

Potrebno je navesti korištenje Generic klase!

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

Generička klasa

Generička klasa objedinjuje operacije koje nisu specifične za određeni tip podatka.

Lista objekata tipa string

```
List<string> gradovi = new()  
{  
    "Osijek",  
    "Donji Miholjac"  
};  
  
foreach (string s in gradovi)  
{  
    Console.WriteLine(s);  
}
```

Terminal – Predavanje7

```
Osijek  
Donji Miholjac
```

Lista objekata tipa int

```
List<int> brojevi = new()  
{  
    2,3,4,8  
};  
  
foreach (int b in brojevi)  
{  
    Console.WriteLine(b);  
}
```

Terminal – Predavanje7

```
2  
3  
4  
8
```

Lista objekata tipa DateTime

```
List<DateTime> datumi = new()  
{  
    new DateTime(),  
    new DateTime(2021,4,1),  
    new DateTime(2022, 2, 12,14,22,30)  
};  
  
foreach (DateTime d in datumi)  
{  
    Console.WriteLine(d);  
}
```

Terminal – Predavanje7

```
1/1/0001 12:00:00 AM  
4/1/2021 12:00:00 AM  
2/12/2022 2:22:30 PM
```

Izvori:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/generic-classes>

<https://www.tutorialsteacher.com/csharp/csharp-generics>

Generička klasa

Rad s vlastitom klasom u generičkoj listi

```
List<Tvrtka> poslovniPartneri = new()
{
    new Tvrtka()
    {
        Sifra=1,
        Naziv="Inchoo"
    },
    new Tvrtka()
    {
        Sifra = 2,
        Naziv = "Mono"
    }
};

foreach (Tvrtka t in poslovniPartneri)
{
    Console.WriteLine(t.Naziv);
}
```

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }
    }
}
```

> Terminal – Predavanje7

```
Inchoo
Mono
```

Generička klasa

Kreiranje i korištenje vlastite generičke klase

Klasa za opis tvrtke

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }

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

Generička klasa za pohranu u bazu podataka

```
using System;
namespace Predavanje7
{
    public class DBA<T> where T: Tvrtka
    {
        public T Objekt { get; set; }

        public void SpremiUBazu()
        {
            Console.WriteLine("Spremam u bazu " + Objekt.Naziv);
        }
    }
}
```

Korištenje

```
DBA<string> db1 = new();

db1.Objekt = "Niz znakova";

db1.SpremiUBazu();

DBA<Tvrtka> db2 = new();

db2.Objekt = new Tvrtka()
{
    Sifra = 1,
    Naziv = "Inchoo"
};

db2.SpremiUBazu();
```

> Terminal – Predavanje7

```
Spremam u bazu Niz znakova
Spremam u bazu Inchoo
```

Generička klasa

Kreiranje i korištenje vlastite generičke klase s primjerom nasljeđivanja

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }

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

```
using System;
namespace Predavanje7
{
    public class Dobavljac: Tvrtka
    {
        public decimal Potrazuje { get; set; }
    }
}
```

```
using System;
namespace Predavanje7
{
    public class Kupac: Tvrtka
    {
        public decimal Duguje { get; set; }
    }
}
```

Generička klasa

Kreiranje i korištenje vlastite generičke klase s primjerom nasljeđivanja

```
using System;
namespace Predavanje7
{
    public class DBA<T> where T: Tvrtka
    {
        public T Objekt { get; set; }

        public void SpremiUBazu()
        {
            Console.WriteLine("Spremam u bazu " + Objekt.Naziv);
        }
    }
}
```

```
DBA<Dobavljac> db1 = new();
db1.Objekt = new Dobavljac()
{
    Sifra = 1,
    Naziv = "Inchoo",
    Potrazuje=12000
};
db1.SpremiUBazu();
```

```
DBA<Kupac> db2 = new();
db2.Objekt = new Kupac()
{
    Sifra = 1,
    Naziv = "Inchoo",
    Duguje = 12000
};
db2.SpremiUBazu();
```

> Terminal – Predavanje7

```
Spremam u bazu Inchoo
Spremam u bazu Mono
```

Atributi

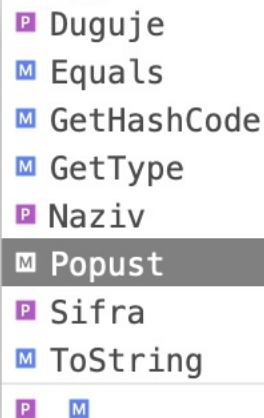
Atributi omogućuju definiranje meta podataka u kodu

- koristi se često prilikom označavanja podataka u kontekstu baze podataka

```
using System;
namespace Predavanje7
{
    public class Kupac: Tvrtka
    {
        public decimal Duguje { get; set; }

        [Obsolete("Ne dajemo više popust")]
        public decimal Popust(decimal iznos)
        {
            return Duguje * ((100 - iznos) / 100);
        }
    }
}
```

```
var k = new Kupac();
k.|
```



A dropdown menu showing the members of the Kupac class. The members are: Duguje (Property), Equals (Method), GetHashCode (Method), GetType (Method), Naziv (Property), Popust (Method), Sifra (Property), and ToString (Method). The Popust method is currently selected and highlighted.

P	Duguje
M	Equals
M	GetHashCode
M	GetType
P	Naziv
M	Popust
P	Sifra
M	ToString
P	
M	

[deprecated] decimal Kupac.Popust(decimal iznos)

depricated: preporuka ne korištenja, u budućim verzijama će se označena metoda izbaciti

Izvori:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/attributes/>

<https://www.infoworld.com/article/3543302/how-to-use-data-annotations-in-csharp.html>

Sučelja

Implementacija IComparable sučelja

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }

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

```
using System;
namespace Predavanje7
{
    public class Dobavljac:Tvrtka, IComparable<Dobavljac>
    {
        public decimal Potrazuje { get; set; }

        public int CompareTo(Dobavljac d)
        {
            return Naziv.CompareTo(d.Naziv);
        }
    }
}
```

```
List<Dobavljac> dobavljac = new()
{
    new Dobavljac()
    {
        Sifra = 1,
        Naziv = "Mono"
    },
    new Dobavljac()
    {
        Sifra = 2,
        Naziv = "Inchoo"
    }
};

dobavljac.Sort();

foreach (Dobavljac d in dobavljac)
{
    Console.WriteLine(d.Naziv);
}
```

>- Terminal – Predavanje7

Inchoo
Mono

Sučelja

Kreiranje i korištenje vlastitog sučelja

```
using System;
namespace Predavanje7
{
    public interface IPosao
    {
        void UnosPodataka(String Naziv);
        int DuzinaNaziva();
        void Ispis();
    }
}
```

```
using System;
namespace Predavanje7
{
    public class ObradaKupac: IPosao
    {
        private String Naziv;

        public int DuzinaNaziva()
        {
            return Naziv == null ? 0 : Naziv.Length;
        }

        public void Ispis()
        {
            Console.WriteLine(Naziv);
        }

        public void UnosPodataka(string Naziv)
        {
            this.Naziv = Naziv;
        }
    }
}
```

```
using System;
namespace Predavanje7
{
    public class ObradaDobavljac : IPosao
    {
        public int DuzinaNaziva()
        {
            return 0;
        }


        public void Ispis()
        {
            Console.WriteLine("Prazno");
        }

        public void UnosPodataka(string Naziv)
        {
        }
    }
}
```


Sučelja

Kreiranje i korištenje vlastitog sučelja

```
IPosao obrada = new ObradaDobavljac();  
obrada.UnosPodataka("Inchoo");  
obrada.Ispis();  
  
obrada = new ObradaKupac();  
obrada.UnosPodataka("Mono");  
obrada.Ispis();
```

 Terminal – Predavanje7

Prazno
Mono

Enumeracije

Enumeracije su vrijednosni tipovi definirani skupom imenovanih konstanti koje prate brojevne vrijednosti

```
using System;
namespace Predavanje7
{
    public enum CRUD
    {
        Pregled,
        Unos,
        Promjena,
        Brisanje
    }
}
```

 `CRUD.Brisanje = 3`

```
public class ObradaKupac: IPosao
{
    public CRUD Akcija { get; set; }

    private String Naziv;

    public int DuzinaNaziva()
    {
        return Naziv == null ? 0 : Naziv.Length;
    }


    public void Ispis()
    {
        Console.WriteLine(Naziv);
    }

    public void UnosPodataka(string Naziv)
    {
        this.Naziv = Naziv;
    }
}
```

