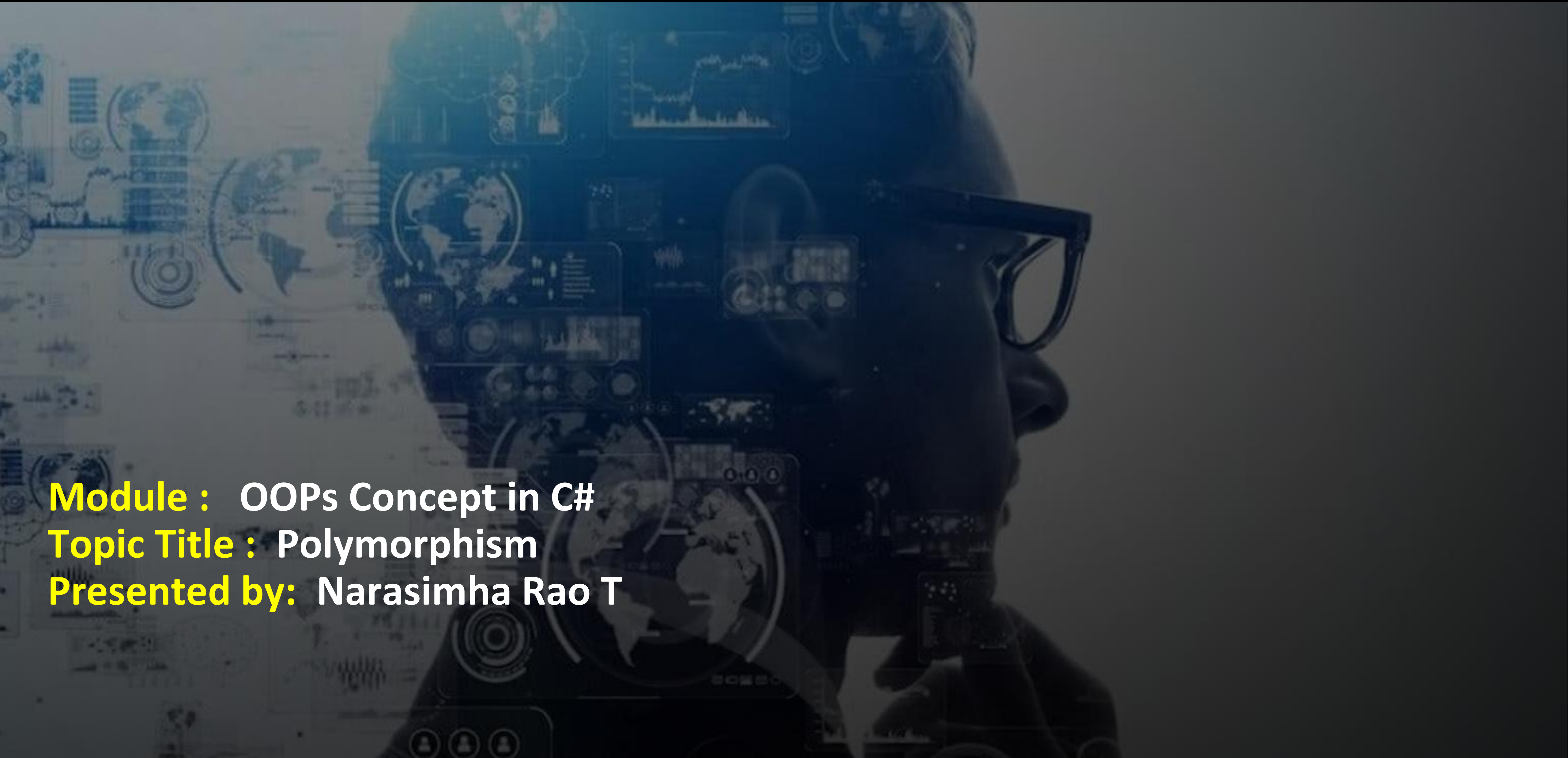
upGrad ENTERPRISE

.NET FSD Bootcamp Training

18th July, 2025



Module : OOPs Concept in C#

Topic Title : Polymorphism

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	OOP in C# - Polymorphism Weekly Assessment

Object-Oriented Concepts in C#: Polymorphism

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

Index

1. Polymorphism: Method Hiding, Overriding
2. Method Overloading (Completed in Day-5)
3. is/as operators
4. Interfaces
5. Weekly Assessment
6. Q & A

1. Polymorphism

Definition:
Polymorphism allows objects to be treated as instances of their parent class rather than their actual class. It enables the same method to behave differently based on the object instance.

Types:

- **Compile-time polymorphism (Static Binding):** Achieved via method overloading and operator overloading.
- **Run-time polymorphism (Dynamic Binding):** Achieved via method overriding using inheritance and `virtual` / `override` keywords.

Example:

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

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

2. Method Hiding

Definition:

Method hiding occurs when a derived class defines a method with the same name as one in its base class without overriding it. It hides the base method using the `new` keyword.

Syntax:

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

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

3. Method Overriding

Definition:

Overriding allows a subclass to provide a specific implementation of a method already defined in its base class.

Keywords:

- `virtual` → in base class
- `override` → in derived class

Example:

```
class Parent {
    public virtual void Print() {
        Console.WriteLine("Parent Print");
    }
}

class Child : Parent {
    public override void Print() {
        Console.WriteLine("Child Print");
    }
}
```

4. Method Overloading

Definition:
Method overloading allows multiple methods in the same class with the same name but different **parameter lists** (type, number, or order).

Example:

```
class MathOps {
    public int Add(int a, int b) {
        return a + b;
    }

    public double Add(double a, double b) {
        return a + b;
    }
}
```

5. `is` and `as` Operators

`is` Operator

- Checks if an object is of a specific type.
- Returns `true` or `false`.

```
object obj = "Hello";
if (obj is string) {
    Console.WriteLine("It's a string");
}
```

`as` Operator

- Attempts to cast an object to a type.
- Returns the object if successful, otherwise `null`.

```
object obj = "Hello";
string str = obj as string;
if (str != null) {
    Console.WriteLine("Casting successful");
}
```

6. Interfaces

Definition:

An interface defines a contract. A class or struct that implements an interface must implement all of its members.

Syntax:

```
interface IShape {
    void Draw();
}

class Circle : IShape {
    public void Draw() {
        Console.WriteLine("Drawing Circle");
    }
}
```

Key Points:

- No implementation in the interface itself.
- A class can implement **multiple interfaces** (unlike class inheritance).
- Members are public by default; access modifiers are not allowed.

Summary Table:

Concept	Type	Key Keyword(s)	Notes
Polymorphism	OOP	virtual, override	Same method behaves differently in derived types
Method Hiding	Compile-Time	new	Hides base method; depends on reference type
Method Overriding	Run-Time	virtual, override	Changes base class behavior in derived class
Method Overloading	Compile-Time	-	Same name, different parameters

Summary Table:

Concept	Type	Key Keyword(s)	Notes
`is` operator	Type check	is	Returns true/false
`as` operator	Type cast	as	Returns casted object or null
Interface	Contract	interface	Enforces implementation of specified members

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com