

Napredno i objektno-
orijentirano programiranje
C#

Koncept OOP

- Klasa
- Objekt
- Metoda
- Svojstva

Četiri temeljna stupa

- Nasljeđivanje
- Učahurivanje
- Polimorfizam
- Kontrola pristupa

Klasa

- Klasu definiramo navođenjem ključne riječi **class**, proizvoljnog imena i bloka naredbi.

```
class Klase
{
    // definicije metoda, svojstva,
    događaja
}
```

- Klasa se nalazi unutar ***imenskog prostora*** .

Imenski prostor

- U C# imenski prostor definiramo pomoću ključne riječi namespace nakon koje dajemo proizvoljan naziv imenskom prostoru i zatim slijedi blok vitičastih zagrada.

```
using System;

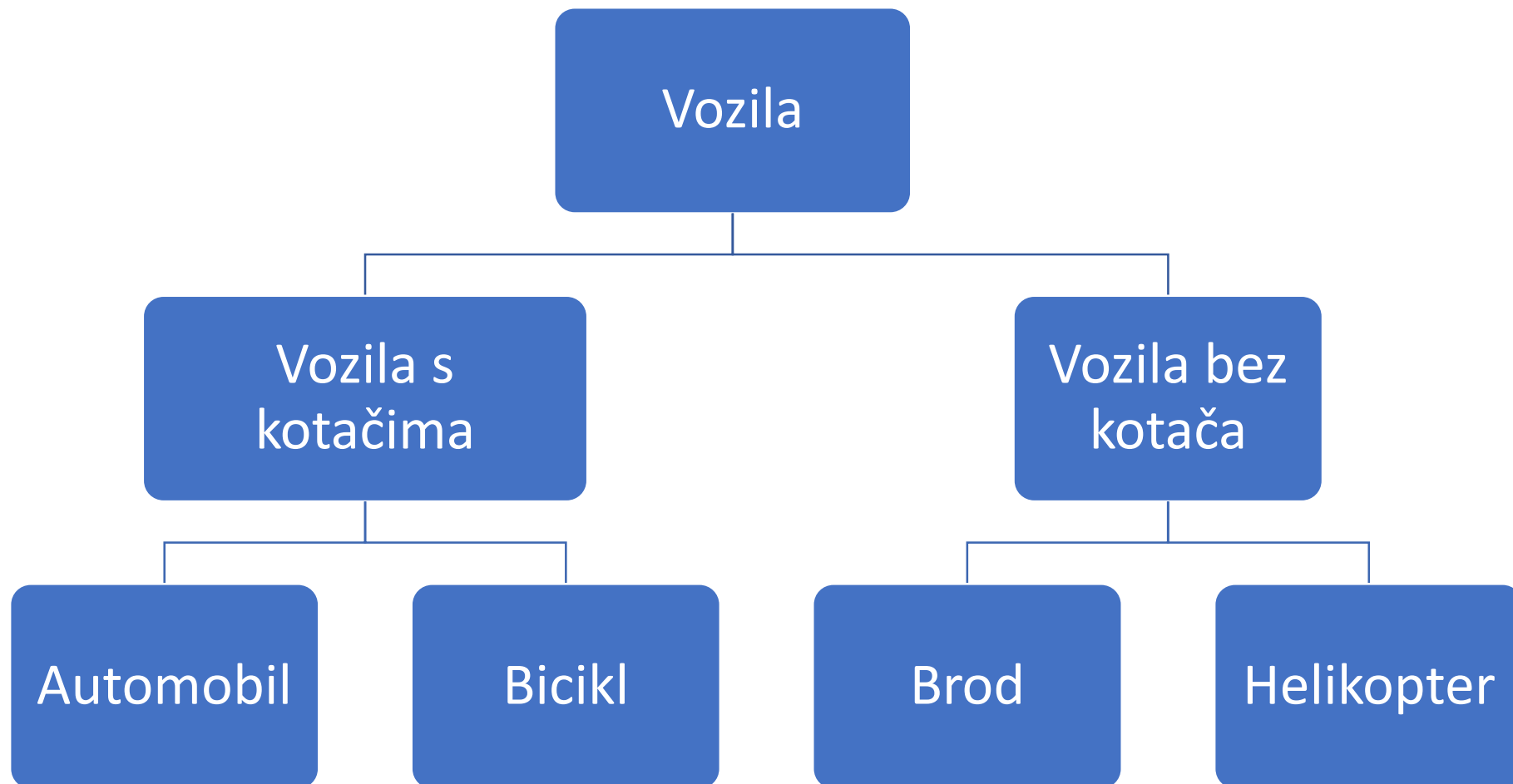
namespace Imenskiprostor
{
    class Klase
    {
        // definicije metoda, svojstva, događaja
    }
}
```

- Pruža mogućnost organizacije klasa na logički smislen način te time olakšava njihovo razlikovanje.

