Delegates and Events (C#)

Delegates in C# allow you to reference a method like a variable , creating more flexible and dynamic programs.

Delegates are particularly useful for event handling , asynchronous programming and creating more modular code.

It is called a function pointer means we can point to the address of a function using a delegate to call the actual function.

if we do not want to call a particular function in the program then we can use the delegate concepts to call that function indirectly means we can say that delegate is an intermediate between function and object.

```
using System;

class Program
{
    // Define a delegate type
    delegate void GreetDelegate(string name);

    // Method that matches the delegate signature
    static void Greet(string name)
    {
        Console.WriteLine($"Hello, {name}!");
    }

    static void Main()
    {
        // Create an instance of the delegate and point it to the Greet method
        GreetDelegate greetDelegate = new GreetDelegate(Greet);

        // Call the delegate
        greetDelegate("John");

        // Alternatively, you can directly call the method via the delegate
        greetDelegate.Invoke("Alice");
```

```
    }
}
```

#Event

In c# events allow objects to communicate with each other by sending signals when something interesting occurs.

Using events allows for separation of concerns , making your code more modular and easier to maintain.

Events are actions that allow classes or objects to inform each other classes or objects when an interesting phenomenon occurs.

```
using System;

class Program
{
    // Define an event using a delegate
    public delegate void NotifyEventHandler(string message);

    // Declare an event of type NotifyEventHandler
    public static event NotifyEventHandler OnNotify;

    static void Main()
    {
        // Subscribe to the event
        OnNotify += NotifyUser;

        // Trigger the event
        OnNotify?.Invoke("Event has been triggered!");

        // Unsubscribe from the event
        OnNotify -= NotifyUser;
    }
```

```
    // Event handler method
    static void NotifyUser(string message)
    {
        Console.WriteLine($"Event message: {message}");
    }
}
```