Иницијализација промењиве типа класе

Класа се може користити као и сваки други тип података: може се декларисти промењива тог типа а затим и иницијализовати промењива са неком вредности:

```
using System;
namespace Proba
{
    class Program
    {
        static void Main()
        {
            Krug k;
            k = new Krug();
        }
    }
    class Krug {}
}
```

Раније, приликом декларације и иницијализације промењиве:

```
int i;
i = 42;
```

Сада се то не може радити на исти начин са промењивима типа неке класе, нпр:

```
Krug k;
k = 42;
```

Разлози су у ограничењима синтаксе доделе литералних вредности класе промењивима као и алокацији меморије за промењиве типа класе.

У процесу иницијализације се користи службена реч `new` са којом се креира нова инстанца класе, коју називамо објекат.

Може се директно доделити инстанца класе другој промењивој истог типа:

```
Krug k;
k = new Krug();
Krug d;
d = k;
```

Дефинисање објекта

Класа је шема, темплејт како ће изгледати објекат а објекат је инстанца класе.

Класе су попут калупа за виџете (windows gadgets) док објекти одговарају виџетима креираним према калупу.

Процес креирања објекта из класе се назива инстанцијација пошто је објекат инстанца класе.

Пример:

```
using System;
namespace Proba
{
    class Program
    {
        static void Main()
        {
            Ljudi covek1 = new Ljudi();
            Ljudi covek2;
            covek2 = new Ljudi();
            Trcanje(covek1);
            Trcanje(covek2);
        }
        static void Trcanje(Ljudi x) {}
    }
    class Ljudi {}
}
```

У примеру оператор new даје инструкцију runtime којом се алоцира меморија за Ljudi објекат, иницијализује објекат и враћа референца на инстанцу.
Иако постоји експлицитни оператор за процес алокације меморије, не постоји оператор за деалокацију меморије.
Уместо тога, runtime аутоматски враћа меморију после поступка којим се објекат чини недоступним.
Колектор ђубрета је одговоран за аутоматску деалокацију; одлучује који објекти нису више референцирани са другим активним објектима а затим деалоцира меморију за те објекте.
Резултат је такав да не постоји програм којим је предефинисано време компајлирања за које ће меморија постати доступна систему.

Поља

Поље је промењива која је члан класе:

```
class Macke
{
     string ime;
     public int godine = 10;
}
```

У примеру су промењиве ime и godine поља класе.
Поља могу користити модификаторе: static, public, internal, private, protected, new, unsafe, readonly, volatile.
Модификатор readonly спречава да се поље модификује после конструкције.
Поље које је readonly се може доделити само приликом своје декларације или унутар типа конструктора.
Поље не мора да се иницијализује пошто неиницијализовано поље има дефолтну вредност (0, '\0', null, false) али ако постоји мора да се изврши пре конструктора.

Иницијализација поља може да садржи изразе или позиве метода.
Могуће је иницијализовати више поља истовремено: static readonly int a = 2, b = 3;

Поља омогућавају да се повежу неки подаци са сваком од будућих инстанци класе.
Ако се користи модификатор public он указује да податак унутар поља је доступан и из других класа.

Приступ пољу инстанце
Подаци у пољу се могу поставити и добити.
Ако поље нема static модификатор, то указује да је то поље инстанце.
Пољу инстанце се може приступити само преко инстанце класе, а не може директно из класе.

```
#nullable enable
using System;
namespace Proba
{
    class Program
    {
        static void Main()
        {
            Zaposleni radnik1 = new Zaposleni();
            Zaposleni radnik2 = new Zaposleni();
            Console.WriteLine($"{radnik1.Ime} {radnik1.Prezime}:
                    {radnik1.Plata}");
           Console.WriteLine($"{radnik2.Ime} {radnik2.Prezime}:
                    {radnik2.Plata}");
        }
    }
    public class Zaposleni
    {
        public string Ime = "Miki";
        public string Prezime = "Mikic";
        public string Plata = "Mala";
    }
}
```

Да би се користио C#8.0 у коду, потребно је убацити у .csproj линију: <LangVersion>8.0</LangVersion>, а у скрипту на врху #nullable enable.
Даје:
Miki Mikic: Mala
Miki Mikic: Mala

У примеру су инстанцирана два објекта (radnik1 и radnik2) али оба објекта добијају иницијализоване вредности из саме класе за сва три поља (Ime, Prezime, Plata).
Очигледно је да постоји могућност доделе почетне вредности за све инстанциране објекте на овај начин али је ту мала практична вредност.

Пример:

```
#nullable enable
using System;
namespace Proba
{
    class Program
    {
        static void Main()
        {
            Zaposleni radnik1 = new Zaposleni();
            Zaposleni radnik2 = new Zaposleni();
            radnik1.Ime = "Miki";
            radnik1.Prezime = "Mikic";
            radnik2.Ime = "Ana";
            radnik2.Prezime = "Anic";
            Console.WriteLine($"{radnik1.Ime} {radnik1.Prezime}:
                    {radnik1.Plata}");
            Console.WriteLine($"{radnik2.Ime} {radnik2.Prezime}:
                    {radnik2.Plata}");
        }
    }
    public class Zaposleni
    {
        public string Ime;
        public string Prezime;
        public string Plata = "Mala";
    }
}
```

Даје:

Miki Mikic: Mala

Ana Anic: Mala

Сада сваки објекат има по два поља која се иницијализују изван класе која их инстанцира, што је практичније и реалније (омогућено јер је модификатор за сва поља public).

<u>Задаци за самосталан рад</u>

1. Креирати класу Zivotinje и инстанцирати објекат macka.

2. Креирати класу Zivotinje и и у класи декларисати три поља: brojNogu, kretanjeOpis, boja.

3. Креирати класу Vozila. У класи декларисати и иницијализовати поља bojaVozila, kretanjeOpis, brojSedista. Инстанцирати класу објектима motor, auto и avion. Доказати непрактичност иницијализације унутар класе.

4. Креирати класу Zivotinje и и у класи декларисати три поља: brojNogu, kretanjeOpis, boja. Инстанцирати објекте muva, riba и ovca и иницијализовати поља за сваку од инстанци.