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

namespace ImenskiProstor
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine();
            Razred.UcimoKlase();
        }
    }
}

namespace MojaKlasa
{
    class Razred
    {
        public static void UcimoKlase()
        {
            Console.WriteLine("Pozdrav razrede!");
        }
    }
}
```



Objekt

BMW Škoda Octavia
Boeing787 GT bicikl

Metoda

Vozi
Stani
Poleti

Svojstvo

Boja, max.brzina, vrsta goriva,

Kontrola pristupa

Modifikatori pristupa su:

✓ public,

✓ private

✓ protected,

✓ internal

- Jedan od osnovnih koncepta OOP jest kontroliranje pristupa članovima sučelja objekta. Za svaki član moramo definirati razinu pristupa.
- Privatni članovi klase dostupni su samo metodama te klase i to je najviša razina zaštite.
- Javnim članovima klase može pristupiti bilo koji vanjski objekt.

Nasljeđivanje

Nasljeđivanjem- nasljeđena klasa (UcenikPrvi) nasljeđuje strukturu bazne klase (Razred).

```
using Nasljeđivanje;  
using System;  
using System.Reflection.PortableExecutable;
```

```
namespace Nasljeđivanje  
{  
    class Program  
    {  
        static void Main (string[] args)  
        {  
            UcenikPrvi Marinela = new UcenikPrvi();  
            Marinela.Metoda();  
        }  
    }  
    class Razred  
    {  
        public string smjer = "";  
        public int brojucenika;  
        public void Metoda()  
        {  
            Console.WriteLine("Pozdrav, prvi učenik u dnevniku je Marinela");  
        }  
    }  
}  
  
class UcenikPrvi:Razred  
{  
    public string ime = "";  
}
```

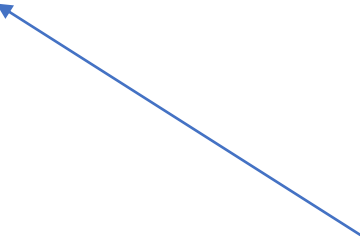
Učahurivanje (enkapsulacija)

- Korisnici objekta ne mogu mijenjati unutrašnja stanja i metode objekta kojim se koriste. Bilo kojem objektu može se pristupiti samo preko njegovih javno dostupnih članova.

Samu varijablu (*ime*) postavili smo kao privatnu, a zatim smo ju učahurili u svojstvo (ImeSvojstva).

```
using System;
using System.Security.Cryptography.X509Certificates;

namespace ucahurivanje
{
    class Program
    {
        static void Main(string[] args)
        {
            Proba podatak = new Proba();
            Console.WriteLine($"{podatak.Ime}");
        }
    }
    class Proba
    {
        private string ime=„mojeime“;
        public string Ime
        {
            get { return ime; }
            set { ime = value; }
        }
    }
}
}
```

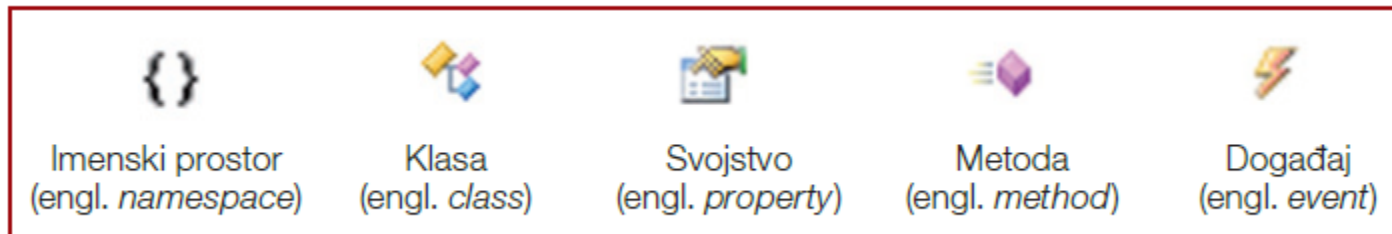


Polimorfizam

- Prikaz metoda koje su po nazivu iste ali drugačije djeluju. Svaka metoda se nalazi u svojoj klasi.

```
class Trokut
{
    public void Crtaj()
    {....
    }
}
class Krug
{
    public void Crtaj()
    {....
    }
}
```

- Više o ovom dijelu kasnije 😊



Operator točka . omogućava nam kretanje kroz hijerarhiju .NET objektnog modela u smjeru odozgo prema dolje.

