📘 Դաս 6. Կոնստրուկտորներ, Lifecycle, Static/Instance, Access Modifiers

🎯 Նպատակ

Ուսանողը դասից հետո պետք է կարողանա.

- օգտագործել **տարբեր տեսակի կոնստրուկտորներ**,
- հասկանալ օբյեկտի **կյանքի ցիկլը (սարքվել, օգտագործվել, քանդվել)**,
- տարբերել **static vs instance** անդամները,
- կիրառել տարբեր **access modifiers**,
- իմանալ destructor-ի գաղափարը և IDisposable pattern-ի հիմունքները:

📖 Տեսական մաս

1) Կոնստրուկտորների տեսակներ

Լռելյայն կոնստրուկտոր

```
public class User
{
    public string Name { get; set; }
    public User() // default ctor
    {
        Name = "Unknown";
    }
}
```

Պարամետրացված կոնստրուկտոր

```
public User(string name)
{
   Name = string.IsNullOrWhiteSpace(name) ? "Unknown" : name.Trim();
}
```

Constructor Chaining (շղթայավորում)

```
public class Car
{
   public string Model { get; }
   public int Year { get; }

   public Car(string model) : this(model, DateTime.Now.Year) { }
   public Car(string model, int year)
   {
      Model = model;
      Year = year;
   }
}
```

Static constructor

- Կանչվում է **մի անգամ**՝ երբ առաջին անգամ օգտագործվում է type-ը:
- Սովորաբար՝ static fields-ի initialization-ի համար:

```
public class Logger
{
   public static readonly DateTime AppStart;
   static Logger() // static ctor
   {
      AppStart = DateTime.UtcNow;
      Console.WriteLine("Logger initialized");
   }
}
```

---

2) Instance vs Static (խորությամբ)

- **Instance member** → պատկանում է կոնկրետ օբյեկտին:
- **Static member** → պատկանում է տիպին (class-ին):

Օրինակ.

```
public class Counter
{
    public int InstanceCount;          // յուրաքանչյուր օբյեկտի համար առանձին
    public static int TotalCount;      // ընդհանուր բոլոր օբյեկտների համար

    public Counter()
    {
        InstanceCount++;
        TotalCount++;
    }
}

var c1 = new Counter();
var c2 = new Counter();

Console.WriteLine(c1.InstanceCount); // 1
Console.WriteLine(c2.InstanceCount); // 1
Console.WriteLine(Counter.TotalCount); // 2
```

---

3) Access Modifiers (մուտքի սահմանափակումներ)

- `public` → հասանելի է ամենուր:
- `private` → հասանելի է միայն տվյալ class-ում:
- `protected` → հասանելի է class-ում և նրա ժառանգներում:
- `internal` → հասանելի է նույն assembly-ի ներսում:
- `protected internal` → հասանելի է ժառանգների համար և նույն assembly-ի ներսում:
- `private protected` → միայն ժառանգներին, որոնք նույն assembly-ում են:

Օրինակ.

```
public class Person
{
    private string _secret = "Hidden";   // միայն class-ի ներսում
    public string Name { get; set; }     // հասանելի է ամենուր
    protected int Age { get; set; }      // հասանելի է ժառանգ class-ում
}
```

---

4) Encapsulation խորացում

Encapsulation = տվյալների պաշտպանություն + բացահայտել միայն անհրաժեշտը:

Օրինակ.

```
public class BankAccount
{
   private decimal _balance;

   public decimal Balance => _balance; // read-only property

   public void Deposit(decimal amount)
   {
      if (amount <= 0) throw new ArgumentException("Amount must be > 0");
      _balance += amount;
   }
}
```

➡ Այստեղ balance-ը դրսից չի փոփոխվում, միայն method-ներով:

---

5) Destructor (Finalizer)

- Օգտագործվում է unmanaged resources մաքրելու համար:
- C#-ում շատ հազվադեպ է պետք, ավելի լավ է օգտագործել IDisposable:

```
public class Demo
{
   ~Demo() // destructor
   {
      Console.WriteLine("Destructor called");
   }
}
```

---

6) IDisposable pattern (ծանոթության համար)

```
public class FileManager : IDisposable
{
   private StreamReader? _reader;
```

```
    public FileManager(string path)
    {
        _reader = new StreamReader(path);
    }

    public void Dispose()
    {
        _reader?.Dispose();
        Console.WriteLine("File closed");
    }
}
```

Օգտագործում.

```
using (var fm = new FileManager("data.txt"))
{
    // Work with file
} // Dispose ինքն իրեն կանչվում է
```

💻 Լաբորատոր աշխատանք

Լաբ 1. Counter + Static ctor

Գրիր class `Counter`, որը.

- Ունի static field `TotalCreated`,
- Instance field `Id`, որը ավտոմատ աճում է,
- Console demo՝ ստեղծիր 3 օբյեկտ և տպիր յուրաքանչյուրի Id և ընդհանուր քանակը:

Լաբ 2. LibraryBook (Constructor chaining)

Գրիր class `LibraryBook`, որը.

- Ունի Title, Author, Year,
- Ունի 3 կոնստրուկտոր (default, 2-param, 3-param),

- Constructor chaining կիրառիր,
- Console demo՝ ստեղծիր տարբեր տարբերակներով և տպիր:

Լաբ 3. BankAccount Advanced

- Deposit / Withdraw, balance read-only,
- Static property → `TotalAccounts`,
- Console demo՝ ստեղծիր մի քանի հաշիվ և կատարիր գործողություններ:

🏠 Տնային առաջադրանք

1. Գրիր class `Student`, որը.
    - Ունի Name, Age, StudentId,
    - Static field → Count (քանի ուսանող է ստեղծվել),
    - Constructor chaining տարբերակներով,
    - Console demo՝ ստեղծիր 3 ուսանող:
2. Գրիր class `Car`, որը.
    - Ունի Model, Year, Speed,
    - Methods → Accelerate(), Brake(),
    - Year < 1900 হলে throw,
    - Console demo՝ ստեղծիր մի քանի մեքենա և փորձարկիր:
3. Գրիր class `Logger`, որը.
    - Static constructor → նշի start time,
    - Method → Log(string msg), որը տպում է ժամանակով:

4. (Բոնուս) Գրիր class `FileWrapper`, որը բացում է ֆայլը constructor-ում և փակվում Dispose-ում: Console demo՝ գրի մեջբերում ֆայլից:

🧩 Քննարկման հարցեր

- Ինչ տարբերություն կա instance և static constructor-ների միջև:
- Ինչու destructor-ը հազվադեպ է պետք:
- Ինչ տարբերություն կա `public` և `internal` modifier-ների միջև:
- Ինչպե՞ս է encapsulation-ը օգնում ծրագրի կայունությանը:

📎 Ամփոփում

Այս դասից հետո ուսանողները.

- Կիմանան **կոնստրուկտորների բոլոր ձևերը**,
- Կկարողանան աշխատել **static vs instance** տարանջատմամբ,
- Կիմանան access modifiers,
- Կունենան առաջին պատկերացումը **IDisposable pattern**-ից: