

Events

This lesson will teach you about events in C# and how they're related with delegates!

An event is a special kind of delegate that facilitates **event-driven programming**.

Events are class members that cannot be called outside of the class regardless of its access specifier.

So, for example, an event declared to be public would allow other classes the use of `+=` and `-=` on the event, but firing the event (i.e. invoking the delegate) is only allowed in the class containing the event.

A simple example:

```
using System;

public class DemoClass
{
    public delegate void MyEventRaiser();
    public event MyEventRaiser DivisibleBy10Event;

    public int isDivisibleBy10(int a, int b)
    {
        int ans = a * b;
        if ((ans % 10 == 0) && (DivisibleBy10Event != null))
        { DivisibleBy10Event(); }
        return ans;
    }
}

class Program
{
    static void Main()
    {
        DemoClass a = new DemoClass();
        a.DivisibleBy10Event += new DemoClass.MyEventRaiser(func);

        int ans = a.isDivisibleBy10(6,9);
        Console.WriteLine(ans);
    }
}
```



```
    ans = a.isDivisibleBy10(5,2);  
    Console.WriteLine(ans);  
}  
  
static void func()  
{  
    Console.WriteLine("Answer is divisible by 10!");  
}  
}
```



This code creates a new instance of `DivisibleBy10Event` event which in turn calls the method `func()`.

Even though the event is declared public, it cannot be directly fired anywhere except in the class containing it.

You've now pretty much grasped the idea of delegates in C#. Let us now have a quick quiz!