```
Imenski prostor.Klasa.Metoda();
```

Pitanja

- Kakav je hijerarhijski ustroj u C#?
- Čemu služi naredba using?
- Gdje koristimo učahurivanje?
- Razlika između private i public modifikatora pristupa.

C#

- C# (CSharp) je nastao u tvrtki Microsoft i razvijen je od strane tima stručnjaka predvođenih sa Anders Hejlsberg-om i Scott Wiltamuth-om.
- pojavio se 2000.godine zajedno sa .NET platformom.
- Jednostavan, siguran, moderan i OOP
- Nastao na temelju objektnih jezika Java, C++ i Visual Basic.
- C# za razliku od Jave nije neovisan o platformi, tj. operativnom sustavu, već je kreiran za izradu stolnih (desktop) i Internet aplikacija u .Microsoft .NET okruženju.
- Sadrži 80 ključnih riječi

C# Keywords

Abstract	As	Base	Bool	Break	Ulong	Unit
Byte	Case	Catch	Char	Event	Value	While
Explicit	Extern	False	Finally	Fixed	Set	ReadOnly
Float	For	Foreach	Namespace	New	Short	Ref
Null	Object	Operator	Out	Override	Unchecked	Return
Params	Private	Static	String	Struct	Virtual	Sbyte
Switch	This	Throw	True	Try	Sizeof	Sealed
Typeof	Checked	Class	Const	Continue	Unsafe	Using
Decimal	Default	Delegate	Double	Do	Volatile	
Else	Enum	Get	Goto	If	Stackalloc	
Implicit	In	Int	Interface	Internal	Ushort	
Is	Lock	Long	Protected	Public	Void	

Tipovi podataka

- Navođenjem tipa podataka određujemo način spremanja podatka u memoriji, skup njegovih mogućih vrijednosti te skup mogućih akcija nad tim podatkom.
- Dijelimo ih na:
 - ✓ Ugrađene tipove podataka (int)
 - ✓ Korisnički definirane tipove (korisničke definirane klase iz kojih stvaramo nove objekte)
- Prema načinu spremanja:
 - ❖ Vrijednosne – spremaju se na stog (eng. stack) -(Radna memorija) ** varijable
 - ❖ Referentne - spremaju se na *hrpi* (eng. Heap) – područje memorije odvojeno od memorijskog bloka u kojem se izvršava program. Dodjela samo po potrebi ** objekti

Ugrađeni jednostavni tipovi podataka

