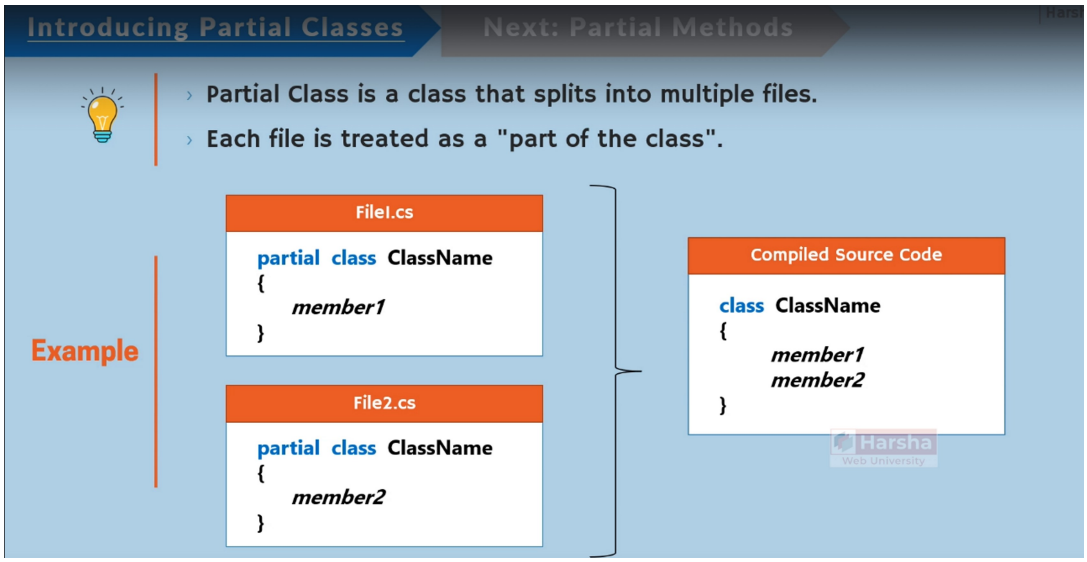
Sec 13: Partial & Static Classes, Enumerations

14 July 2024 11:29


The Goal of Partial Classes is that, allowing the first developer to create one part of the class and allowing the second developer to develop the second part of the class.

## Understanding Partial Classes

Next: Partial Methods

- At compilation time, all partial classes that have same name, become as a "single class".
- All the partial classes (that want to be a part of a class) should have same name and should be in the same namespace and same assembly & should have same access-modifier (such as 'internal' or 'public').
- Duplicate members are not allowed in partial classes.
- Any attributes / modifiers (such as abstract, sealed) applied on one partial class, will be applied to all partial classes that have same name.
- The 'partial' keyword can be used only at before the keywords 'class', 'struct', 'interface' and 'void'.



## Understanding Partial Classes

Next: Partial Methods

**Advantage**

- Each partial class can be developed individually, by different developers / teams.
- In WinForms / WebForms, the 'Designer-generated code' will be kept in one partial class; the 'code written by developer' will be kept in another partial class with same name; so both become as a single class at compilation time.

## Introducing Partial Methods

**What**

- Partial Methods are "declared in one partial class" (just like abstract method), and "implemented in another partial class", that have same name.

**First Partial Class**

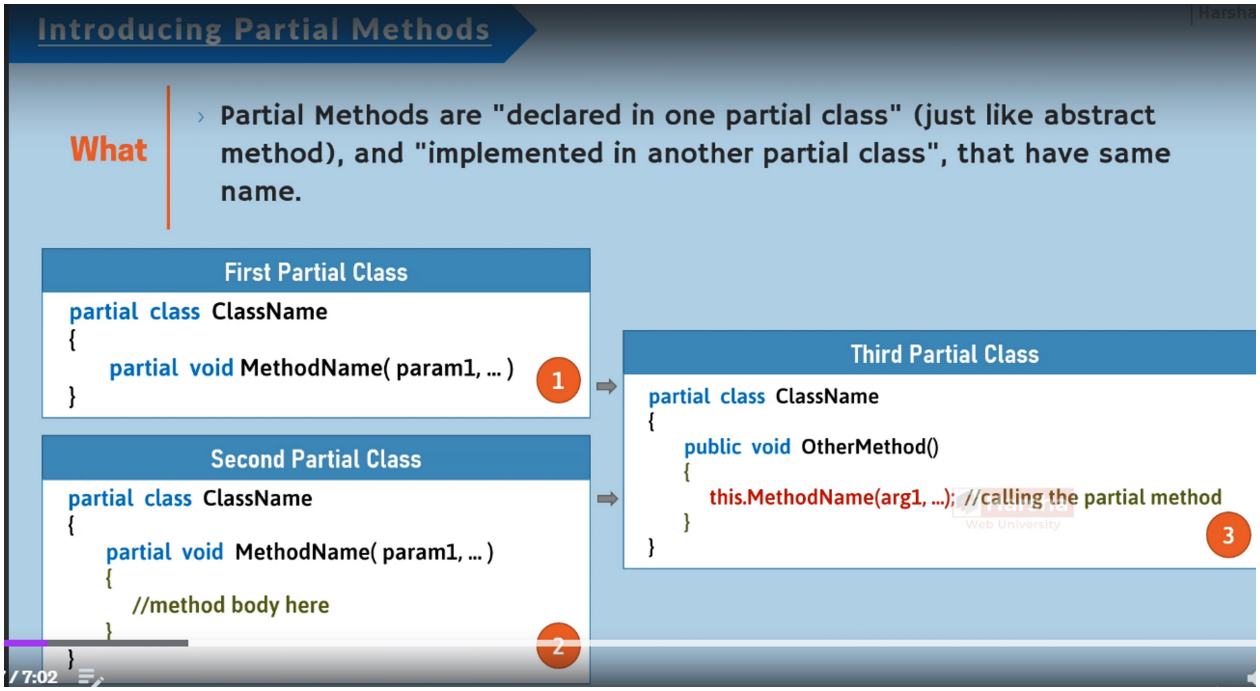
```
partial class ClassName
{
    partial void MethodName( param1, ... )
}
```

1

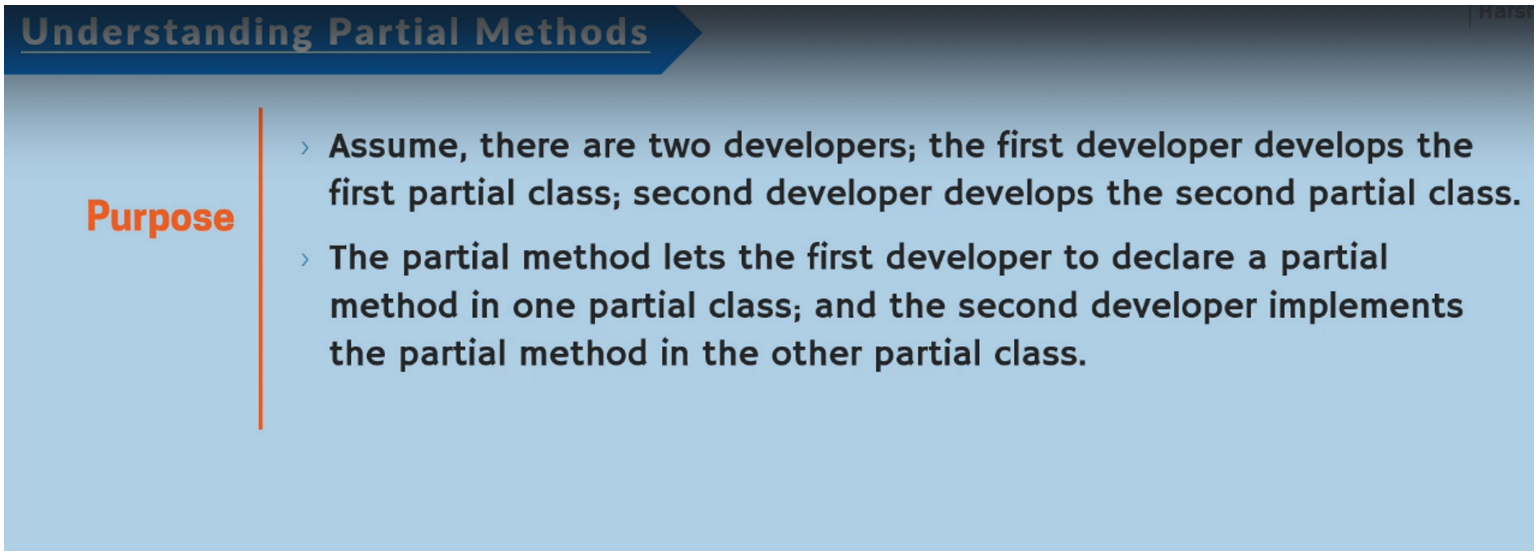
**Second Partial Class**

```
partial class ClassName
{
    partial void MethodName( param1, ... )
    {
        //method body here
    }
}
```

2




Partial Methods are by default private.



