Дефиниције чланова класе

Унутар дефиниције класе, даје се и дефиниција свих чланова класе укључујући поља, методе и својства (properties).

Сваки члан класе има дефинисан свој ниво приступа преко неке од службених речи:

- public – чланови су доступни из било којег кода
- private – чланови су доступни само из кода који је члан класе (сматра се дифолтним приступом ако се не користи никаква службена реч)
- internal – чланови су доступни само из кода унутар пројекта (assembly) где су дефинисани
- protected – чланови су доступни само из кода који је део или класе или настао из класе

Две последње се могу комбиновати (нпр protected internal па је тада члан доступан из кода унутар пројекта или из кода насталог из класе унутар другог пројекта).

Дефинисање поља користи стандардне формате декларације као и претходно описане модификаторе (public int mojInt;)

Поља могу користити и службену реч readonly, што значи да се пољу може доделити вредност само током извршења конструктора или иницијалним исказом: public readonly int mojInt = 17;.

Поља се могу декларисати и као static: public static int mojInt; статичким пољима се приступа преко класе која их дефинише (нпр MojaKlasa.mojInt) а не преко инстанци класе.

Може се користити const за креирање константних вредности, константни чланови су static према дефиницији.

Дефинисање метода

Ако се користи static, онда је метода доступна само кроз класу а не кроз објекат инстанце.

Могу се користити следеће службене речи:

- virtual – метод се може overridden
- abstract – допуштено смао код апстрактних класа
- override – метод overrides методу основне класе (мора да се користи ако се метод override)
- extern – дефиниција методе се налази негде другде

Дефинисање својстава

Својства (properties) имају два посебна блока функције: први за добијање вредности особености и други за постављање вредности особености.

Оби блокови (називају се још и приступници, accessors) су дефинисани коришћењем службених речи get и set и могу се користити за контролисање нивоа приступа особеностима.

Ако се не користи get блок, добија се само приступ уписивања, а ако се не користи set блок, добија се само приступ читања; и то се односи само на екстерне кодове пошто код унутар класе има право приступа истим подацима као ови блокови; најчешће се један од ова два блока користи као public.

Основна структура својстава се састоји од стандардне службене речи модификације приступа (public, private…) а затим назива типа, имена својства и једног или оба блока у којима се врши обрада својства:

```
public int MojaIntSvojstva
{
	get
	{
		//get blok koda svojstva
	}
	set
	{
		//set blok koda svojstva
	}
}
```

Блок get мора имати вредност која се враћа, истог типа као и својство.
Једноставнија својства се често повезују са једним private пољем (backing field) којима се контролише приступ том пољу, у којем случају блок get може вратити вредност поља директно:

```
private int mojInt; //polje koje svojstvo koristi
public int MojaIntSvojstva //svojstvo
{
	get { return mojInt; }
	set { /*set blok koda svojstva*/ }
}
```

Пошто екстерни код не може директно приступити пољу mojInt (приватно), може користити својство.
Функција set додељује вредност пољу; у примеру се користи vrednost која се односи на вредност добијену од корисника својства: `set { mojInt = vrednost; }`
Сада vrednost одговара вредности истог типа каошто је својство, па ако својство користи исти тип као и поље, неће бити проблема са кастовањем.
Да би се обезбедило допуштења рада са null, може се користити:

```
private int? mojInt;
public int? MojaIntSvojstva
{
	get => mojInt;
	set => mojInt = value ?? 0;
}
```

Проста својства не раде веће ствари од обезбеђења директног приступа пољу mojInt.
Већа снага лежи у већој контроли у обради:

```
set
{
	if (value >= 0 && value <= 10)
		mojInt = value;
}
```

Пример: Написати програм који коришћењем својства са гетером и сетером поставља и користи име корисника као приватну промењиву класе

```
using System;
namespace Proba
{
    class Osoba
    {
        private string ime = "Ana"; //polje
        public string Ime           //svojstvo
        {
            get
            {
                Console.WriteLine("B");
                return ime;
            }
            set
            {
                ime = value;
                Console.WriteLine("A");
            }
        }
    }
    class Program
    {
        static void Main()
        {
            Osoba covek = new Osoba();
            covek.Ime = "Miki";     //startuje se set aksesor
            Console.WriteLine(covek.Ime); //startuje se get aksesor
        }
    }
}
```

Даје:

A

B

Miki

У примеру својство Ime је придружено са пољем ime (често се дају иста имена за својство и поље само је својство са великим словом).

Метода set додељује value промењивој ime; службена реч value представља вредност коју додељујемо својству.

Метода get враћа вредност промењиве ime.

Такође је могуће написати аутоматизовано својство (automatic properties), где се не мора дефинисати поље својства већ само get; и set; унутар својства.

У претходном примеру без употребе приватне промењиве класе:

```
using System;
namespace Proba
{
    class Osoba
    {
        public string Ime
        { get; set; }
    }
    class Program
    {
        static void Main()
        {
            Osoba covek = new Osoba();
            covek.Ime = "Miki";
            Console.WriteLine(covek.Ime);
        }}
}
```

Заправо, део кода у својству је идентичан са:

```
class Osoba
{
     private string ime;
     public string Ime
     {
          get
          {
              return this.ime;
          }
          set
          {
              this.ime = value;
          }
     }
}
```

<u>Задаци за самосталан рад</u>

1. Креирати класу Broj и класу Програм. Класа Broj има својства са методама гетера и сетера и приватну промењиву posebanBroj = 100. Приказати како се може прићи приватном пољу у класи ако корисник покуша да приђе промењивој са модификованом вредности posebanBroj = 1, коришћењем својства.
2. Модификовати претходни пример којим се приказује како се применом сетера штити садржај приватне промењиве унутар класе Broj.
3. Написати програм који помоћу својства допушта промену вредности стринга приватне промењиве класе, али је представља у великим словима.