Tip	Veličina (u bajtovima)	Raspon vrijednosti
bool	1	<i>true</i> ili <i>false</i>
byte	1	od 0 do 255 (od 0 do 2^8-1)
char	2	pojedinačni Unicode znak
sbyte	1	od -128 do 127 (od -2^7 do 2^7-1)
short	2	od -32 768 do 32 767 (od -2^{15} do $2^{15}-1$)
ushort	2	od 0 do 65 535 (od 0 do $2^{16}-1$)
int	4	od -2 147 483 648 do 2 147 483 647 (od -2^{31} do $2^{31}-1$)
uint	4	od 0 do 4 294 967 295 (od 0 do $2^{32}-1$)
float	4	7 decimala, od $\pm 1,5 \times 10^{-45}$ do $\pm 3,4 \times 10^{38}$
double	8	15-16 decimala, od $\pm 5,0 \times 10^{-324}$ do $\pm 3,4 \times 10^{308}$
decimal	16	28-29 decimala, od $\pm 1,0 \times 10^{-28}$ do $\pm 3,4 \times 10^{28}$
long	8	od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807 (od -2^{63} do $2^{63}-1$)
ulong	8	od 0 do 18 446 744 073 709 551 615 (od 0 do $2^{64}-1$)

Kontrolni znakovi

Znak	Značenje
\'	Jednostruki navodnik (apostrof)
\”	Dvostruki navodnik
\\	Obrnuta kosa crta (engl. <i>backslash</i>)
\0	Null
\a	Upozorenje
\b	Brisanje unatrag
\f	Nova stranica
\n	Novi red
\r	Prijelaz u sljedeći red
\t	Vodoravni tabulator
\v	Okomiti tabulator

Logički tip podatka - *bool*

- Dvije dopuštene vrijednosti: true ili false
- Bitan u uvjetnim izrazima

Znakovni tip podatka - *string*

Referentni tip podatka

Podatci tipa string pišu se unutar dvostrukih navodnika

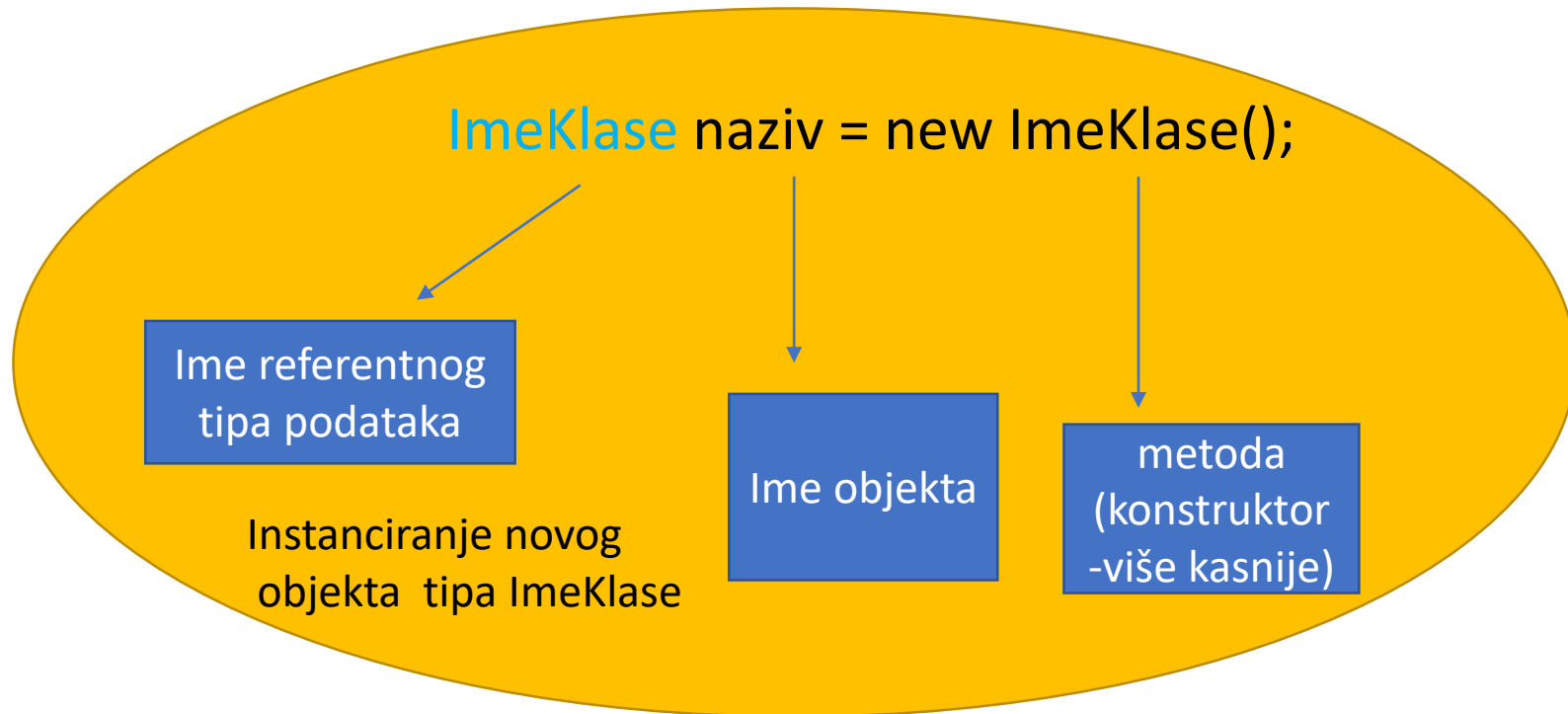
"Ovo je string"

Varijable

- Deklaracija
- Inicijalizacija

- Varijabla vrijednosnog tipa
 - `int broj = 5;`

- Varijable referentnog tipa
 - Operator *new*



Konstante

Unutar C# postoje 3 vrste konstanti:

1. Literali
2. Simboličke konstante
3. Enumeracija

Vrijednosni tip sastavljen od popisa imena brojčanih konstanti



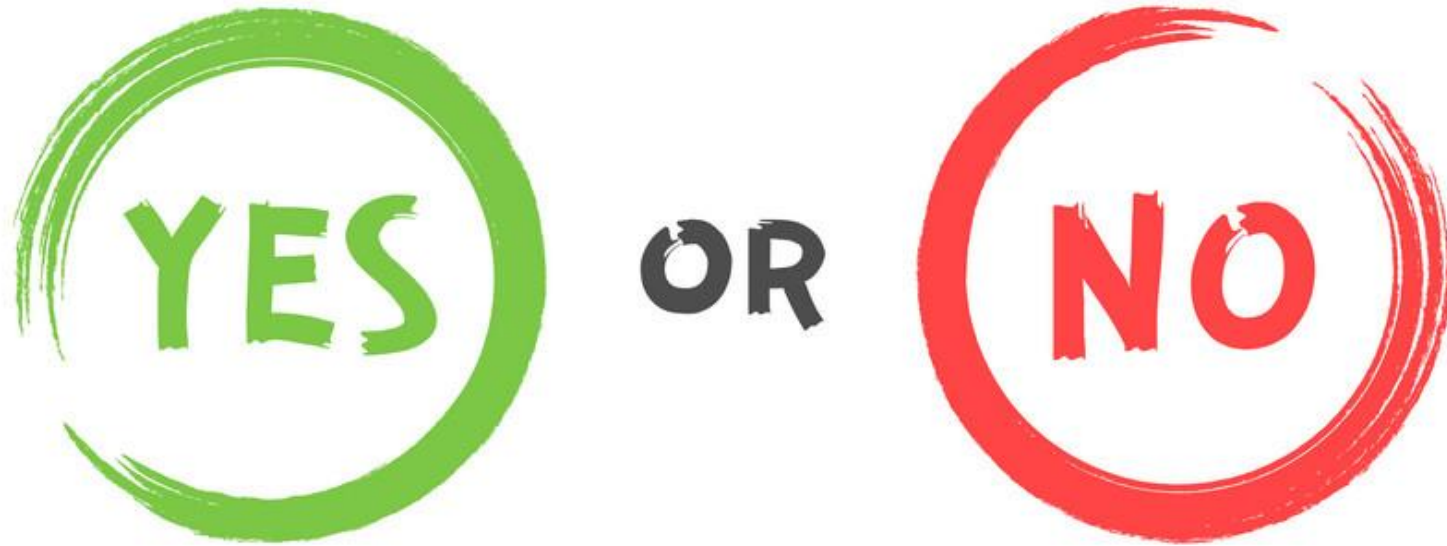
Enumeraciju ne možemo definirati unutar metode, već kao člana klase ili imenskog prostora

doslovno navedena vrijednost

`const tip naziv = vrijednost;`

```
using System;
namespace Enum
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine((int)ime.const3);
        }
        enum ime
        {
            const1 = 50,
            const2 = 67,
            const3 = 90
        }
    }
}
```

Var



„Šta je uopće var??“

Pretvaranje tipova

Implicitna pretvorba

```
int x=3;  
double y=3;
```

//razlika?

```
double y=3;  
int x=y;
```

EksPLICITna pretvorba

```
double y=3;  
int x=(int)y;
```


Operatori



```
using System;
namespace Dijeljenje
{
class Program
{
static void Main(string[] args)
{
int x1 = 10;
int y1 = 3;
double x2 = 10;
double y2 = 3;
Console.WriteLine("Dijeljenje tipa int: {0} / {1} =
{2}", x1, y1, x1 / y1);
Console.WriteLine("Ostatak pri dijeljenju {0} s {1} je
{2}.", x1, y1, x1 % y1);
Console.WriteLine("Dijeljenje tipa double: {0} / {1} =
{2}", x2, y2, x2 / y2);
Console.ReadKey();
}
}
}
```

Petlje

```
for (int i = 0; i < n; i++)  
{  
  ...  
}
```

```
while (true)  
{  
  ...  
}
```

```
do  
  ...  
} while (true);
```

Vježba

- Napisati program koji će ispisati sve brojeve od 10 do 1.
- Napisati program koji će ispisati sve parne brojeve od 5 do 27.

Nizovi

- **Niz** je skup varijabli koje imaju isto ime, a međusobno se razlikuju po indeksu.

```
int[] niz = new int[N];
```

```
int[] niz = new int[] { 10, 102, 10, 321, 312, 432, 423, 42 };
```

```
static void Main(string[] args)
{
    int N = 10;
    int[] niz = new int[N];
    for (int i = 0; i < N; i++)
    {
        Console.WriteLine("Unesite {0}. broj: ", i);
        niz[i] = int.Parse(Console.ReadLine());
    }
    for (int i = 0; i < N; i++)
    {
        Console.WriteLine("niz od {0}" ,niz[i]);
    }
}
```

```
static void Main(string[] args)
{
    int[] niz = new int[10];

    for (int i = 0; i < niz.Length; i++)
    {
        Console.WriteLine("Unesite {0}. broj: ", i);
        niz[i] = int.Parse(Console.ReadLine());
    }
    for (int i = 0; i < niz.Length; i++)
    {
        Console.WriteLine("niz od {0}" ,niz[i]);
    }
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("Unesite jednu riječ:");

    string rijec = Console.ReadLine();

    Console.WriteLine("Unesena riječ je: ");

    for (int i = 0; i < rijec.Length; i++)
    {
        Console.Write("{0},", rijec[i]);

    }

}
```

Zadatak

Učitati 5 brojeva u polje. Svaki element kvadrirati. Kvadrante ispisati na zaslonu. Unutar zadatka postaviti funkciju koja omogućava da se nakon svakog unosa mora pritisnuti jedna tipka tipkovnice.



Metode



ToString metoda

Metoda ili svojstvo	Svrha
Compare	Statička metoda koja uspoređuje dva niza znakova.
Concat	Statička metoda koja spaja više nizova znakova u jedan.
EndsWith	Označava završava li pozivajući niz znakova zadanim znakom.
IndexOf	Vraća poziciju prvog pojavljivanja uzorka unutar niza znakova.
Insert	Umeće navedeni tekst u niz znakova.
Length	Daje duljinu stringa.
Replace	Unutar pozivajućeg niza znakova zamjenjuje navedeni stari podniz novim.
Remove	Briše zadani broj znakova.
Split	Razdvaja niz znakova na podnizove prema navedenom znaku razdvajanja.
StartsWith	Označava počinje li pozivajući niz znakova zadanim znakom.
Substring	Dohvaća podniz znakova.
ToLower	Pretvara niz znakova u mala slova.
ToUpper	Pretvara niz znakova u velika slova.
Trim	Uklanja bjeline s početka i završetka niza znakova.

Zadatak...primjeni sve metode klase String

```
string recenica = " Danas radimo s klasom String";
```



```

static void Main(string[] args)
{
    string recenica = "DAnas radimo s klasom String ";

    Console.WriteLine("Broj znakova u rečenici: {0}", recenica.Length);

    Console.WriteLine("\n-- ToLower i ToUpper--");
    Console.WriteLine(recenica.ToLower());
    Console.WriteLine(recenica.ToUpper());

    Console.WriteLine("\n-- SubString--");
    Console.WriteLine(recenica.Substring(6));
    Console.WriteLine(recenica.Substring(0, 5));

    Console.WriteLine("\n-- Replace--");
    Console.WriteLine(recenica.Replace("Danas", "Sutra"));

    Console.WriteLine("\n-- Split--");
    string[] rijeci = recenica.Split(' ');
    for (int i = 0; i < rijeci.Length; i++)
    {
        Console.WriteLine(rijeci[i]);
    }

    Console.WriteLine("\n-- IndexOf i Insert--");
    int pozicija = recenica.IndexOf("radimo");
    Console.WriteLine(pozicija);
    if (pozicija >= 0)
    {
        Console.WriteLine(recenica.Insert(pozicija, "intenzivno "));
    }

    Console.WriteLine(recenica.EndsWith("f"));

    Console.WriteLine(recenica.Trim());
    Console.ReadKey();
}

```

```

Broj znakova u recenici: 30

-- ToLower i ToUpper--
danas radimo s klasom string
DANAS RADIMO S KLASOM STRING

-- SubString--
radimo s klasom String
DAnas

-- Replace--
DAnas radimo s klasom String

-- Split--
DAnas
radimo
s
klasom
String

-- IndexOf i Insert--
6
DAnas intenzivno radimo s klasom String
False
DAnas radimo s klasom String

```

DateTime metoda

- Za dohvaćanje trenutnog datuma i vremena rabi se statičko svojstvo

Now

```
DateTime datumIvrijeme = DateTime.Now;  
Console.WriteLine(datumIvrijeme);
```

```
DateTime datum = DateTime.Parse(Console.ReadLine());
```

Instanciranje datuma

```
DateTime datum = new DateTime(2011, 7, 4);
```

```
DateTime datumIVrijeme = new DateTime(2011, 6, 29, 12, 35, 12);
```

6/29/2011 12:35:12 PM

Metoda ili svojstvo	Opis
AddYears	Pozivajućoj instanci dodaje zadani broj godina.
AddMonths	Pozivajućoj instanci dodaje zadani broj mjeseci.
AddDays	Pozivajućoj instanci dodaje zadani broj dana.
AddHours	Pozivajućoj instanci dodaje zadani broj sati.
AddMinutes	Pozivajućoj instanci dodaje zadani broj minuta.
Year	Iz pozivajuće instance vraća godinu.
Month	Vraća mjesec.
Day	Vraća dan.
Hour	Vraća sat.
Minute	Vraća minute.
Second	Vraća sekunde.
DayOfWeek	Vraća dan u tjednu.
DayOfYear	Vraća dan u godini.
ToLongDateString	Vraća duži oblik zapisa datuma (reg. postavke).
ToShortDateString	Vraća kraći oblik zapisa datuma.
ToLongTimeString	Vraća duži oblik zapisa vremena.
ToShortTimeString	Vraća kraći oblik zapisa vremena.
ToString	Vraća datumski zapis prema zadanom formatu.

System.Collections

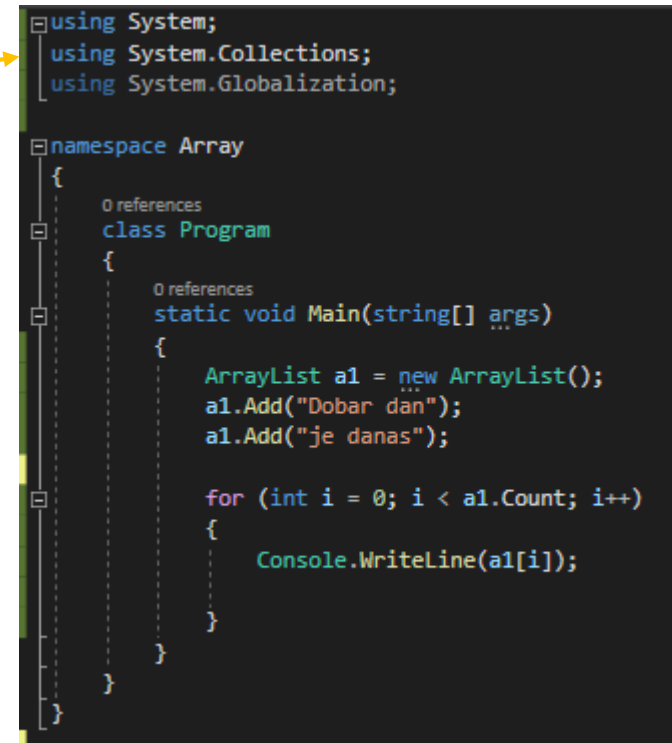
- Kolekcija- klasa koja omogućuje rad s odgovarajućom kolekcijom sastavljenom od više elemenata.
- Kolekcije:
 - ✓ ArrayList
 - ✓ List<T>

Klasa ArrayList

- Prednost u odnosu na niz - nema fiksni broj elemenata
- Ispred definicije klase u koju dodajemo ArrayList potrebno je uključiti imenski prostor

- Instanciranje novog objekta:

```
ArrayList a1 = new ArrayList();
```

A screenshot of a code editor showing C# code. The code includes using statements for System, System.Collections, and System.Globalization. It defines a namespace 'Array' containing a class 'Program' with a 'Main' method. Inside 'Main', an ArrayList 'a1' is instantiated, two strings are added, and a loop prints each element. A yellow arrow points from the text 'potrebno je uključiti imenski prostor' to the 'using System.Collections;' line.

```
using System;
using System.Collections;
using System.Globalization;

namespace Array
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            ArrayList a1 = new ArrayList();
            a1.Add("Dobar dan");
            a1.Add("je danas");

            for (int i = 0; i < a1.Count; i++)
            {
                Console.WriteLine(a1[i]);
            }
        }
    }
}
```

Bitnije metode i svojstva ArrayList objekta

Metoda ili svojstvo	Svrha
Add	Dodaje novi element u kolekciju.
Clear	Uklanja sve elemente iz kolekcije.
Contains	Utvrdjuje pripada li element kolekciji.
IndexOf	Traži prosljedeni objekt u kolekciji i vraća njegov indeks.
Insert	Umeće novi element u kolekciju na zadanu poziciju.
Remove	Uklanja objekt iz kolekcije.
RemoveAt	Uklanja element na zadanoj poziciji iz kolekcije.
Reverse	Obrće redoslijed elemenata u kolekciji.
Sort	Sortira kolekciju.
ToArray	Kopira elemente kolekcije u novi niz zadanog tipa.

Klasa List<T>

- Prema funkciji posjeduje iste funkcionalnosti kao i klasa ArrayList (broj elemenata nije fiksna, elemente možemo dohvaćati preko indeksa....), **ali zahtijeva da njezini elementi moraju biti točno određenog tipa.**
- Spada u skupinu *generičkih* kolekcija
`using System.Collections.Generic;`

Instanciranje novog objekta klase List<T>

```
List<T> ts = new List<T>();
```

T predstavlja tip elemenata

Primjer:

```
List<string> ts = new List<string>();
```

```
static void Main(string[] args)
{
    string ime = "Filip";
    string godina = "20";
    List<string> ts = new List<string>();
    ts.Add(ime);
    ts.Add(godina);

    foreach (string item in ts)
    {
        Console.WriteLine(item);
    }
}
```

```
static void Main(string[] args)
{
    List<string> ts = new List<string>()
    {
        "Marko",
        "Petar",
        "Lucija"
    };

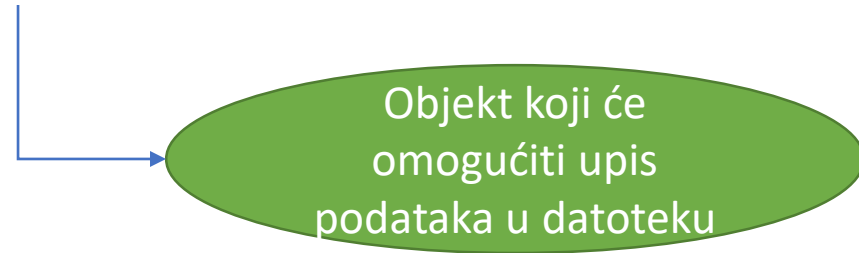
    foreach (string item in ts)
    {
        Console.WriteLine(item);
    }
}
```

Klasa imenskog prostora System.IO;

- Klase:
 1. Klasa Directory
 2. Klasa File
 3. Klasa Path
 4. Klasa DirectoryInfo
 5. Klasa FileInfo
- Spremanje teksta u datoteku
- Čitanje sadržaja tekstualne datoteke
- Otpuštanje resursa i objekta

Spremanje teksta u datoteku

```
StreamWriter sw = new StreamWriter("datoteka.txt");
```



```
sw.WriteLine("Ovo je linija u datoteci");  
sw.Close(); // poziv na kraju za zatvaranje datoteke
```



```
using System;
using System.IO;
namespace SpremanjeTeksta
{
class Program
{
static void Main(string[] args)
    {
        Console.WriteLine("Unesite ime: ");
        string ime = Console.ReadLine();
        Console.WriteLine("Unesite prezime: ");
        string prezime = Console.ReadLine();
        StreamWriter sw = new
        StreamWriter(@"D:\MyDocuments\NOOP\datoteka.txt");
        sw.WriteLine("Ime: {0}", ime);
        sw.WriteLine("Prezime: {0}", prezime);
        sw.Close();
    }
}
}
```

Čitanje sadržaja tekstualne datoteke