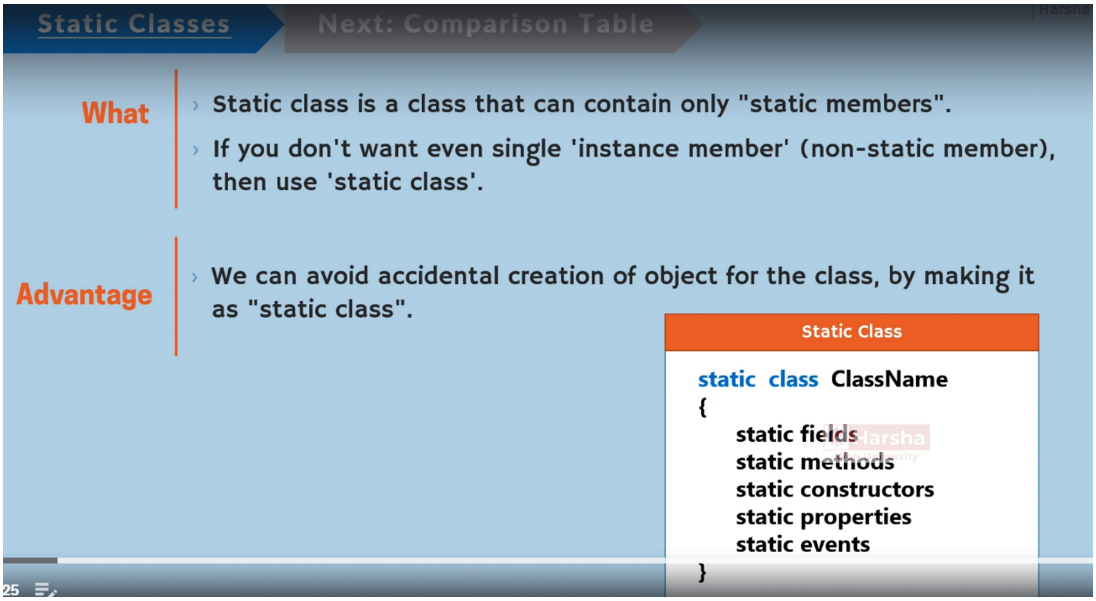
## Understanding Partial Methods

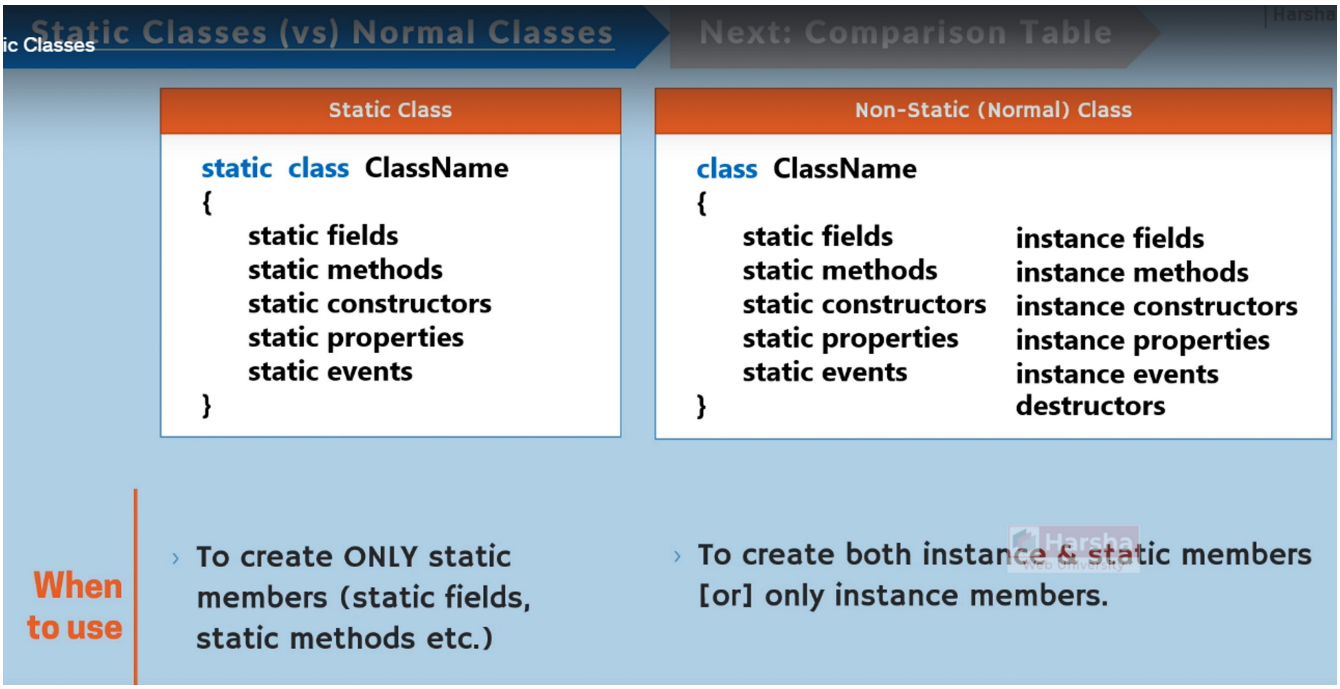
**Rules**

- Partial Methods can only be created in partial class or partial structs.
- Partial Methods are implicitly private. It can't have any other access modifier.
- Partial Methods can have only "void" return type.
- Implementation of partial methods is optional. If there is no implementation of partial methods in any parts of the partial class, the method calls are removed by the compiler, at compilation time.
- If you are building large class libraries and decide extension of methods to other developers, partial methods can be used.




Whenever you want only static members in your class, then define that class as a static class.

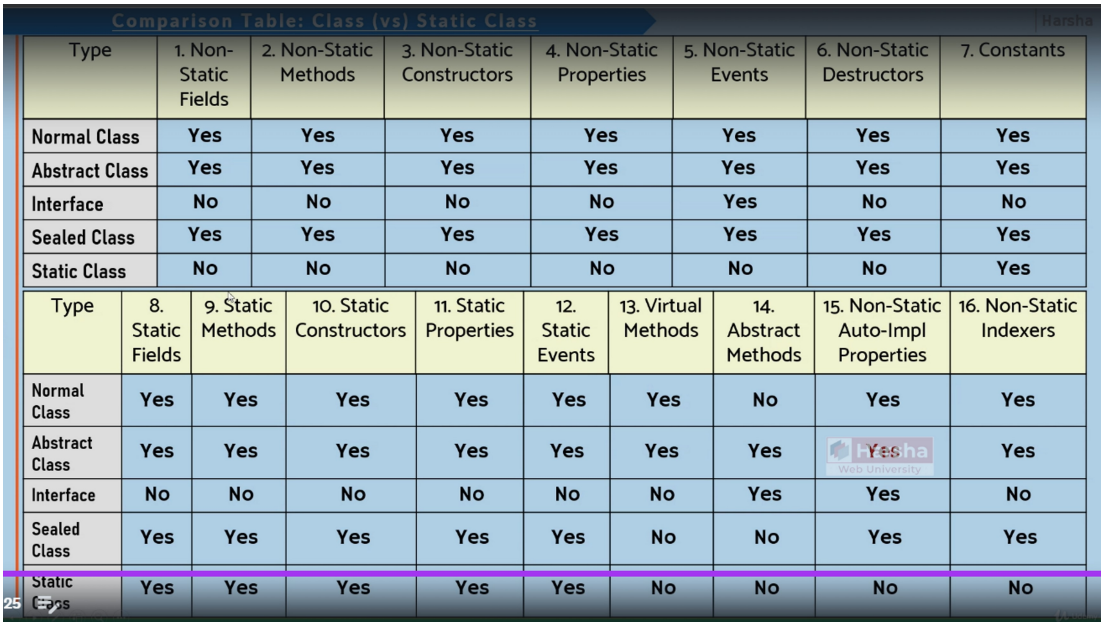

Static Classes
Next: Comparison Table
What
› Static class is a class that can contain only "static members".
› If you don't want even single 'instance member' (non-static member), then use 'static class'.
Advantage
› We can avoid accidental creation of object for the class, by making it as "static class".
Static Class
static class ClassName
{
static fields
static methods
static constructors
static properties
static events
}


Static Classes (vs) Normal Classes
Next: Comparison Table
Static Class
static class ClassName
{
static fields
static methods
static constructors
static properties
static events
}
Non-Static (Normal) Class
class ClassName
{
static fields
static methods
static constructors
static properties
static events
instance fields
instance methods
instance constructors
instance properties
instance events
destructors
}
When to use
› To create ONLY static members (static fields, static methods etc.)
› To create both instance & static members [or] only instance members.