```
var obradaKupac = new ObradaKupac();
obradaKupac.Akcija = CRUD.Brisanje;

if (obradaKupac.Akcija == CRUD.Pregled)
{
    Console.WriteLine("Trenutno pregledavam");
}

Console.WriteLine(obradaKupac.Akcija);
Console.WriteLine((int)obradaKupac.Akcija);
```

 **Terminal – Predavanje7**

```
Brisanje
3
```

Strukture podataka: rječnik, lista, red, stog

Strukture podataka koristimo kada želimo organizirati pohranu u programskom jeziku
- jednostavnu strukturu niz (polja – array) smo prethodno upoznali

Osnovne strukture dostupne u C#

- Rječnik (Dictionary)
- Lista (List)
- Red (Queue) – FIFO
- Stog (Stack) – LIFO

I want to...	Generic collection options	Non-generic collection options	Thread-safe or immutable collection options
Store items as key/value pairs for quick look-up by key	Dictionary<TKey,TValue>	Hashtable (A collection of key/value pairs that are organized based on the hash code of the key.)	ConcurrentDictionary<TKey,TValue> ReadOnlyDictionary<TKey,TValue> ImmutableDictionary<TKey,TValue>
Access items by index	List<T>	Array ArrayList	ImmutableList<T> ImmutableArray
Use items first-in-first-out (FIFO)	Queue<T>	Queue	ConcurrentQueue<T> ImmutableQueue<T>
Use data Last-In-First-Out (LIFO)	Stack<T>	Stack	ConcurrentStack<T> ImmutableStack<T>
Access items sequentially	LinkedList<T>	No recommendation	No recommendation
Receive notifications when items are removed or added to the collection. (implements INotifyPropertyChanged and INotifyCollectionChanged)	ObservableCollection<T>	No recommendation	No recommendation
A sorted collection	SortedList<TKey,TValue>	SortedList	ImmutableSortedDictionary<TKey,TValue> ImmutableSortedSet<T>
A set for mathematical functions	HashSet<T> SortedSet<T>	No recommendation	ImmutableHashSet<T> ImmutableSortedSet<T>

Izvor:

<https://docs.microsoft.com/en-us/dotnet/standard/collections/>

Rječnik

Skupina zapisa uređena prema principu ključ – vrijednost

- ključevi su jedinstveni, u rječniku ne mogu biti dva ključa s istim vrijednostima

Osnovna deklaracija i postavljanje vrijednosti

```
Dictionary<string, string> akronimi =  
    new Dictionary<string, string>();  
  
akronimi.Add("FIFO", "First In First Out");  
akronimi.Add("LIFO", "Last In First Out");
```

Pojednostavljena deklaracija i postavljanje vrijednosti

```
Dictionary<string, string> akronimi =  
    new()  
    {  
        { "FIFO", "First In First Out" },  
        { "LIFO", "Last In First Out" }  
    };
```

Dohvaćanje podataka u rječniku

```
Console.WriteLine("Pojedinačno: " + akronimi["FIFO"]);  
  
foreach (KeyValuePair<string, string> kv in akronimi)  
{  
    Console.WriteLine("Key = {0}, Value = {1}",  
        kv.Key, kv.Value);  
}
```

Terminal – Predavanje7

```
Pojedinačno: First In First Out  
Key = FIFO, Value = First In First Out  
Key = LIFO, Value = Last In First Out
```

Izvor:

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=net-5.0>

Rječnik

Korištenje vlastite klase

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }

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

```
Dictionary<string, Tvrtka> kategorije =
    new()
    {
        { "web", new Tvrtka() { Naziv="Inchoo"} },
        { "desktop", new Tvrtka() { Naziv = "Mono" } }
    };
};
```

```
Console.WriteLine("Za razvoj mrežnih aplikacija radim s {0}",
    kategorije["web"]);
```

> Terminal – Predavanje7

Za razvoj mrežnih aplikacija radim s Inchoo

Značajnije metode: Add, ContainsKey, ContainsValue, Remove

Lista

Skupina zapisa objekata kojima možemo pristupiti pomoću indeksa

- indeks počinje vrijednošću 0
- korištenje prikazana tijekom gradiva vezanog za generičke klase

Pojednostavljena deklaracija i postavljanje vrijednosti

```
List<string> gradovi = new()  
{  
    "Osijek",  
    "Donji Miholjac"  
};
```

Značajnije metode: Add, Contains, Exists, Insert, Remove

Izvor:

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-5.0>

Dohvaćanje podataka iz liste

