C# - EVENTS

Events are user actions such as key press, clicks, mouse movements, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur. For example, interrupts. Events are used for inter-process communication.

Using Delegates with Events

The events are declared and raised in a class and associated with the event handlers using delegates within the same class or some other class. The class containing the event is used to publish the event. This is called the **publisher** class. Some other class that accepts this event is called the **subscriber** class. Events use the **publisher-subscriber** model.

A **publisher** is an object that contains the definition of the event and the delegate. The event-delegate association is also defined in this object. A publisher class object invokes the event and it is notified to other objects.

A **subscriber** is an object that accepts the event and provides an event handler. The delegate in the publisher class invokes the method *eventhandler* of the subscriber class.

Declaring Events

To declare an event inside a class, first a delegate type for the event must be declared. For example,

```
public delegate void BoilerLogHandler(string status);
```

Next, the event itself is declared, using the **event** keyword:

```
//Defining event based on the above delegate
public event BoilerLogHandler BoilerEventLog;
```

The preceding code defines a delegate named *BoilerLogHandler* and an event named *BoilerEventLog*, which invokes the delegate when it is raised.

Example 1

```
using System;
namespace SimpleEvent
{
   using System;

   public class EventTest
   {
      private int value;
      public delegate void NumManipulationHandler();
      public event NumManipulationHandler ChangeNum;
      protected virtual void OnNumChanged()
      {
         if (ChangeNum != null)
         {
            ChangeNum();
         }
         else
         {
            Console.WriteLine("Event fired!");
         }
      }

      public EventTest(int n )
      {
         SetValue(n);
```

```
      }

      public void SetValue(int n)
      {
         if (value != n)
         {
            value = n;
            OnNumChanged();
         }
      }
   }

   public class MainClass
   {
      public static void Main()
      {
         EventTest e = new EventTest(5);
         e.SetValue(7);
         e.SetValue(11);
         Console.ReadKey();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result:

```
Event Fired!
Event Fired!
Event Fired!
```

Example 2

This example provides a simple application for troubleshooting for a hot water boiler system. When the maintenance engineer inspects the boiler, the boiler temperature and pressure is automatically recorded into a log file along with the remarks of the maintenance engineer.

```
using System;
using System.IO;

namespace BoilerEventAppl
{
   // boiler class
   class Boiler
   {
      private int temp;
      private int pressure;
      public Boiler(int t, int p)
      {
         temp = t;
         pressure = p;
      }

      public int getTemp()
      {
         return temp;
      }

      public int getPressure()
      {
         return pressure;
      }
   }

   // event publisher
   class DelegateBoilerEvent
   {
      public delegate void BoilerLogHandler(string status);
```

```
    //Defining event based on the above delegate
    public event BoilerLogHandler BoilerEventLog;

    public void LogProcess()
    {
        string remarks = "O. K";
        Boiler b = new Boiler(100, 12);
        int t = b.getTemp();
        int p = b.getPressure();
        if(t > 150 || t < 80 || p < 12 || p > 15)
        {
            remarks = "Need Maintenance";
        }
        OnBoilerEventLog("Logging Info:\n");
        OnBoilerEventLog("Temparature " + t + "\nPressure: " + p);
        OnBoilerEventLog("\nMessage: " + remarks);
    }

    protected void OnBoilerEventLog(string message)
    {
        if (BoilerEventLog != null)
        {
            BoilerEventLog(message);
        }
    }
}

// this class keeps a provision for writing into the log file
class BoilerInfoLogger
{
    FileStream fs;
    StreamWriter sw;
    public BoilerInfoLogger(string filename)
    {
        fs = new FileStream(filename, FileMode.Append, FileAccess.Write);
        sw = new StreamWriter(fs);
    }

    public void Logger(string info)
    {
        sw.WriteLine(info);
    }

    public void Close()
    {
        sw.Close();
        fs.Close();
    }
}

// The event subscriber
public class RecordBoilerInfo
{
    static void Logger(string info)
    {
        Console.WriteLine(info);
    }//end of Logger

    static void Main(string[] args)
    {
        BoilerInfoLogger filelog = new BoilerInfoLogger("e:\\boiler.txt");
        DelegateBoilerEvent boilerEvent = new DelegateBoilerEvent();
        boilerEvent.BoilerEventLog += new
        DelegateBoilerEvent.BoilerLogHandler(Logger);
        boilerEvent.BoilerEventLog += new
        DelegateBoilerEvent.BoilerLogHandler(filelog.Logger);
        boilerEvent.LogProcess();
        Console.ReadLine();
        filelog.Close();
```

```
      }//end of main

   }//end of RecordBoilerInfo
}
```

When the above code is compiled and executed, it produces the following result:

```
Logging info:

Temperature 100
Pressure 12

Message: O  K
```