## Comparison Table: Class (vs) Static Class

**Based on inheritance and object**

| Class Type | Can Inherit from Other Classes | Can Inherit from Other Interfaces | Can be Inherited | Can be Instantiated |
|---|---|---|---|---|
| Normal Class | Yes | Yes | Yes | Yes |
| Abstract Class | Yes | Yes | Yes | No |
| Interface | No | Yes | Yes | No |
| Sealed Class | Yes | Yes | No | Yes |
| Static Class | No | No | No | No |




| Type | 1. Non-Static Fields | 2. Non-Static Methods | 3. Non-Static Constructors | 4. Non-Static Properties | 5. Non-Static Events | 6. Non-Static Destructors | 7. Constants |
|---|---|---|---|---|---|---|---|
| Normal Class | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Abstract Class | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Interface | No | No | No | No | Yes | No | No |
| Sealed Class | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Static Class | No | No | No | No | No | No | Yes |

| Type | 8. Static Fields | 9. Static Methods | 10. Static Constructors | 11. Static Properties | 12. Static Events | 13. Virtual Methods | 14. Abstract Methods | 15. Non-Static Auto-Impl Properties | 16. Non-Static Indexers |
|---|---|---|---|---|---|---|---|---|---|
| Normal Class | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes |
| Abstract Class | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Interface | No | No | No | No | No | No | Yes | Yes | No |
| Sealed Class | Yes | Yes | Yes | Yes | Yes | No | No | Yes | Yes |
| Static Class | Yes | Yes | Yes | Yes | Yes | No | No | No | No |

1. Enumerations

C#

Harsha Vardhan

- UI Expert
- .NET Expert
- Module Lead
- Corporate Instructor

# Enumerations

## Enumerations

**What**

- Enumeration is a collection of constants.
- Enumeration is used to specify the list of options allowed to be stored in a field / variable.

**Goal**

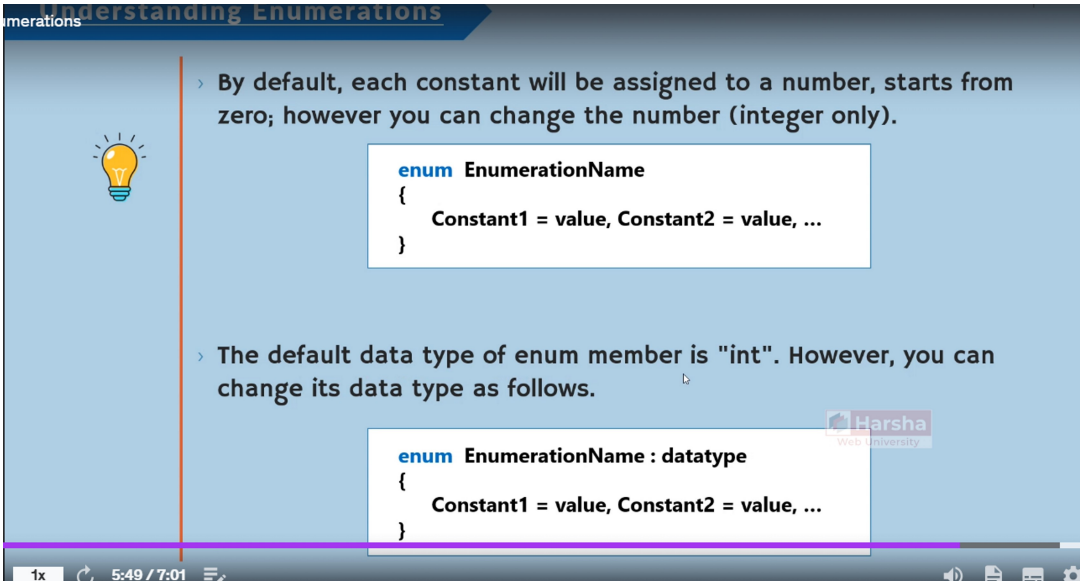
- Use enumeration if you don't want to allow other developers to assign other value into a field / variable, other than the list of values specified in the enumeration

**Accessing member**

- EnumerationName.ConstantName

**Enumeration**

```
enum EnumerationName
{
    Constant1, Constant2, ...
}
```


Understanding Enumerations
By default, each constant will be assigned to a number, starts from zero; however you can change the number (integer only).
enum EnumerationName
{
Constant1 = value, Constant2 = value, ...
}
The default data type of enum member is "int". However, you can change its data type as follows.
enum EnumerationName : datatype
{
Constant1 = value, Constant2 = value, ...
}
Harsha
Web University
1x
5:49 / 7:01