

C# Fundamentals

6. Delegates, Events, and Basic Event HandlingObjective:Requirements:

- Build a console-based event-driven application (e.g., a counter that triggers an event at a threshold).
- Define a delegate and an event that fires when a counter reaches a specific value.
- Create multiple event handler methods that perform actions when the event is raised.
- In your main loop, increment the counter and raise the event when appropriate.
- Demonstrate how events can decouple the producer and consumer logic.

Program code:

```
using System;
```

```
// declare a delegate that defines the event signature
```

```
// sender is the object that raised it, e holds event data.
```

```
public delegate void ThresholdReachedEventHandler(object sender, EventArgs e);
```

```
// counter class that raises an event when a threshold is hit
```

```
public class Counter
```

```
{
```

```
    private int _threshold;
```

```
    private int _currentCount;
```

```
// declare the event using the delegate
```

```
public event ThresholdReachedEventHandler ThresholdReached;
```

```
public Counter(int threshold)
```

```
{
```

```
    _threshold = threshold;
```

```
    _currentCount = 0;
```

```
}
```

```

// method to increment the counter
public void Increment()
{
    _currentCount++;
    Console.WriteLine($"Counter: {_currentCount}");

    // Raise the event when threshold is reached
    if (_currentCount == _threshold)
    {
        OnThresholdReached(EventArgs.Empty);
    }
}

// Protected method to raise the event
protected virtual void OnThresholdReached(EventArgs e)
{
    // checks if any method is subscribed (?) and invokes the event.
    ThresholdReached?.Invoke(this, e);
}
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Enter the threshold for the counter:");
        int threshold = int.Parse(Console.ReadLine());

        Counter counter = new Counter(threshold);

```

```

// subscribe multiple event handlers
counter.ThresholdReached += ThresholdReachedMessage;
counter.ThresholdReached += LogThresholdReached;

// counter loop
Console.WriteLine("\nStarting counter... Press Enter to increment.");
while (true)
{
    Console.ReadLine();
    counter.Increment();
}

// event handler 1
static void ThresholdReachedMessage(object sender, EventArgs e)
{
    Console.WriteLine("----- Threshold reached! Performing action... -----");
}

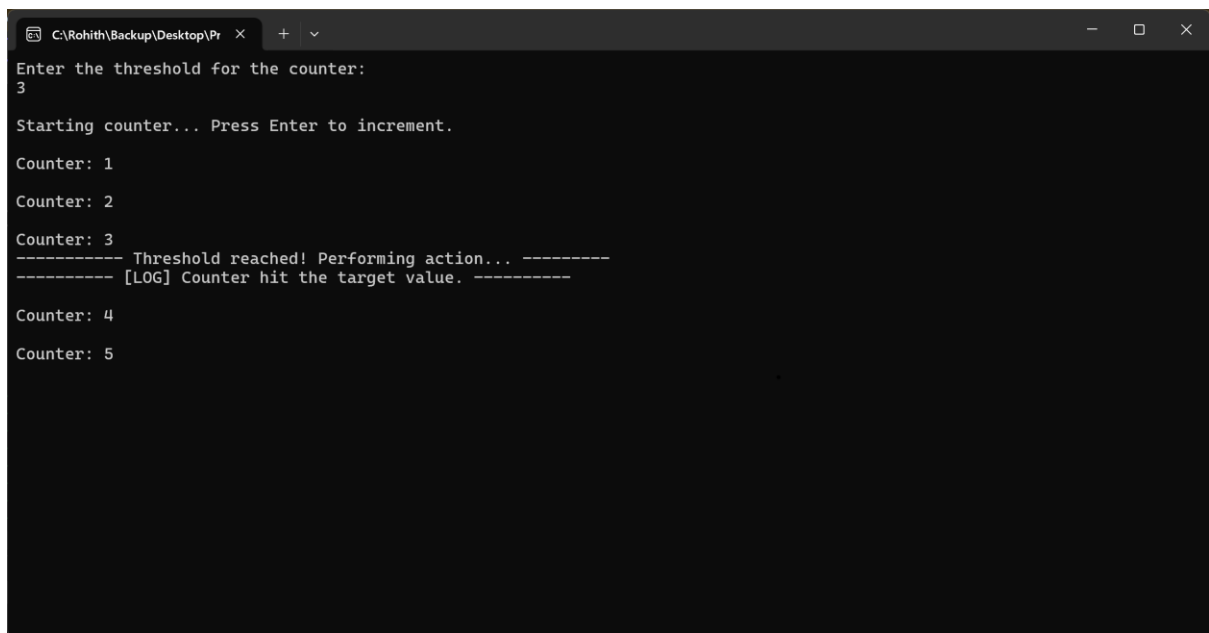
// event handler 2
static void LogThresholdReached(object sender, EventArgs e)
{
    Console.WriteLine("----- [LOG] Counter hit the target value. -----");
}

// Delegate: Specifies method signature for event handlers.
// Event: Mechanism for communication between objects.
// Event Handler: A method that responds to the event.
// Decoupling: The Counter class does not care what the subscribers do—just that they get notified.

```

```
// EventArgs - A base class for passing event data.  
  
// EventArgs.Empty - A predefined static instance for when no extra data is needed.  
  
// OnThresholdReached(EventArgs.Empty) - Raises the event without custom data.  
  
// Custom EventArgs - Used when we need to pass additional event data.
```

Output:



```
C:\Rohith\Backup\Desktop\Pr x + v  
Enter the threshold for the counter:  
3  
  
Starting counter... Press Enter to increment.  
  
Counter: 1  
Counter: 2  
Counter: 3  
----- Threshold reached! Performing action... -----  
----- [LOG] Counter hit the target value. -----  
  
Counter: 4  
Counter: 5
```