

# Advance Programming Techniques (APT)

Lecture # 11

**Ehtisham Rasheed**

Department of Computer Science  
University of Gurjat, Gurjat

# Delegates in C#

- A **delegate** is a reference type variable that holds the reference to a method
- C# delegates are similar to pointers to functions, in C or C++
- All delegates are implicitly derived from the **System.Delegate** class
- Key Points
  - A delegate defines the signature of methods it can point to
  - It can reference both static and instance methods
  - Delegates are type-safe, meaning the method signature must match the delegate declaration
  - They are the foundation of events and anonymous functions in C#

# When to Use Delegates

- Callback Mechanism
  - Delegates are used to implement callback. This means we can specify what function to call later when an event happens, without hardcoding the function call
- Event Handling
  - Delegates are extensively used in event-driven programming to define event handlers. Multiple methods can subscribe to the same event using multicast delegates.
- Decoupling
  - Code that uses a delegate does not need to know exactly which method will be called. This allows swapping different methods dynamically or at runtime without changing the calling code.
- Reusability and Extensibility
  - Delegates provide a way to reuse calling logic with different methods passed in, making the code more extensible to future changes without modifying existing code

# Declaring Delegates

- We define a delegate just like we define a normal method
- Delegate also has a return type and parameter

```
public delegate int myDelegate(int x);
```

Here,

- `delegate` - a keyword
- `int` - return type of delegate
- `myDelegate` - delegate name
- `int x` - parameter that the delegate takes

# Example 1

```
1 reference
static int calculateSum(int a, int b)
{
    return a + b;
}
public delegate int myDelegate(int num1, int num2);
0 references
static void Main(string[] args)
{
    myDelegate d = new myDelegate(calculateSum);
    int result = d(5, 10);
    Console.WriteLine(result);
}
```

# Example 2

```
public delegate int Operation(int num1, int num2);

1 reference
public static int Add(int a, int b) => a + b;
1 reference
public static int Multiply(int a, int b) => a * b;

0 references
static void Main(string[] args)
{
    Operation op = Add;
    Console.WriteLine("Addition: " + op(5,10));

    op = Multiply;
    Console.WriteLine("Multiplication: " + op(5,10));
}
```

# Example 3 – Multicast Delegate

```
public delegate void Notify();

1 reference
public static void MethodA() => Console.WriteLine("Method A executed");
1 reference
public static void MethodB() => Console.WriteLine("Method B executed");

0 references
static void Main(string[] args)
{
    Notify notify = MethodA;
    notify += MethodB;
    notify();
}
```