {
      return false;
     }

    if (Ime.ToLower() == other.Ime.ToLower()
        &&
        Prezime.ToLower() == other.Prezime.ToLower()
        )
    {
        return true;
    }
    return false;
 }
}
```

Ubacivanje u listu samo različitih objekata

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

Ubacivanje elementa na poziciju

```
private void button2_Click(object sender, EventArgs e)
{
    Trkac t = KreirajTrkaca();
    if (t != null)
    {
        int pozicija = int.Parse(textBoxPozicija.Text);

        if (pozicija >= 0 && pozicija <= lista.Count)
        {
            lista.Insert(pozicija, t);
            StampajListu();
        }
        else
        {
            MessageBox.Show("Prekoracili ste poziciju", "Poruka");
        }

        Resetuj();
        textBoxPozicija.Clear();
    }
}
```

Uklanjanje elementa iz liste

```
private void button3_Click(object sender, EventArgs e)
{
    int pozicija = int.Parse(textBoxPozicija.Text);

    if (pozicija >= 0 && pozicija < lista.Count)
    {
        lista.RemoveAt(pozicija);
        StampajListu();
    }
    else
    {
        MessageBox.Show("Ne postoji član liste", "Poruka");
    }
}
```

Pristup elementima generičke liste

```
private void button4_Click(object sender, EventArgs e)
{
    int pozicija = int.Parse(textBoxPozicija.Text);

    if (pozicija >= 0 && 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("Prekoračili ste poslednju poziciju", "Poruka");
    }
}
```

Pitanje 1

Ukoliko želimo da iščitamo datum kada se desio klik na dugme bez iščitavanja informacija o vremenu koristimo sledeće svojstvo strukture DateTime:

a. Now
b. Today
c. Date

Odgovor: b

Pitanje 2

Kod koji može da dovede do greške pri izvršavanju stavlja se unutar bloka:

a. try
b. catch
c. finally

Odgovor: a

Pitanje 3

Generisani izuzetak od strane CLR komponente prosleđuje se bloku:

a. try
b. catch
c. finally

Odgovor: b

Pitanje 4

Svaki izuzetak može se dodeliti referenci tipa Exception:

a. Da
b. Ne

Odgovor: a

Pitanje 5

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

a. int<List> intLista = new int<List>();
b. List<int> intLista = new List<int>();
c. List[int] intLista = new List[int];

Odgovor: b

Pitanje 6

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

a. Da
b. Ne

Odgovor: b

Pitanje 7

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