



ASICODE

Курс Основи - C#

ЗАНЯТТЯ № 10 - ENUM, READONLY, STATIC



Що таке ENUM

Enum – користувальницький тип даних, який являє собою набір іменованих цілих констант. По суті це список однотипних значень: днів тижня, кодів відповіді сервера, арифметичних дій.

C# має вбудований список кольорів. Це чудовий приклад enum. Користувачеві не потрібно вводити код кольору щоразу. Він просто вибирає значення підготовленого заздалегідь списку.



оголошення enum

Для оголошення цього використовується оператор enum. Після оператора enum вказується ім'я типу, що створюється. Через двокрапку можна вказати тип констант перерахування (має бути ціличисленним). Якщо не вказувати тип явно, то буде використано int. Після цього наводиться список символічних констант через кому. Приклад типу переліку:

```
enum TypeOS
{
    Windows, Linux, MacOS, Android
}
```



CHECKLIST





ASICODE

Особливості оголошення



Кожному елементу перерахування надається ціле значення, у прикладі вище, ці значення починаються з нуля і, далі, збільшуються на одиницю. Можна явно встановити значення елементам перерахування. Для цього C# надає два варіанти: перший - це завдання чисельного значення першому елементу, при цьому всі інші, по черзі, набирають значення на одиницю більше:

```
enum TypeOS
{
    Windows = 1 ,
    Linux,
    MacOS,
    Android
}
```

Другий варіант передбачає явне завдання чисельних значень кожному елементу перерахування індивідуально:

```
enum TypeOS
{
    Windows = 1 ,
    Linux = 3
    MacOS = 5
    Android = 7
}
```



Робота зі змінними типу перерахування

З елементами типу перерахування можна проводити різні операції - це приведення значення до цілісного типу, операції порівняння (`==`, `!=`, `<`, `>` і т.п.), арифметичні операції: додавання та віднімання, логічні операції (`&`, `|`, `^`), операція побітового доповнення (`~`) та операції постфіксного та префіксного інкременту (`++`) та декременту (`-`). Enum також може бути результатом вираження в операторі `switch`.





модифікатор **readonly**

Поля читання представляють такі поля класу чи структури, значення яких не можна змінити. Таким полям можна надати значення або при безпосередньо при їх оголошенні, або в конструкторі. В інших місцях програми надавати значення таким полям не можна, можна лише зчитувати їх значення.

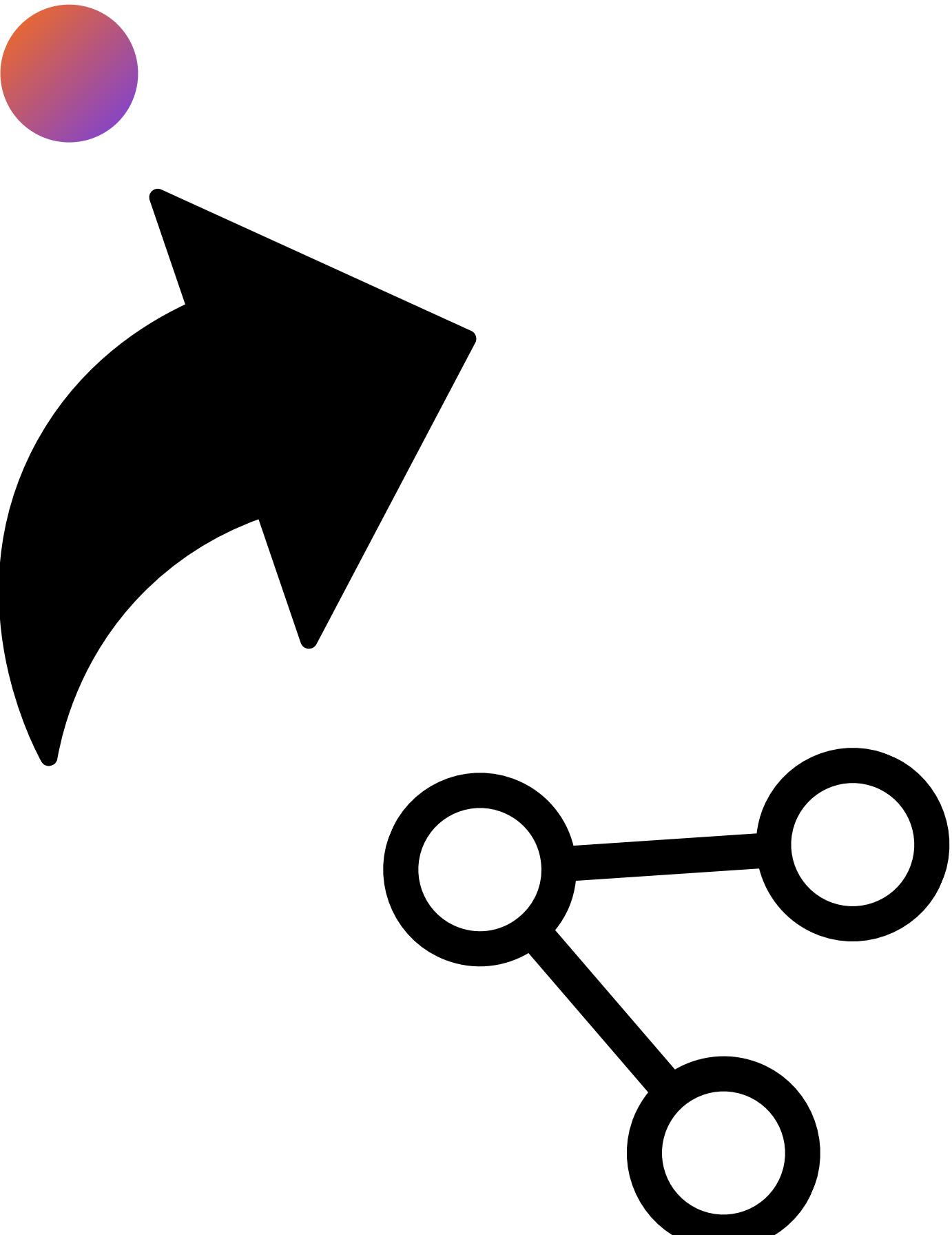
Поле для читання оголошується із ключовим словом **readonly**:

```
public readonly string name = "Undefined"; // можно так
инициализировать
public Person(string name)
{
    this.name = name; // в конструкторе также можно
присвоить значение полю для чтения
}
```



