Практическая работа по паттерну Singleton в C#

Цель работы:

Целью этой практической работы является изучение и применение События (event) на языке программирования C#.

Задачи:

1. Изучить основные концепции События (event).

2. Реализовать простой События (event).

3. Создать несколько сценариев использования События (event) в разных контекстах.

Теоретическая часть.

События сигнализируют системе о том, что произошло определенное действие. И если нам надо отследить эти действия, то как раз мы можем применять события.

В C# событие определяется следующим образом:

event — ключевое слово, которое сообщает нам, что перед нами событие.

delegateType — это тип делегата Делегат описывает то, как должен выглядеть метод в подписчике, который будет обрабатывать событие, а также то, какие параметры необходимо передавать подписчику. То есть, делегат, образно выражаясь, представляет из себя некий договор между издателем и подписчиком как они будут между собой общаться.

EventName — название события.

Реализация События:

Подготовим эти три простейших класса, оставив точку входа в программу **main** нетронутой.

Класс **ClassCounter** и его метод *Count()* в котором будет производится счет.

```
class ClassCounter  //Это класс - в котором производится счет.
  {
    public void Count()
    {
      //Здесь будет производиться счет
    }
}
```

Два других класса (имена им **Handler_I** и **Handler_II**), которые должны реагировать на возникновение события методами public void *Message()*. У каждого по методу, как и договаривались.

```
class Handler_I //Это класс, реагирующий на событие (счет равен 71) записью строки в консоли.
  {
    public void Message()
    {
      //Не забудьте using System для вывода в консольном приложении

      Console.WriteLine("Пора действовать, ведь уже 71!");
    }
  }

class Handler_II
  {
    public void Message()
    {
      Console.WriteLine("Точно, уже 71!");
    }
  }
```

Напомню, когда счетчик будет считать до 100 и достигнет 71, должны сработать методы Message() для классов Handler_I и Handler_II.
Теперь вернемся к классу ClassCounter и создадим счетчик при помощи цикла for с переменной-счетчиком int i.

```
class ClassCounter  //Это класс - в котором производится счет.
  {
    public void Count()
    {
      for (int i = 0; i < 100; i++)
      {
```

```
        }
    }
}
```

Первый этап завершен. У нас есть класс счетчик и два класса, которые будут выводить сообщения. Условия задачи: как только i=71, должны сработать методы Message() для двух классов Handler_I и Handler_II.

Оформление события.

Абстрагируемся от программирования. Событие, которое мы хотим создать, будет представлять фразу "… счетчик считает. И как только он будет равен 71, должны выполниться действия". Значит, нам необходимо условие «как только он будет равен 71». Представим его при помощи условного оператора if.

```
class ClassCounter  //Это класс - в котором производится счет.
  {
    public void Count()
    {
      for (int i = 0; i < 100; i++)
      {
        if (i == 71)
        {
        }
      }
    }
  }
```

Конструируем событие **event**. Определяем по методам, которые должны сработать при i=71 их **сигнатуру** (или прототип).

Сигнатура метода — это так называемая спецификация (или простыми словами «шаблон») какого-л. метода или методов. Представляет собой сочетание названия типа, который метод возвращает, плюс название типов входящих параметров (по порядку! порядок очень важен.)

Например, метод *int NewMethod(int x, char y)* будет иметь сигнатуру **int (int, char)**, а метод *void NewMethod()* — **void (void)**.

Как толкует MSDN, события (event) основаны на делегатах (delegate), а делегат, говоря очень простым языком — «переменная, хранящая ссылку на метод». Как Вы уже поняли, т.к. наше событие будет ссылаться на два метода void Message(), мы должны определить сигнатуру этих методов, и составить на основе этой сигнатуры делегат. Сигнатура выглядит так: **void (void)**.

Определяем делегат (назовем его MethodContainer):

```
class ClassCounter  //Это класс - в котором производится счет.
  {
    //Синтаксис по сигнатуре метода, на который мы создаем делегат:
    //delegate <выходной тип> ИмяДелегата(<тип входных параметров>);
    //Мы создаем на void Message(). Он должен запуститься, когда условие выполнится.

    public delegate void MethodContainer();

    public void Count()
    {
      for (int i = 0; i < 100; i++)
      {
        if (i == 71)
        {
        }
      }
    }
  }
```

Далее, мы создаем событие при помощи ключевого слова **event** и связываем его с этим делегатом (**MethodContainer**), а, следовательно, с методами, имеющими сигнатуру *void (void)*. Событие должно быть public, т.к. его должны использовать разные классы, которым нужно как-то отреагировать (классы Handler_I и Handler_II).

Событие имеет синтаксис: public event <НазваниеДелегата> <НазваниеСобытия>;
Название делегата — это имя делегата, на который «ссылаются» наши методы.

```
class ClassCounter  //Это класс - в котором производится счет.
   {
      public delegate void MethodContainer();

      //Событие OnCount с типом делегата MethodContainer.
      public event MethodContainer onCount;

      public void Count()
      {
         for (int i = 0; i < 100; i++)
         {
            if (i == 71)
            {
            }
         }
      }
   }
```

Теперь запустим наше событие onCount, в условии когда i=71:

```
if (i == 71)
{
    onCount();
}
```

Все. **Событие создано**. Методы, которые вызовет это событие, определены по сигнатурам и на основе их создан делегат.

Событие, в свою очередь, создано на основе делегата. Пора показать событию *onCount*, какие же все-таки методы должны сработать (мы ведь указали только их сигнатуру).

Подписка.

Вернемся в точку входа программы main и создадим экземпляр класса ClassCounter. А также создадим по экземпляру классов, которые должны запуститься. (Они должны быть *public*).

```
class Program
   {
```

```
    static void Main(string[] args)
    {
      ClassCounter Counter = new ClassCounter();
      Handler_I Handler1 = new Handler_I();
      Handler_II Handler2 = new Handler_II();
    }
  }
```

Теперь укажем событию *onCount*, методы, которые должны запуститься.

```
class Program
  {
    static void Main(string[] args)
    {
      ClassCounter Counter = new ClassCounter();
      Handler_I Handler1 = new Handler_I();
      Handler_II Handler2 = new Handler_II();

      //Подписались на событие
      Counter.onCount += Handler1.Message;
      Counter.onCount += Handler2.Message;
    }
  }
```

Проверка.

Теперь осталось запустить счетчик класса *ClassCounter* и подождать, пока i станет равным 71. Как только i=71, запустится событие *onCount* по делегату *MethodContainer*, который (в свою очередь) запустит методы *Message()*, которые были *подписаны* на событие.

```
class Program
  {
    static void Main(string[] args)
```

```
    {
        ClassCounter Counter = new ClassCounter();
        Handler_I Handler1 = new Handler_I();
        Handler_II Handler2 = new Handler_II();

        Counter.onCount += Handler1.Message;
        Counter.onCount += Handler2.Message;

        //Запустили счетчик
        Counter.Count();
    }
}
```

Практическая часть:

Создайте класс "Магазин", содержащий следующие поля: название магазина и количество посетителей. Создайте событие, которое будет вызываться при изменении количества посетителей в магазине. Напишите обработчик события, который будет выводить сообщение о новом количестве посетителей.

Создайте класс "Банковский счет", содержащий поля: баланс и процентная ставка. Создайте событие, которое будет вызываться при уменьшении баланса до нуля. Напишите обработчик события, который будет выводить сообщение об этом событии.

Создайте класс "Клавиатура", содержащий событие, которое будет вызываться при нажатии на клавишу. Создайте обработчик события, который будет выводить код и название нажатой клавиши.

Создайте класс "Таймер", содержащий событие, которое будет вызываться каждую секунду. Создайте обработчик события, который будет выводить текущее время.

Создайте класс "Счетчик", содержащий событие, которое будет вызываться при достижении определенного значения счетчика. Создайте обработчик события, который будет выводить сообщение о достижении значения.