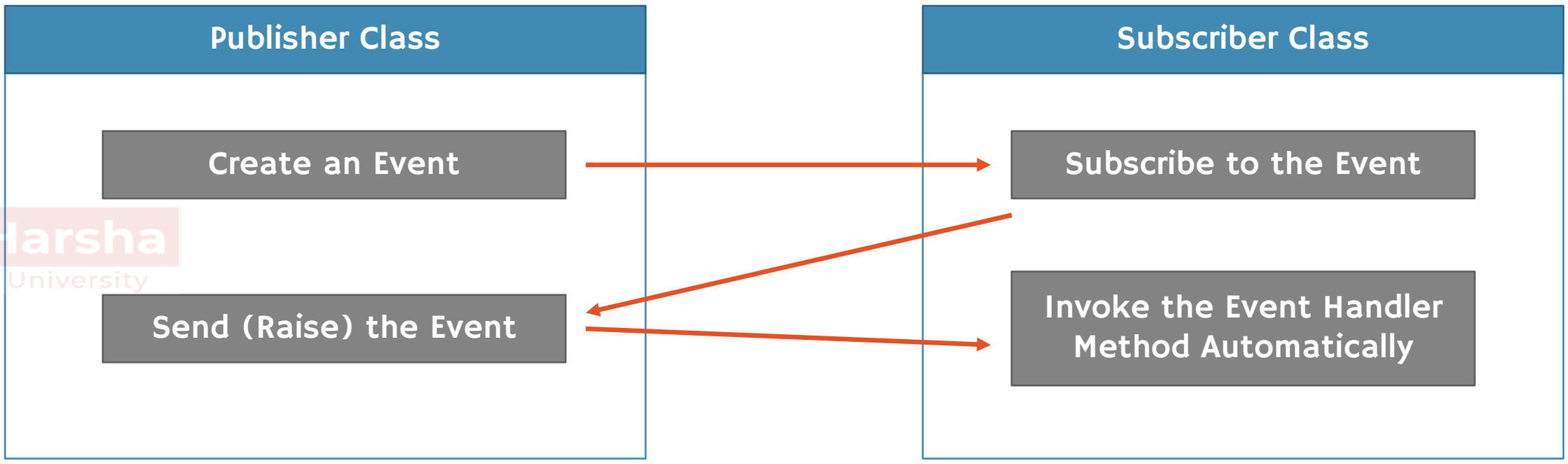
C#

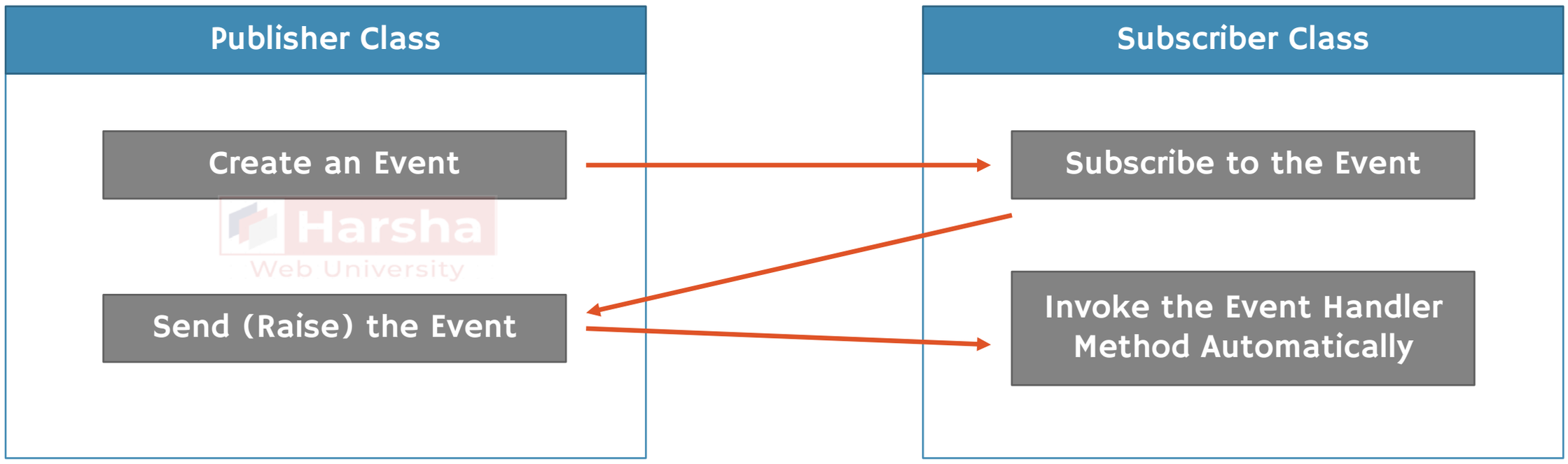
Events

What

- Event is a multi-cast delegate that stores one or more methods; and invoke them every time when the event is raised (called).
- The event can be raised only in the same class, in which it is created.



Publisher vs Subscriber



- The class that sends (or raises) events (notifications), is called as "publisher class".
- Publisher class sends events; then Subscriber class receives events.

- The class that receives (or subscribes or handles) events (notifications), is called as "subscriber class".

- Events enable a class to send notifications to other classes, when something occurs.
- Publisher class sends events; Subscriber class receives events.

Process flow of Events

- The Publisher class creates an event.
- The Subscriber class subscribes to the event; that means an "event handler" method is created in the subscriber class. The "event handler" method is nothing but, the method which is dedicated to be executed when the event is raised.
- The publisher class can send (raise) events.
- Every time, when the event is raised by the publisher, the corresponding "event handler" method executes automatically.

Creating Events

Steps for creating Events

Create a Delegate — 1

```
public delegate ReturnType DelegateTypeName(paramI, param2, ...);
```

Create an Event in Publisher Class — 2

```
class Publisher
{
    private DelegateTypeName eventVariable;

    public event DelegateTypeName EventName
    {
        add
        {
            eventVariable += value;
        }
        remove
        {
            eventVariable -= value;
        }
    }
}
```

Raise the event in Publisher Class — 3

```
if (EventName != null) EventName(argI, arg2, ...);
```

Create Event Handler Method in Subscriber Class — 4

```
class Subscriber
{

    public ReturnType EventHandlerMethodName(paramI, param2,
    ...)
    {
        Method body here
    }
}
```

Subscribe to the Event [Inside or Outside the Subscriber class] 5

```
EventName += EventHandlerMethodName;
```


- The event should be created based on the delegate. That means, the event accepts the methods that are having specific parameters and return type, defined in the delegate.
- An event can have multiple subscribers.
- A subscriber can subscribe multiple events from multiple publishers.
- Events are basically signals to inform to other classes, that some important thing happened in the publisher class.
- Events are special kind of "multi-cast delegates", which can be raised only within the same class, in which they are created.
- Events can be static, virtual, sealed and abstract.
- Events will not be raised (throws exception), if there is no at least one subscriber.
- Events can be defined in interfaces.
- It's not a good idea to return value in events.

Auto-Implemented Events

What

- "Auto-Implemented Events" provide a shortcut syntax to create events with less code.
- In this case, you need not create "add" and "remove" accessors; the compiler does the same automatically.

How

Create an Auto-Implemented Event in Publisher Class

```
class Publisher
{
    public event MyDelegateType MyEvent;
}
```

- You also not required to create a private multi-cast delegate; the compiler does the same automatically.
- Disadvantage: We can't define custom logic for "add accessor" and "remove accessor".