```
Console.WriteLine("Pojedinačno " + gradovi[0]);  
  
foreach(string s in gradovi)  
{  
    Console.WriteLine(s);  
}
```

Terminal – Predavanje7

```
Pojedinačno Osijek  
Osijek  
Donji Miholjac
```

Red

Skupina zapisa organizirana prema principu prvi unutra, prvi van (FIFO)
- omogućuje implementaciju obrade prema redoslijedu dolazaka

Osnovna deklaracija i stavljanje u red (Enqueue)

```
Queue<string> redoslijed = new();  
redoslijed.Enqueue(".html");  
redoslijed.Enqueue(".css");  
redoslijed.Enqueue(".js");
```

Temeljne metode za rad: Peek i Dequeue

```
Console.WriteLine("Sljedeći za obradu: " + redoslijed.Peek());  
var trenutniUObradi = redoslijed.Dequeue();  
Console.WriteLine("Trenutno obrađujem: " + trenutniUObradi);  
Console.WriteLine("Preostalo za obradu: " + redoslijed.Count);
```

Terminal – Predavanje7

```
Sljedeći za obradu: .html  
Trenutno obrađujem: .html  
Preostalo za obradu: 2
```

Pregled preostalog reda čekanja

```
foreach (string ekstenzija in redoslijed)  
{  
    Console.WriteLine(ekstenzija);  
}
```

Terminal – Predavanje7

```
.css  
.js
```

Izvor:

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.queue-1?view=net-5.0>

Red

Korištenje vlastite klase

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }

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

```
Queue<Tvrtka> placanja = new();
placanja.Enqueue(new Tvrtka() { Naziv = "Inchoo" });
placanja.Enqueue(new Tvrtka() { Naziv = "Mono" });

while (placanja.Count > 0)
{
    Console.WriteLine("Izvrši plaćanje za: " +
        placanja.Dequeue().Naziv);
}
```

Terminal – Predavanje7

```
Izvrši plaćanje za: Inchoo
Izvrši plaćanje za: Mono
```

Značajnije metode: Add, ContainsKey, ContainsValue, Remove

Stog

Skupina zapisa organizirana prema principu prvi unutra, zadnji van (LIFO)
- omogućuje implementaciju obrade suprotno od redoslijeda dolazaka

Osnovna deklaracija i stavljanje u stog (Push)

```
Stack<string> verzijeProgramaProdaja = new();  
verzijeProgramaProdaja.Push("Academio V1");  
verzijeProgramaProdaja.Push("Academio V2");  
verzijeProgramaProdaja.Push("Academio 2021");
```

Temeljne metode za rad: Peek i Pop

```
Console.WriteLine("Koji verziju kupiti: " +  
    verzijeProgramaProdaja.Peek());  
var trenutnoSeProdaje = verzijeProgramaProdaja.Pop();  
Console.WriteLine("Kupio: " + trenutnoSeProdaje);  
Console.WriteLine("Preostali broj verzija: " +  
    verzijeProgramaProdaja.Count);
```

Terminal – Predavanje7

```
Koji verziju kupiti: Academio 2021  
Kupio: Academio 2021  
Preostali broj verzija: 2
```

Pregled preostalog stavaka na stogu

```
foreach (string verzija in verzijeProgramaProdaja)  
{  
    Console.WriteLine(verzija);  
}
```

Terminal – Predavanje7

```
Academio V2  
Academio V1
```

Izvor:

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.stack-1?view=net-5.0>

Stog

Korištenje vlastite klase

```
using System;
namespace Predavanje7
{
    public class Tvrtka
    {
        public int Sifra { get; set; }
        public string Naziv { get; set; }

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

```
Stack<Tvrtka> zadnjiVazan = new();
zadnjiVazan.Push(new Tvrtka() { Naziv = "Inchoo" });
zadnjiVazan.Push(new Tvrtka() { Naziv = "Mono" });

while (zadnjiVazan.Count > 0)
{
    Console.WriteLine("Izvrši plaćanje za: " +
        zadnjiVazan.Pop().Naziv);
}
```

Terminal – Predavanje7

```
Izvrši plaćanje za: Mono
Izvrši plaćanje za: Inchoo
```

Značajnije metode: Add, ContainsKey, ContainsValue, Remove