OOP

Klase i objekti

U ovom poglavlju naučit ćete…

- Deklariranje i definiranje klasa
- Instanciranje objekata, konstruktori
- Ključna riječ this
- Svojstva
- Statički članovi

Deklariranje klase (1)

```
[modifikator pristupa] class identifikator [:osnovna klasa [, sučelje(a)]]
{
    // tijelo klase: definicija svojstava, metoda i događaja
}
```

Deklariranje klase (2)

```
public class Osoba
{
    // javno dostupna polja
    public string Ime;
    public string Prezime;
    public int Starost;
    public string Spol;
    public string Zanimanje;

    // privatno polje
    private DateTime VrijemeInstanciranja = DateTime.Now;

    // javna metoda
    public void IspisiDetalje()
    {
        Console.WriteLine("Objekt klase Osoba - detalji:");
        Console.WriteLine(
            "Ime: " + Ime + "\n" +
            "Prezime: " + Prezime + "\n" +
            "Starost: " + Starost + "\n" +
            "Spol: " + Spol + "\n" +
            "Zanimanje: " + Zanimanje + "\n" +
            "Vrijeme instanciranja: " + VrijemeInstanciranja.ToLongTimeString()
            );
    }
}
```

Operator konkatenacije +

- operator plus (+) primijenjen na podacima tipa String predstavlja **operator konkatenacije** (lijepljenja znakova)

Modifikatori pristupa

- **Modifikator pristupa** kontrolira dostupnost elementa kojeg određuje vanjskim objektima.
- Pod tim podrazumijevamo da bilo koji objekt iz bilo kojeg imenskog prostora može vidjeti klasu Osoba i iz nje instancirati novi objekt, tj. može ju normalno koristiti.

Konstruktor (1)

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

namespace Konstruktori
{
    public class Osoba
    {
        // Eksplicitno navedeni konstruktor
        public Osoba(string ime, string prezime, int starost,
            string spol, string zanimanje)
        {
            this.Ime = ime;
            this.Prezime = prezime;
            this.Starost = starost;
            this.Spol = spol;
            this.Zanimanje = zanimanje;
        }

        // javno dostupna polja
        public string Ime;
        public string Prezime;
        public int Starost;
        public string Spol;
        public string Zanimanje;

        // privatno polje
        private DateTime VrijemeInstanciranja = DateTime.Now;  // inicijalizator

        // javna metoda
        public void IspisiDetalje()
        {
            Console.WriteLine("Objekt klase Osoba - detalji:");
            Console.WriteLine(
                "Ime: " + Ime + "\n" +
                "Prezime: " + Prezime + "\n" +
```

Konstruktor (2)

Konstruktor se poziva svaki put kad instanciramo novi objekt.

Zadatak konstruktora je stvaranje objekta koji je definiran klasom

Ako klasa ne definira konstruktor, prevoditelj pruža **podrazumijevani (default) konstruktor**

Podrazumijevani konstruktor stvara u memoriji objekt, ali ne izvodi nikakvu drugu akciju.

this (1)

Ključna riječ this pokazuje na *trenutnu instancu objekta*. Možemo reći i ovako: kad se instancira objekt iz klase, this predstavlja taj objekt.

Referenca this je skrivena referenca dostupna svim *nestatičkim* metoda klase (metodama koje nemaju ključnu riječ static u svojoj deklaraciji).

this (2)

```
public void IspisiDetalje()
{
    Console.WriteLine("Objekt klase Osoba - detalji:");
    Console.WriteLine(
        "Ime: " + this.Ime + "\n" +
        "Prezime: " + this.Prezime + "\n" +
        "Starost: " + this.Starost + "\n" +
        "Spol: " + this.Spol + "\n" +
        "Zanimanje: " + this.Zanimanje + "\n" +
        "Vrijeme instanciranja: " + this. VrijemeInstanciranja.ToLongTimeString()
        );
}
```

Uništavanje objekata

- Budući da se objekti neke klase instanciraju na hrpi, nakon upotrebe u programu oni se ne brišu automatski, nego je memoriju koju zauzimaju potrebno osloboditi.
- Programski jezik C# osigurava mehanizam sakupljanja otpada (Garbage Collector), pa objekte ne moramo sami uništavati.

Uništavanje objekata (2)

- Ako objekt kontrolira vrijedne *neupravljive resurse* (poput identifikatora datoteka) koje želimo što prije zatvoriti i odložiti, trebamo implementirati *sučelje* IDisposable i definirati njegovu jedinu metodu Dispose() koja će obavljati sva važna čišćenja.

Statički članovi

- Svi članovi klase (varijable, metode, događaji, indekseri itd.) mogu biti ili **članovi instance** ili **statički članovi**.
- Članovi instance povezani su s instancama klase (objektima), dok su statički članovi dio sâme klase.
- Statičkom članu stoga možemo pristupiti s pomoću naziva klase u kojoj je deklariran.

Metode (1)

- Kako smo već prije spomenuli u 1. poglavlju **metode** su članovi klase koji u pravilu simuliraju ponašanje (engl. *behaviour*).

Metode (2)

```
[modifikator pristupa] tip identifikator ([lista parametara s tipovima])
{
    // tijelo metode – izračun vrijednosti
    [ return vrijednost; ]  // samo ako je tip metode različit od void
}
```

Preopterećivanje metoda i konstruktora

- U kreiranju što logičnijih i prirodnijih klasa često će nam biti potrebno imati dvije ili više metoda s istim imenom.
- Ova potreba često se manifestira i prilikom definiranja konstruktora.

Preopterećivanje metoda i konstruktora

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

namespace PreopterecivanjeKonstruktora
{
    class Osoba
    {
        public Osoba() { }  // defaultni konstruktor
        public Osoba(string ime)
        {
            this.Ime = ime;
        }
        public Osoba(string ime, string prezime)
        {
            this.Ime = ime;
            this.Prezime = prezime;
        }

        // polja
        public string Ime;
        public string Prezime;
    }
    class Program
    {
        static void Main(string[] args)
        {
            Osoba o1 = new Osoba();  // defaultni konstruktor
            Osoba o2 = new Osoba("Ana");
            Osoba o3 = new Osoba("Ivan", "Ivanković");
        }
    }
}
```

Svojstva

- posebno oblikovani članovi klase koji dopuštaju klijentima pristup stanju klase kao da izravno pristupaju poljima članovima, dok zapravo implementiraju pristup kroz metodu klase.
- pružaju jednostavno sučelje jer izgledaju kao varijable članice, a dizajneri klase ih implementiraju kao metode što im omogućuje sakrivanje podataka koje zahtijeva dobar objektno orijentirani dizajn.

Svojstva

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

namespace Svojstva
{
    class Osoba
    {
        // Privatne varijable članice
        private string ime;
        private int starost;

        // Svojstva
        public string Ime
        {
            get { return ime; }
            set { ime = value; }
        }
        public int Starost
        {
            get
            {
             return starost;
            }
            set
            {
             // Prije dodjeljivanja provjeravamo jel nova vrijednost ispravna
             if (value < 0)
             {
                    throw new Exception("Starost ne smije biti manja od nule.");
             }
             else
             {
                    starost = value;
             }
            }
        }
    }
```

Hvala na pažnji!