# OOP

Polja i kolekcije

# U ovom poglavlju  naučit ćete

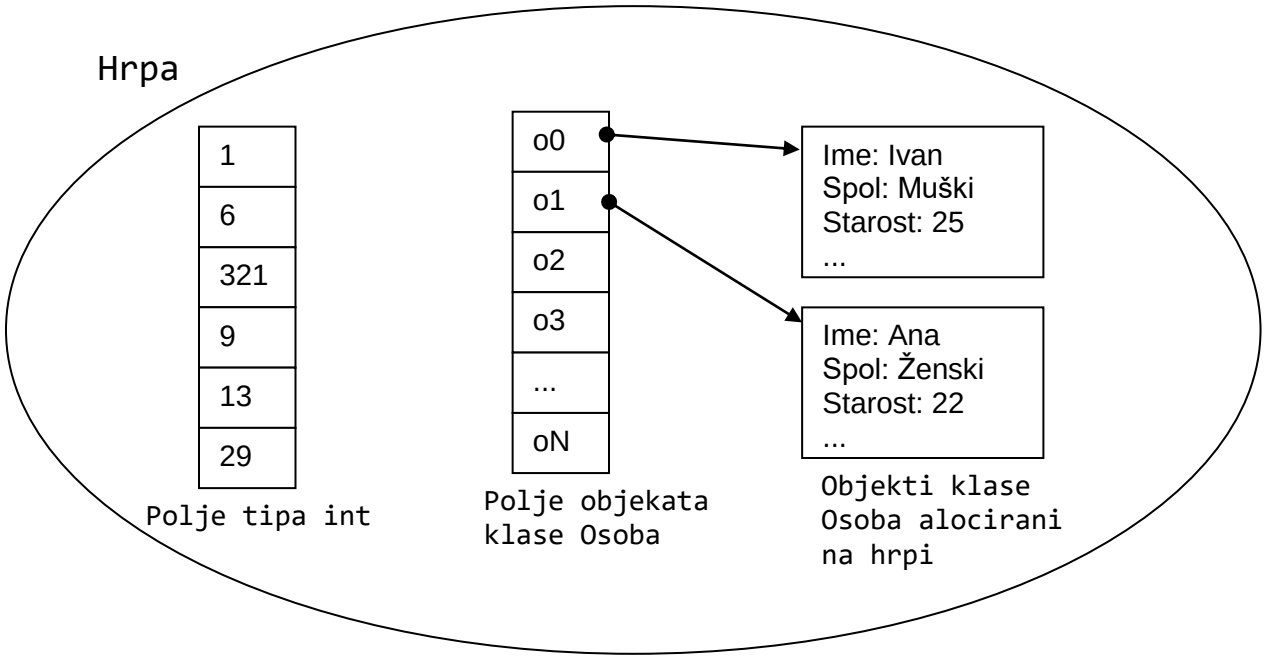
- Polja
- Iskaz foreach i ključna riječ params
- Granice polja, pretvaranje, sortiranje
- Indekseri
- Sučelja kolekcija
- Redovi, Stogovi, Rječnici

# Polje

- Polje (engl. *array*) je indeksirana kolekcija (niz, polje) objekata koji su svi istog tipa

- Instanca tipa System.Int32[] – nasljeđuje System.Array
- Deklaracija i instanciranje
- Alocirano na hrpi

# Polje..


Hrpa
1
6
321
9
13
29
Polje tipa int
o0
o1
o2
o3
...
oN
Polje objekata klase Osoba
Ime: Ivan
Spol: Muški
Starost: 25
...
Ime: Ana
Spol: Ženski
Starost: 22
...
Objekti klase Osoba alocirani na hrpi

# Polje..

- Elementi u polju su uvijek inicijalizirani
- Elementima pristupamo koristeći indeksni operator []
- Javno svojstvo Length (otkud dolazi???)

# foreach petlja

```
int[] poljeCijelihBrojeva = new int[5]; // autom. inicijalizacija elemenata
            Osoba[] poljeOsoba = new Osoba[3];  // elementi inicijalizirani na null

            // Inicijalizacija elemenata polja referentog tipa
            poljeOsoba[0] = new Osoba("Ivan", "Ivanković");
            poljeOsoba[1] = new Osoba("Ana", "Horvat");
            poljeOsoba[2] = new Osoba("Mislav", "Balković");

            // pristupanje elementima polja kroz foreach petlju
            foreach(int broj in poljeCijelihBrojeva)
            {
                Console.WriteLine(broj.ToString());
            }
            foreach(Osoba o in poljeOsoba)
            {
                Console.WriteLine(o.ToString());
            }
```

# Polje..

- Inicijalizacija prilikom kreiranja

```
int[] poljeCijelihBrojeva = new int[5] { 2, 4, 6, 8, 10 };
```

```
int[] poljeCijelihBrojeva = { 2, 4, 6, 8, 10 };
```

# Ključna riječ params

- omogućava prosljeđivanje promjenjivog broja parametara bez eksplicitnog stvaranja polja

```
class Program
    {
        static void Main(string[] args)
        {
            int[] eksplicitnoPolje = new int[5] { 2, 4, 6, 8, 10 };

            // poziv metode prosljeđivanjem eksplicitno definiranog polja
            PrikaziVrijednosti(eksplicitnoPolje);

            // poziv metode prosljeđivanjem niza vrijednosti
            PrikaziVrijednosti(5, 6, 7, 8);
        }

        static void PrikaziVrijednosti(params int[] poljeCjelobrojnihVrijednosti)
        {
            foreach (int i in poljeCjelobrojnihVrijednosti)
            {
                Console.WriteLine("PrikaziVrijednosti {0}", i);
            }
        }
    }
```

# Višedimenzionalna polja

- Pravokutna i nejednaka
  - pravokutna

```
int[,] pravokutnoIntPolje = new int[4, 3];
```

```
int[,] pravokutnoIntPolje =
{
  {0, 1, 2}, {3, 4, 5}, {6, 7, 8}, {9, 10, 11}
};
```

# Višedimenzionalna polja

- Nejednaka

```
const int brojRedaka = 4;
int[][] nejednakoIntPolje = new int[brojRedaka][];
```

# Indekseri

- konstrukcija koja dopušta da kolekcijama unutar klase pristupimo koristeći poznatu sintaksu uglatih zagrada ([])
- posebna vrsta svojstva i sadrži pristupnike get i set koji određuju njegovo ponašanje

```
[modifikator pristupa] tip this[argument tipa] {
get; set; }
```

# Kolekcije

- **List<T>**
  - Efikasno i dinamički alocirano polje
  - Postoje i druge implementacije (npr. LinkedList)
  - Nije idealan za pretraživanja
  - Sučelje IList
- **Dictionary<K,V>**
  - Implementacija hash tablice
  - Ekstremno efikasan za pretraživanje prema ključu
  - Sučelje IDictionary

# Kolekcije...

- **Stack<T>, Queue<T>**
- **HashSet<T>**
  - Nema duplikata
  - Skupovne operacije
  - Sučelje ISet

# Sortiranja kolekcija

- **IComparable<>**
- **IComparer<>**
  - Zgodno kad nemamo pristup source-u klase, ili kad želimo sortiranje po više kriterija

# Hvala na pažnji!