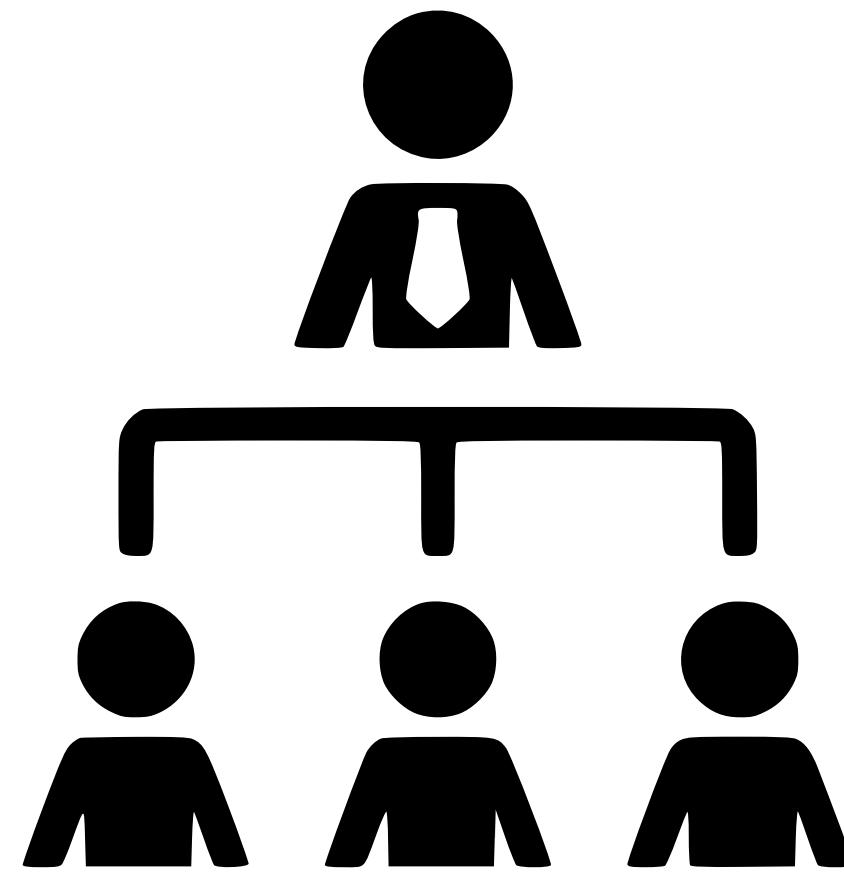
модификатор **static**

Крім звичайних полів, методів, властивостей класи та структури можуть мати статичні поля, методи, властивості. Вони відносяться до всього класу/усієї структури та для звернення до подібних членів необов'язково створювати екземпляр класу/структурі.