```
StreamReader sr = new StreamReader("datoteka.txt");
```

```
string savTekstDatoteke = sr.ReadToEnd();
```



Objekt/Konstruktor()

```
DateTime date = new DateTime();
```

↳ Konstruktor

- Pri **instanciranju objekta** pozivom operatora ***new*** zapravo se poziva konstruktor klase.
- Metoda koja se zove isto kao i klasa i nema posebno naveden tip, već je njezin tip zapravo sama klasa.
- Konstruktor nema povratnog tipa (niti void), ime konstruktora mora biti jednako kao i ime klase u kojoj je definiran
- Konstruktor – vraća novi objekt te klase.
- Tipovi konstruktora:
 - Podrazumijevani konstruktor
 - Parametrizirani konstruktor

Podrazumijevani konstruktor

Microsoft Visual Studio Debug Console

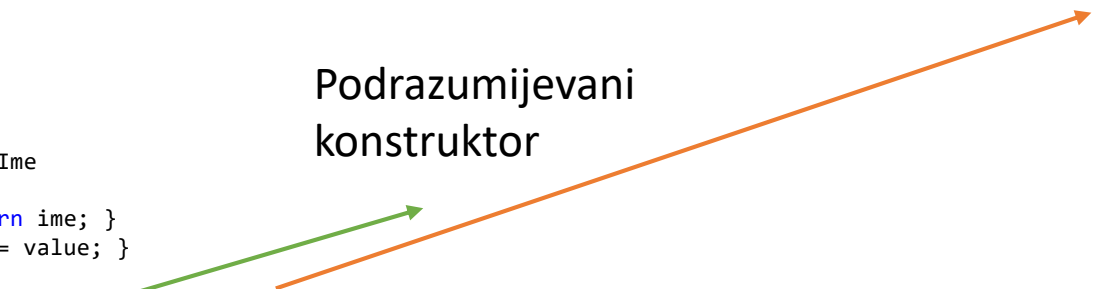
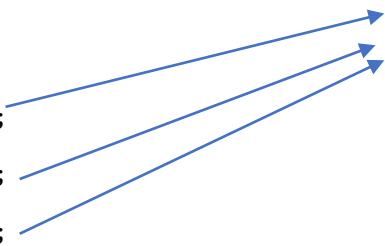
```
Mijo  
Marta  
Nije definirano
```

```
using System;  
  
namespace Konstruktori  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Ucenik ucenik1 = new Ucenik();  
            ucenik1.Ime = "Mijo";  
            Ucenik ucenik2 = new Ucenik();  
            ucenik2.Ime = "Marta";  
            Ucenik ucenik3 = new Ucenik();  
  
            Console.WriteLine(ucenik1.Ime);  
            Console.WriteLine(ucenik2.Ime);  
            Console.WriteLine(ucenik3.Ime);  
        }  
    }  
}  
  
class Ucenik  
{  
    string ime;  
  
    public string Ime  
    {  
        get { return ime; }  
        set { ime = value; }  
    }  
  
    public Ucenik()  
    {  
        ime = "Nije definirano";  
    }  
}
```

Konstruktor klase *Ucenik*

Ne moramo ga čak niti navoditi

Podrazumijevani konstruktor



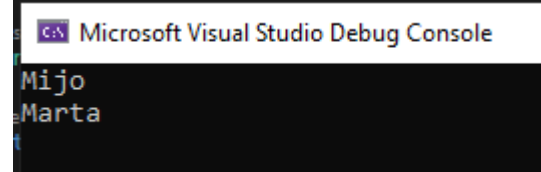
```
using System;

namespace Konstruktori
{
    class Program
    {
        static void Main(string[] args)
        {
            Ucenik ucenik1 = new Ucenik();
            ucenik1.Ime = "Mijo";
            Ucenik ucenik2 = new Ucenik();
            ucenik2.Ime= "Marta";
            Ucenik ucenik3 = new Ucenik();

            Console.WriteLine(ucenik1.Ime);
            Console.WriteLine(ucenik2.Ime);
            Console.WriteLine(ucenik3.Ime);

        }
    }
    class Ucenik
    {
        string ime;

        public string Ime
        {
            get { return ime; }
            set { ime = value; }
        }
    }
}
```



Microsoft Visual Studio Debug Console

```
Mijo
Marta
```

```
using System;

namespace Konstruktori
{
    class Program
    {
        static void Main(string[] args)
        {
            Ucenik ucenik = new Ucenik("Mark", „Markovic");
            Console.WriteLine(ucenik.ime + " " + ucenik.prezime);

        }
    }
    class Ucenik
    {
        public string ime;
        public string prezime;

        public Ucenik( string imeUcenika, string prezimeUcenika)
        {
            ime = imeUcenika;
            prezime =prezimeUcenika;

        }

    }
}
```

Parametrizirani konstruktor

