

Դաս.7.2 OOP-ի և C# լեզվի ընդլայնված հնարավորություններ

(Զիմք՝ Andrew Troelsen, Pro C# 10 with .NET 6)

1. Understanding Partial Classes

Ի՞նչ է.

- `partial` keyword-ով class-ը կարող է բաժանվել մի քանի ֆայլերի:
- Compiler-ը compile time-ին միավորում է բոլոր մասերը:

Օգտակար է.

- Designer-generated code + user code separation (WinForms/WPF, EF Core scaffolding):
- Մեծ class-երի բաժանում logical մասերի:
- Source generators → ավտոմատ լրացված մասեր:

Օրինակ.

```
// File: Person.Part1.cs
public partial class Person
{
    public string Name { get; set; } = "";
}
```

```
// File: Person.Part2.cs
public partial class Person
{
    public int Age { get; set; }
}
```

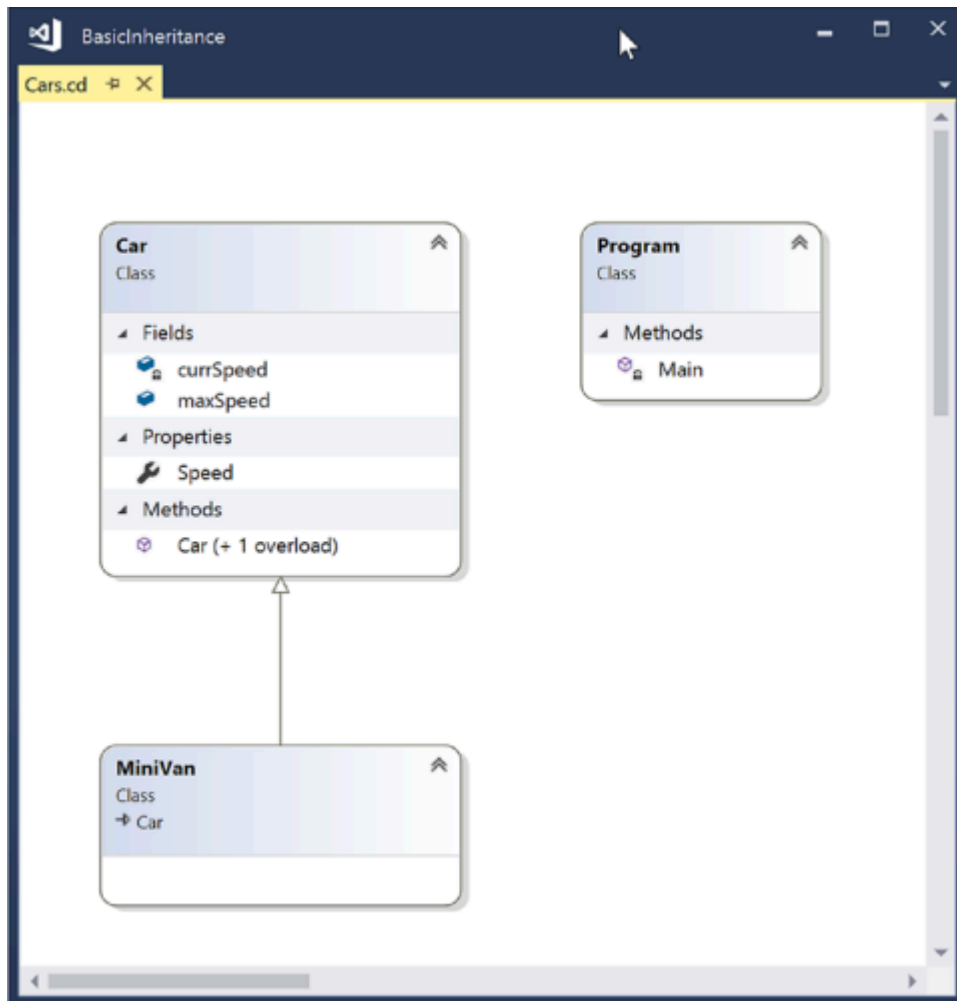
2. Use Cases for Partial Classes

- **WPF/WinForms** → Visual Studio designer-ի կողքը պահվում է առանձին:
 - **EF Core Models** → Scaffold code մեկ ֆայլում, custom logic – մյուսում:
 - **Large teams** → տարբեր developers աշխատում են class-ի տարբեր մասերի վրա:
 - **Source Generators** → ավտոմատ methods ավելացնել compile time-ին:
-

3. Revisiting Visual Studio Class Diagrams


- VS Class Diagram (.cd file) → UML նմանացված visualization:
- Ցույց է տալիս class hierarchy, inheritance, associations:

- Կարող է օգտագործվել reverse engineering-ի համար:



Օրինակ Workflow.