Статичні поля



Статичні поля зберігають стан всього класу/структурі. Статичне поле визначається як і просте, лише перед типом поля вказується ключове слово `static`. Наприклад, розглянемо клас `Person`, який представляє людину:

```
class Person
{
    int age;
    public static int retirementAge = 65;
    public Person(int age)
    {
        this.age = age;
    }
}
```

У цьому випадку клас `Person` має два поля: `age`(зберігає вік людини) та `retirementAge`(зберігає пенсійний вік). Однак поле `retirementAge` є статичним. Таким чином, поле `retirementAge` відноситься не до окремого об'єкта і зберігає значення НЕ окремого об'єкта класу `Person`, а відноситься до всього класу `Person` і зберігає загальне значення для класу.



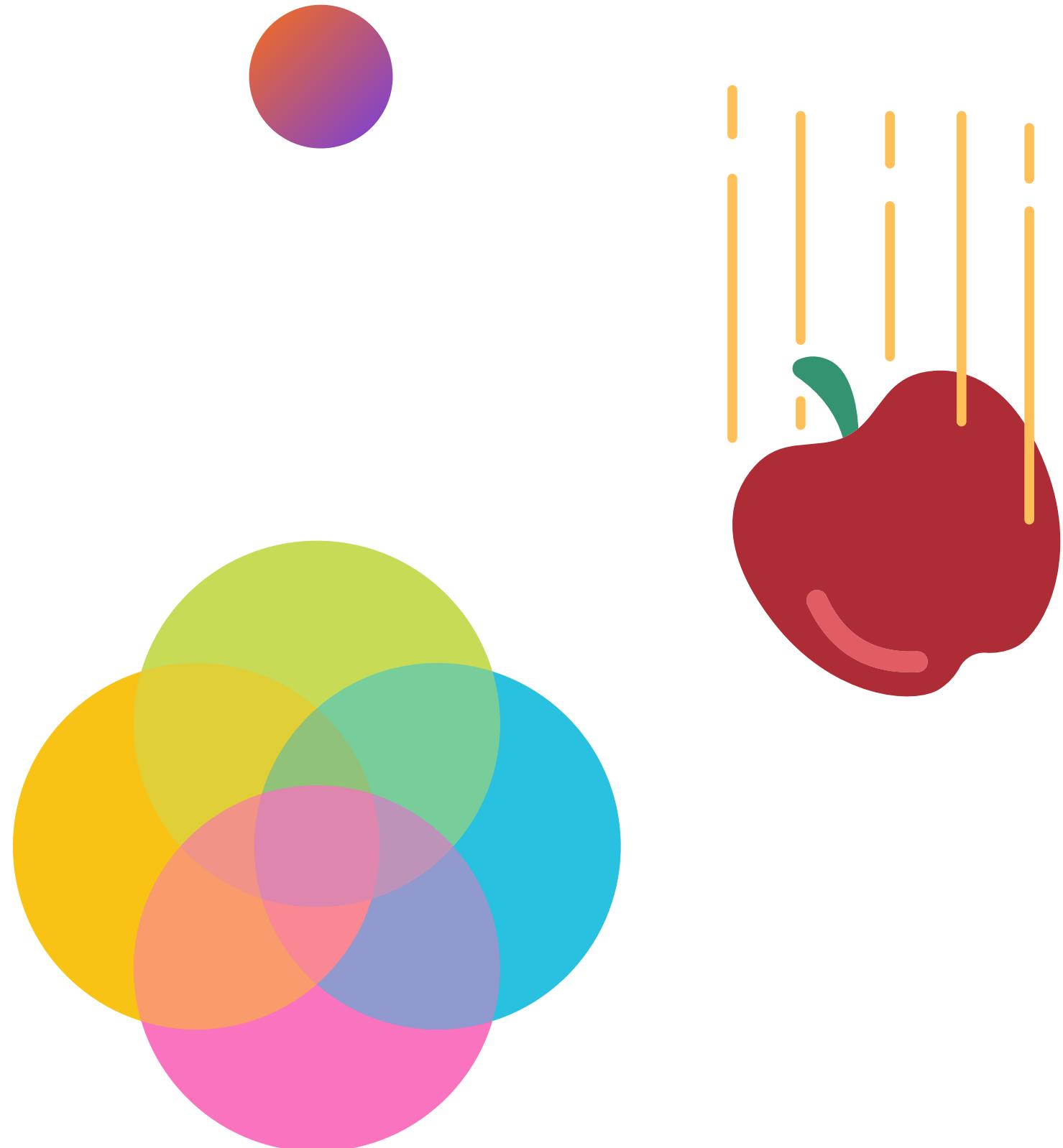
Статичні властивості

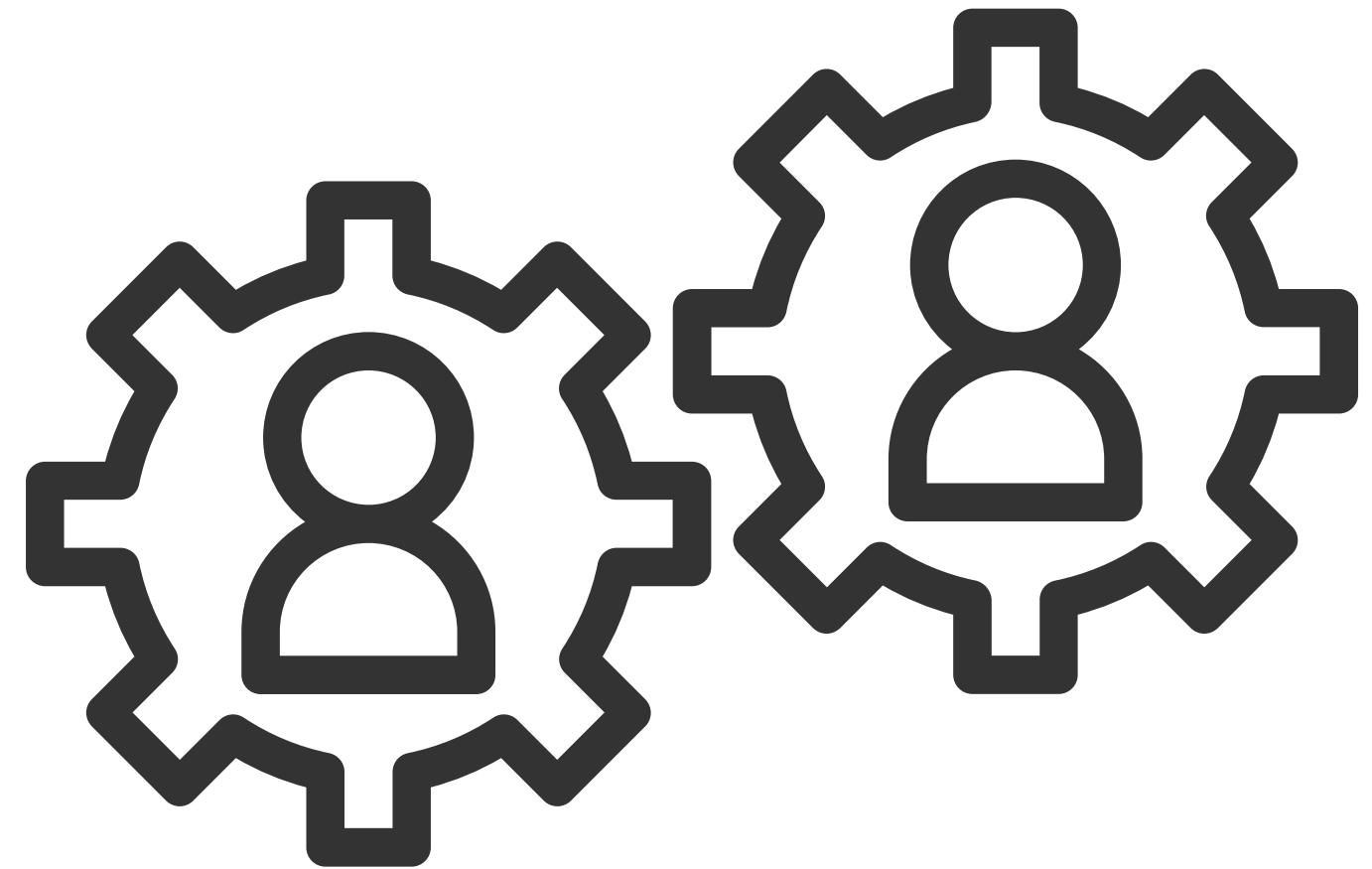
Подібним чином ми можемо створювати та використовувати статичні властивості:

```
static int retirementAge = 65;  
public static int RetirementAge  
{  
    get { return retirementAge; }  
    set { if (value > 1 && value < 100) retirementAge =  
value; }  
}
```

В даному випадку доступ до статичної змінної відновлення оподаткування опосередковується за допомогою статичної властивості RetirementAge.

Таким чином, змінні та властивості, які зберігають стан, загальний для всіх об'єктів класу/структурі, слід визначати як статичні.





Статичні методи

Статичні методи визначають загальне всім об'єктів поведінка, яке залежить від конкретного об'єкта. Для звернення до статичних методів також застосовується ім'я класу/структурі:

```
class Person
{
    public int Age { get; set; }
    static int retirementAge = 65;
    public Person(int age) => Age = age;
    public static void CheckRetirementStatus(Person person)
    {
        if (person.Age >= retirementAge)
            Console.WriteLine("Уже на пенсии");
        else
            Console.WriteLine($"Сколько лет осталось до пенсии:
{retirementAge - person.Age}");
    }
}
```

Слід враховувати, що статичні методи можуть звертатися лише до статичних членів класу. Звертатися до нестатичних методів, полів, властивостей усередині статичного методу ми не можемо.

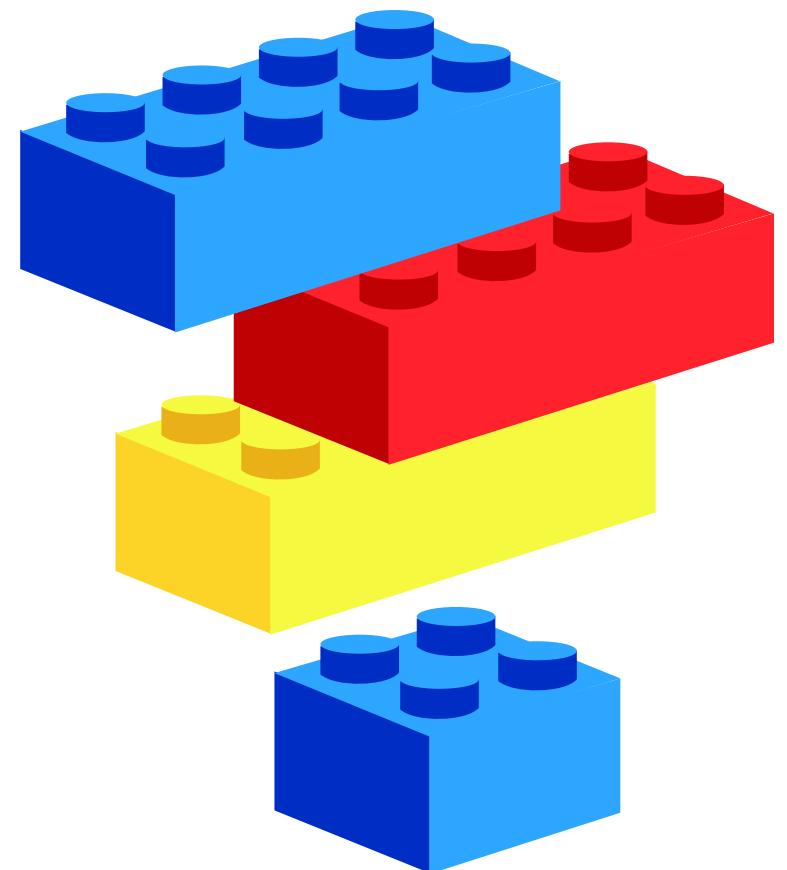


Статичний конструктор

Крім звичайних конструкторів, у класу також можуть бути статичні конструктори. Статичні конструктори мають такі відмінні риси:

- Статичні конструктори не повинні мати модифікатор доступу та не приймають параметрів
- Як і в статичних методах, в статичних конструкторах не можна використовувати ключове слово для посилання на поточний об'єкт класу і можна звертатися тільки до статичних членів класу
- Статичні конструктори не можна викликати у програмі вручну. Вони виконуються автоматично при першому створенні об'єкта даного класу або при першому зверненні до його статичних членів (якщо такі є)

Статичні конструктори зазвичай використовуються для ініціалізації статичних даних, або виконують дії, які потрібно виконати лише один раз





ASICODE

Дякую за увагу