События

События позволяют сигнализируют системе о том, что произошло определенное действие. События объявляются в классе с помощью ключевого слова **event**, после которого идет название делегата:

```
// Объявляем делегат
public delegate void AccountStateHandler(string message);
// Событие, возникающее при выводе денег
public event AccountStateHandler Withdrawn;
```

Связь с делегатом означает, что метод, обрабатывающий данное событие, должен принимать те же параметры, что и делегат, и возвращать тот же тип, что и делегат.

Итак, посмотрим на примере. Для этого возьмем класс Account из прошлой темы и изменим его следующим образом:

```
class Account
{
    // Объявляем делегат
    public delegate void AccountStateHandler(string message);
    // Событие, возникающее при выводе денег
    public event AccountStateHandler Withdrawn;
    // Событие, возникающее при добавление на счет
    public event AccountStateHandler Added;

    int _sum; // Переменная для хранения суммы

    public Account(int sum)
    {
        _sum = sum;
    }

    public int CurrentSum
    {
        get { return _sum; }
    }

    public void Put(int sum)
    {
        _sum += sum;
        if (Added != null)
            Added($"На счет поступило {sum}");
    }
    public void Withdraw(int sum)
    {
        if (sum <= _sum)
        {
            _sum -= sum;
            if (Withdrawn != null)
                Withdrawn($"Сумма {sum} снята со счета");
        }
```

```
36          else
37          {
38              if (Withdrawn != null)
39                  Withdrawn("Недостаточно денег на счете");
40          }
41      }
42  }
```

Здесь мы определили два события: Withdrawn и Added. Оба события объявлены как экземпляры делегата AccountStateHandler, поэтому для обработки этих событий потребуется метод, принимающий строку в качестве параметра.

Затем в методах Put и Withdraw мы вызываем эти события. Перед вызовом мы проверяем, закреплены ли за этими событиями обработчики (if (Withdrawn != null)). Так как эти события представляют делегат AccountStateHandler, принимающий в качестве параметра строку, то и при вызове событий мы передаем в них строку.

Теперь используем события в основной программе:

```
1   class Program
2   {
3       static void Main(string[] args)
4       {
5           Account account = new Account(200);
6           // Добавляем обработчики события
7           account.Added += Show_Message;
8           account.Withdrawn += Show_Message;
9
10          account.Withdraw(100);
11          // Удаляем обработчик события
12          account.Withdrawn -= Show_Message;
13
14          account.Withdraw(50);
15          account.Put(150);
16
17          Console.ReadLine();
18      }
19      private static void Show_Message(string message)
20      {
21          Console.WriteLine(message);
22      }
23  }
```

Для прикрепления обработчика события к определенному событию используется операция += и соответственно для откреплния - операция -=: событие += метод_обработчика_события. Опять же обращаю внимание, что метод обработчика должен иметь такие же параметры, как и делегат события, и возвращать тот же тип. В итоге мы получим следующий консольный вывод:

```
Сумма 100 снята со счета
На счет поступило 150
```

Кроме использованного выше способа прикрепления обработчиков есть и другой с использованием делегата. Но оба способа будут равноценны:

```
1   account.Added += Show_Message;
2   account.Added += new Account.AccountStateHandler(Show_Message);
```

Класс данных события AccountEventArgs

Нередко при возникновении события обработчику события требуется передать некоторую информацию о событии. Например, добавим и в нашу программу новый класс *AccountEventArgs* со следующим кодом:

```
1    class AccountEventArgs
2    {
3        // Сообщение
4        public string Message{get;}
5        // Сумма, на которую изменился счет
6        public int Sum {get;}
7
8        public AccountEventArgs(string mes, int sum)
9        {
10           Message = mes;
11           Sum = sum;
12       }
13   }
```

Данный класс имеет два свойства: Message - для хранения выводимого сообщения и Sum - для хранения суммы, на которую изменился счет.

Теперь применим класс AccoutEventArgs, изменив класс Account следующим образом:

```
1    class Account
2    {
3        // Объявляем делегат
4        public delegate void AccountStateHandler(object sender,
5    AccountEventArgs e);
6        // Событие, возникающее при выводе денег
7        public event AccountStateHandler Withdrawn;
8        // Событие, возникающее при добавлении на счет
9        public event AccountStateHandler Added;
10
11       int _sum; // Переменная для хранения суммы
12
13       public Account(int sum)
14       {
15           _sum = sum;
16       }
17
18       public int CurrentSum
19       {
20           get { return _sum; }
21       }
22
23       public void Put(int sum)
24       {
25           _sum += sum;
26           if (Added != null)
27               Added(this, new AccountEventArgs($"На счет поступило {sum}",
28   sum));
```

```
29      }
30      public void Withdraw(int sum)
31      {
32          if (_sum >= sum)
33          {
34              _sum -= sum;
35              if (Withdrawn != null)
36                  Withdrawn(this, new AccountEventArgs($"Сумма {sum} снята со
37  счета", sum));
38          }
39          else
40          {
41              if (Withdrawn != null)
42                  Withdrawn(this, new AccountEventArgs("Недостаточно денег на
    счете", sum));
            }
        }
    }
```

По сравнению с предыдущей версией класса Account здесь изменилось только количество параметров у делегата и соответственно количество параметров при вызове события. Теперь они также принимают объект AccountEventArgs, который хранит информацию о событии, получаемую через конструктор.

Теперь изменим основную программу:

```
1   class Program
2   {
3       static void Main(string[] args)
4       {
5           Account account = new Account(200);
6           // Добавляем обработчики события
7           account.Added += Show_Message;
8           account.Withdrawn += Show_Message;
9
10          account.Withdraw(100);
11          // Удаляем обработчик события
12          account.Withdrawn -= Show_Message;
13
14          account.Withdraw(50);
15          account.Put(150);
16
17          Console.ReadLine();
18      }
19      private static void Show_Message(object sender, AccountEventArgs e)
20      {
21          Console.WriteLine($"Сумма транзакции: {e.Sum}");
22          Console.WriteLine(e.Message);
23      }
24  }
```

По сравнению с предыдущим вариантом здесь мы только изменяем количество параметров и сущность их использования в обработчике Show_Message.