1. Ավելացրու class diagram project-ում:
2. Drag-drop class-եր:
3. Կտեսնես inheritance arrows, properties, methods:

 (Այստեղ projector-ով ցույց տալու տեղը կլինեն քո նկարից):

4. Programming for Containment/Delegation

- **Containment (HAS-A)** → class A-ն իր մեջ ունի class B instance:
- **Delegation** → class A-ն method-ը փոխանցում է contained class-ին:

```

public class Engine
{
    public void Start() => Console.WriteLine("Engine started");
}

public class Car
{
    private Engine _engine = new Engine();

    public void Drive()
    {
        _engine.Start(); // Delegation
        Console.WriteLine("Car is driving");
    }
}

```

➔ Սա այլընտրանք է inheritance-ին (IS-A vs HAS-A):

5. Understanding Nested Type Definitions

- Nested types = type-ի ներսում type:
- Օգտագործվում է.
 - **Helper class** (visible only inside parent):
 - **Enums inside class** → logically grouped:
 - **Encapsulation** → ոչ public detail:

```

public class Outer
{
    public class Nested
    {
        public void Print() => Console.WriteLine("Nested class");
    }
}

var n = new Outer.Nested();
n.Print();

```

6. Controlling Base Class Creation with the **base** Keyword

- **base** → թույլ է տալիս derived class-ից կանչել base constructor-ը կամ method-ը:

```
public class Person
{
    public Person(string name) => Console.WriteLine($"Person: {name}");
}
```

```
public class Student : Person
{
    public Student(string name) : base(name) // base ctor call
    {
        Console.WriteLine("Student created");
    }
}
```

- Կարևոր է հասկապես, երբ base class-ը չունի default constructor:

7. Sealing Virtual Members

- **sealed override** → derived class-ում override արած method-ը այլևս չի կարելի override անել հաջորդ ժառանգներում:

```
public class Animal
{
    public virtual void Speak() => Console.WriteLine("Sound");
}

public class Dog : Animal
{
    public sealed override void Speak() => Console.WriteLine("Woof!");
}
```

➡ Design decision → կանխում ես անիմաստ կամ վտանգավոր վերաիմպլեմենտացիան:

8. Understanding Member Shadowing

- Երբ child class-ը հայտարարում է նույն անունով անդամ, ինչ parent-ում:

- **new** keyword-ը հստակ ցույց է տալիս shadowing:

```
public class Base
{
    public void Show() => Console.WriteLine("Base");
}
```

```
public class Derived : Base
{
    public new void Show() => Console.WriteLine("Derived");
}
```

```
Base b = new Derived();
b.Show(); // Base (not virtual!)
```

→ Օգտագործիր միայն անհրաժեշտության դեպքում: Ավելի լավ է virtual/override:

9. Base Class / Derived Class Casting Rules

- **Upcasting (Derived → Base)** → միշտ ապահով:
- **Downcasting (Base → Derived)** → պետք է ստուգել:

```
Animal a = new Dog(); // Upcast
Dog d = (Dog)a;       // OK
```

```
Animal b = new Animal();
Dog d2 = (Dog)b;       // ❌ InvalidCastException
```

10. The **as** Keyword

- Փորձում է cast անել, եթե չի ստացվում → վերադարձնում է null:
- Ավելի անվտանգ, քան hard cast:

```
object obj = "hello";
```

```
string s = obj as string; // "hello"
Dog d = obj as Dog;      // null
```

11. The **is** Keyword (Updated)

- Հին տարբերակ. **if** (**obj is** Dog)
- Նոր տարբերակ (pattern matching).

```
if (obj is Dog dog)
    dog.Bark();
```

- Աջակցում է logical patterns (**is not**, **and**, **or**):

```
if (obj is string s and { Length: > 3 })
    Console.WriteLine($"Long string: {s}");
```

12. Pattern Matching Revisited (New)

- Switch expressions, relational patterns, logical patterns:

```
object shape = new Circle { Radius = 5 };
```

```
string desc = shape switch
{
    Circle { Radius: > 10 } c => $"Big circle: {c.Radius}",
    Circle c                  => $"Circle: {c.Radius}",
    Rectangle r               => $"Rect: {r.Width}x{r.Height}",
    _                          => "Unknown shape"
};
```

➔ Շատ ավելի ընթերցանելի և declarative code:



Լաբորատոր աշխատանքներ

1. **Partial Class Demo** → Բաժանիկ class-ը երկու ֆայլի, ավելացրու properties տարբեր մասերում:

2. **Containment vs Inheritance** → Ստեղծիր Engine/Car և Animal/Dog hierarchy, համեմատիր:
 3. **Nested Class Demo** → Ստեղծիր Outer/Nested օրինակ:
 4. **Casting + as/is Demo** → Ստուգիր upcast/downcast տարբերությունները:
 5. **Pattern Matching Demo** → Shape hierarchy (Circle, Rectangle) switch expression-ով:
-

Քննարկման հարցեր

- Partial classes-ը ինչպե՞ս օգտվում են WinForms/WPF ծրագրերում:
 - Ինչ տարբերություն կա override vs new member shadowing միջև:
 - Ինչու sealed override-ն երբեմն անհրաժեշտ է:
 - Ինչու downcast-ը վտանգավոր է և ինչպես **is/as** keyword-երը ապահովում են անվտանգություն:
 - Ինչ առավելություն է տալիս pattern matching-ը դասական switch-ի նկատմամբ:
-

Ամփոփում

Այս դասից հետո ուսանողները.

- Կիմանան **partial classes**-ի դերը,
- Կկարողանան տարբերել **containment vs inheritance**,
- Կիմանան **nested types**, **base keyword**, **sealed override**, **shadowing** կիրառումը,
- Կկարողանան անվտանգ օգտագործել **casting**, **as/is**,
- Տիրապետեն **pattern matching** նոր հնարավորություններին: