События – способ описания связи одного объекта с другими по действиям. Работу с событиями можно условно разделить на три этапа:

- объявление события (*publishing*);
- регистрация получателя события (*subscribing*);
- генерация события (*raising*).

Событие можно объявить в пределах класса, структуры или интерфейса. Базовый синтаксис объявления события следующий:

<модификаторы> `event` *<тип делегата> <имя события>*`;`

Ключевое слово `event` указывает на объявление события. При объявлении события требуется указать делегат, описывающий метод обработки события. Этот делегат должен иметь тип возвращаемого значения `void`.

Фактически, события являются полями типа делегатов. При объявлении события компилятор добавляет в класс `private`-поле с именем *<имя события>* и типом *<тип делегата>*. Кроме этого, для обслуживания события компилятор создаёт два метода `add_`*Name*`()` и `remove_`*Name*`()`, где *Name* – имя события. Эти методы содержат код, добавляющий и удаляющий обработчик события в цепочку группового делегата, связанного с событием.

В блоке добавления и удаления обработчиков обычно размещается код, добавляющий или удаляющий метод в цепочку группового делегата. Поле-делегат в этом случае должно быть явно объявлено в классе.

Для генерации события в требуемом месте кода помещается вызов в формате *<имя события>*`(`*<фактические аргументы>*`)`. Предварительно обычно проверяют, назначен ли обработчик события. Генерация события может происходить только в том же классе, в котором событие объявлено[1].

Рассмотрим этап регистрации получателя события. Чтобы отреагировать на событие, его надо ассоциировать с *обработчиком события*. Обработчиком может быть метод, приводимый к типу события (делегату). В качестве обработчика может выступать анонимный метод или лямбда-оператор. Назначение/удаление обработчиков события выполняется при помощи операторов `+=` и `-=`.

Используем класс `ExampleClass` и продемонстрируем назначение и удаление обработчиков событий:

Платформа .NET предлагает средства стандартизации работы с событиями. В частности, для типов событий зарезервированы следующие делегаты:

```
public delegate void EventHandler(object sender, EventArgs e);

public delegate void EventHandler<TEventArgs>(object sender,
                                               TEventArgs e)
                                    where TEventArgs : EventArgs;
```

Как видим, данные делегаты предполагают, что первым параметром будет выступать объект, в котором событие было сгенерировано. Второй параметр используется для передачи информации события. Это либо класс `EventArgs`, либо наследник этого класса с необходимыми полями.

Сама генерация события обычно выносится в отдельный виртуальный метод класса. В этом методе проверяется, был ли установлен обработчик события. Также можно создать копию события перед обработкой (это актуально в многопоточных приложениях)

[1] Поведение, аналогичное событиям, можно получить, используя открытие поля делегатов. Ключевое слово `event` заставляет компилятор проверять, что описание и генерация события происходят в одном классе, и запрещает для события все операции, кроме `+=` и `-=`.