Лабораторная работа №13

Тема: Разработка классов с событиями и использование их в программах.

Цель: Научиться разрабатывать классы с событиями и использовать их в программах.

Теоретические сведения:

События в .NET основаны на модели делегата. Модель делегата соответствует шаблону разработки наблюдателя, который позволяет подписчику зарегистрироваться у поставщика и получать от него уведомления. Отправитель события отправляет уведомление о событии, а приемник событий получает уведомление и определяет ответ на него.

Событие — это сообщение, посланное объектом, чтобы сообщить о совершении действия. Это действие может быть вызвано взаимодействием с пользователем, например при нажатии кнопки, или другой логикой программы, например изменением значения свойства. Объект, вызывающий событие, называется отправителем событий. Отправителю событий не известен объект или метод, который будет получать (обрабатывать) созданные им события. Обычно событие является членом отправителя событий; например, событие Click — член класса Button, а событие PropertyChanged — член класса, реализующего интерфейс INotifyPropertyChanged.

Чтобы определить событие, необходимо использовать ключевое слово Event в сигнатуре класса события и задать тип делегата для события.

Как правило, для вызова события добавляется метод, помеченный как protected и virtual. Назовите этот метод OnEventName; например, OnDataReceived. Метод должен принимать один параметр, который определяет объект данных события, являющийся объектом типа EventArgs или производного типа. Этот метод предоставляется, чтобы производные классы могли переопределять логику для вызова события. Производный класс должен вызывать метод OnEventName базового класса, чтобы зарегистрированные делегаты получили событие.

В следующем примере показан способ объявления события ThresholdReached. Событие связано с делегатом EventHandler и возникает в методе OnThresholdReached.

```
class Counter
{
    public event EventHandler ThresholdReached;

    protected virtual void OnThresholdReached(EventArgs e)
    {
```

```
        EventHandler handler = ThresholdReached;
        handler?.Invoke(this, e);
    }

    // provide remaining implementation for the class
      }
```

Данные, связанные с событием, могут быть предоставлены с помощью класса данных события. .NET предоставляет множество классов данных событий, которые можно использовать в приложениях. Например, класс SerialDataReceivedEventArgs — класс данных события SerialPort.DataReceived. В .NET имена всех классов данных событий оканчиваются ключевым словом EventArgs. Определить, какой класс данных события связан с событием, можно по делегату этого события. Например, делегат SerialDataReceivedEventHandler содержит класс SerialDataReceivedEventArgs в качестве одного из своих параметров.

Класс EventArgs является базовым типом для всех классов данных событий. Класс EventArgs используется также, если событие не содержит связанных данных. При создании события, которое лишь уведомляет другие классы о том, что что-то произошло, и не передает никаких данных, используйте класс EventArgs в качестве второго параметра в делегате. Если данные не предоставляются, можно передать значение EventArgs.Empty. Делегат EventHandler содержит класс EventArgs в качестве параметра.

Если требуется создать пользовательский класс данных события, создайте класс, производный от класса EventArgs, а затем укажите все члены, необходимые для передачи данных, связанных с событием. В большинстве случаев следует использовать схему именования .NET и завершать имя класса данных события ключевым словом EventArgs.

В следующем примере показан класс данных события с именем ThresholdReachedEventArgs. Он содержит свойства, относящиеся только к вызываемому событию.

```
public class ThresholdReachedEventArgs : EventArgs
{
    public int Threshold { get; set; }
    public DateTime TimeReached { get; set; }
      }
```

Для обработки события в приемнике события необходимо определить метод обработчика события. Этот метод должен соответствовать сигнатуре делегата обрабатываемого события. В обработчике событий выполняются действия, необходимые при возникновении события, например сбор данных, введенных пользователем при нажатии кнопки. Чтобы получать уведомления

при возникновении события, метод обработчика события должен быть подписан на событие.

В следующем примере показан метод обработчика события c_ThresholdReached, который соответствует сигнатуре делегата EventHandler. Метод подписывается на событие ThresholdReached.

```
class Program
{
    static void Main()
    {
        var c = new Counter();
        c.ThresholdReached += c_ThresholdReached;

        // provide remaining implementation for the class
    }

    static void c_ThresholdReached(object sender, EventArgs e)
    {
        Console.WriteLine("The threshold was reached.");
    }
  }
```

NET позволяет подписчикам регистрироваться для получения уведомлений о событиях как статически, так и динамически. Обработчики статических событий действуют в течение всего жизненного цикла класса, события которого они обрабатывают. Обработчики динамических событий активируются и деактивируются во время выполнения программы, обычно в ответ на определенную условную логику программы. Например, они могут использоваться, если уведомления о событиях требуются только в определенных условиях, либо приложение предоставляет несколько обработчиков событий и выбор конкретного обработчика зависит от условий среды выполнения. Если класс создает несколько событий, компилятор создает одно поле для каждого экземпляра делегата события. При большом количестве событий стоимость хранения одного поля на делегата может оказаться неприемлемой. Для таких случаев .NET предоставляет свойства события, которые можно использовать вместе с другой структурой данных для хранения делегатов события.

Свойства событий состоят из объявлений событий и методов доступа к событиям. Методы доступа к событиям — это определяемые пользователем методы, добавляющие или удаляющие экземпляры делегата события из структуры данных хранения. Обратите внимание, что использование свойств события снижает быстродействие по сравнению с полями события, поскольку перед вызовом каждого делегата события его необходимо извлечь. Необходимо найти компромисс между памятью и скоростью. Если ваш класс определяет много событий, которые вызываются нечасто, необходимо реализовать свойства событий.

Выполнение работы:

Используя среду разработки Microsoft Visual Studio, создать консольное приложение на языке программирования C#. На основе результата лабораторной работы №11 реализовать обработку событий, реагирующих на изменение данных в реализованных структурах. Реакции на событие должны быть в виде вывода сообщения на экран. Ввод данных должен производиться с клавиатуры.

Содержание отчета:

1. Номер и тема лабораторной работы.
2. Цель лабораторной работы.
3. Техническое оснащение.
4. При выполнении индивидуальных заданий в отчет внести изображение кода программы и окно выполнения программы.
5. Вывод по лабораторной работе.