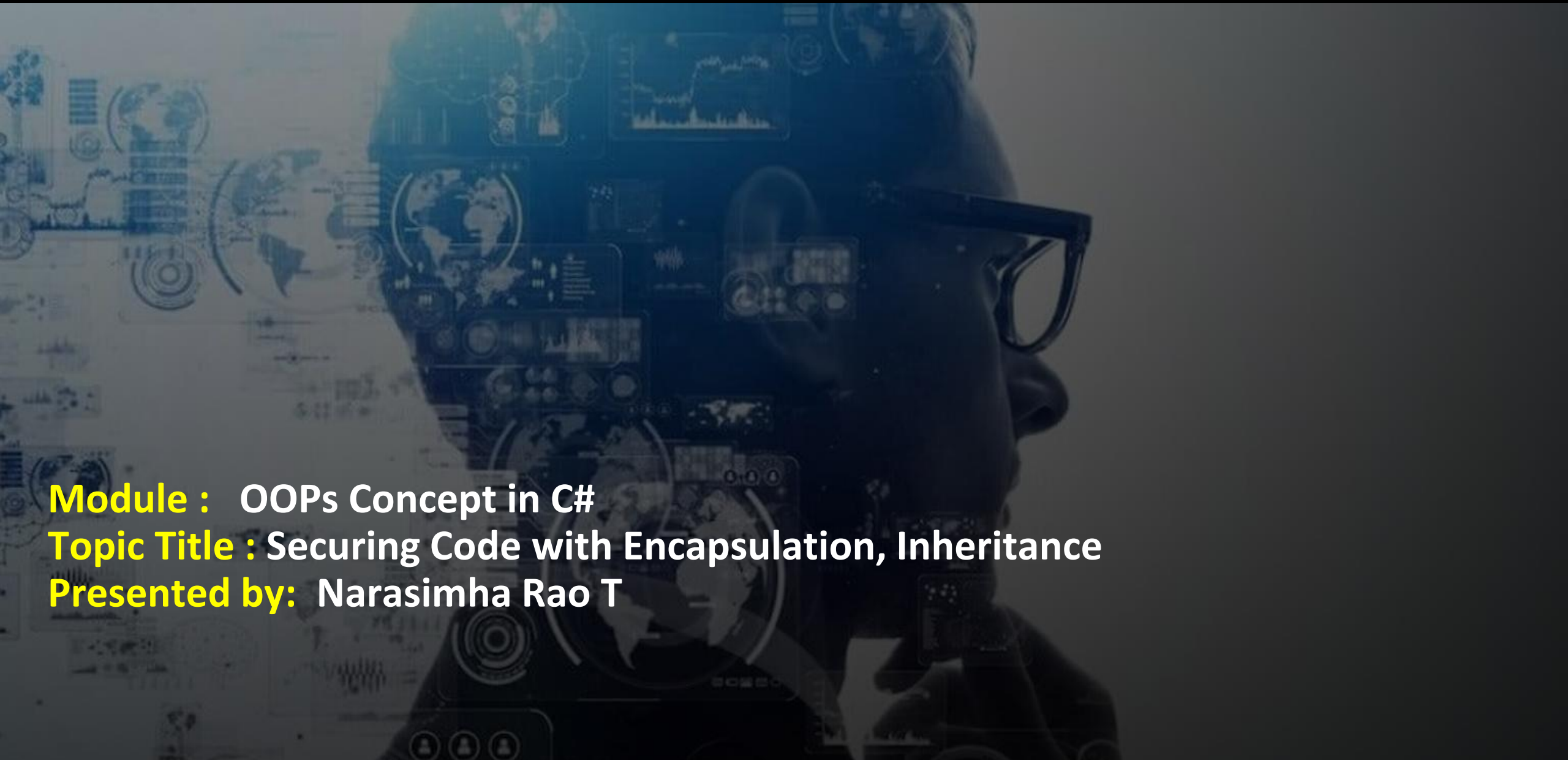
upGrad ENTERPRISE

.NET FSD Bootcamp Training

17th July, 2025

upGrad ENTERPRISE



Module : OOPs Concept in C#

Topic Title : Securing Code with Encapsulation, Inheritance

Presented by: Narasimha Rao T

Weekly Schedule

Day	Date	Topic
Day-4	14-07-2025	Control Structures & Loops
Day-5	15-07-2025	Working with Methods
Day-6	16-07-2025	Object Oriented Programming – P1
Day-7	17-07-2025	Object Oriented Programming – P2
Day-8	18-07-2025	Object Oriented Programming – P3

upGrad ENTERPRISE

OOPs Concepts in C# : Inheritance and Abstract

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

Day-7

Index

Securing Code with Encapsulation, Inheritance

1. Access Modifiers
2. Inheritance and base keyword
3. Abstract Classes & Methods
4. Static Class
5. Partial Class
6. Q & A

1. Access Modifier

Access modifiers control the **visibility** and **accessibility** of classes, methods, and other members.

Types:

Modifier	Description
`public`	Accessible from anywhere.
`private`	Accessible only within the containing class.
`protected`	Accessible within the containing class and by derived classes.
`internal`	Accessible within the same assembly.
`protected internal`	Accessible within the same assembly and by derived classes.

2. Inheritance

Allows a class to inherit members from another class.

Syntax:

```
class Parent {
    public void Greet() => Console.WriteLine("Hello");
}

class Child : Parent {
    public void Play() => Console.WriteLine("Playing");
}
```

- **Single Inheritance** is supported in C# (one base class).
- Use `: BaseClass` syntax to inherit.

3. Usage of `base` Keyword

Used to **refer to the base class** from a derived class.

Use Cases:

- Calling **base class constructor**
- Calling **base class method or property**

Example

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

class Child : Parent {
    public override void Show() {
        base.Show(); // Call base class method
        Console.WriteLine("Child");
    }
}
```

4. Abstract Classes & Methods

Used to define a base class with **incomplete implementation**.

Abstract Class:

- Cannot be instantiated.
- Can have **abstract** and **non-abstract** members.
- Used as a base class.

Abstract Class Example

```
abstract class Animal {
    public abstract void MakeSound(); // Abstract method
    public void Sleep() {
        Console.WriteLine("Sleeping...");
    }
}
```

Abstract Method:

- Declared in abstract class.
- No body (implementation) in base class.
- Must be **overridden** in derived class.

```
class Dog : Animal {
    public override void MakeSound() {
        Console.WriteLine("Bark");
    }
}
```

5. Static Class

- Cannot be instantiated.
- Contains only **static members**.
- Useful for utility/helper classes.

```
static class MathHelper {
    public static int Square(int x) => x * x;
}
```

6. Static Members

- Belong to the class, **not instances**.
- Shared across all instances.

```
class Counter {
    public static int Count = 0;

    public Counter() {
        Count++;
    }
}
```

7. Static Constructor

- Initializes **static members**.
- Called **once**, automatically before any access to static members.

```
class Logger {
    static Logger() {
        Console.WriteLine("Logger initialized");
    }

    public static void Log(string msg) => Console.WriteLine(msg);
}
```

8. Partial Class

Allows a class to be **split across multiple files**.

Use Case:

- Helps organize large classes.
- Used by code generators (e.g., Windows Forms).

```
// File1.cs
partial class MyClass {
    public void MethodA() {}
}

// File2.cs
partial class MyClass {
    public void MethodB() {}
}
```

9. Sealed Class

- Cannot be inherited.
- Used to **prevent further inheritance** for security or design reasons.

```
sealed class FinalClass {
    public void Show() => Console.WriteLine("No further inheritance");
}

// class Derived : FinalClass { } // Error
```

10. Summary

Concept	Key Point
Access Modifiers	Control visibility of members
Abstract Class	Cannot be instantiated, base class with abstract methods
Inheritance	Enables reuse of base class functionality
`base` keyword	Access base class methods or constructor
Static Members	Shared across all objects
Static Constructor	Initializes static data, runs once

10. Summary

Concept	Key Point
Static Class	No instances, only static members
Partial Class	Class split into multiple files
Sealed Class	Prevents class inheritance

upGrad ENTERPRISE

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com