СЕДЬМОЕ ЗАНЯТИЕ

Делегаты

- Кроме свойств и методов классы могут содержать делегаты и события. Делегаты представляют такие объекты, которые указывают на другие методы. То есть делегаты - это указатели на методы. С помощью делегатов мы можем вызвать определенные методы в ответ на некоторые произошедшие действия. То есть, по сути, делегаты раскрывают нам функционал функций обратного вызова.

Примеры делегатов

Делегат определяет то, какие параметры он принимает, и что именно возвращает. Далее мы можем присваивать переменным типа нашего делегата методы с такой же сигнатурой

```
delegate int Operation(int x, int y);
delegate void GetMessage();
```

```
class Program {
    delegate void GetMessage(); // 1. Объявляем делегат
    static void Main(string[] args)  {
        GetMessage del; // 2. Создаем переменную делегата
        if (DateTime.Now.Hour < 12)   {
            del = GoodMorning; // 3. Присваиваем этой переменной адрес метода
        }  else  {
            del = GoodEvening;
        }
        del.Invoke(); // 4. Вызываем метод
        Console.ReadLine();
    }
    private static void GoodMorning()  {
        Console.WriteLine("Good Morning");
    }
    private static void GoodEvening() {
        Console.WriteLine("Good Evening");
    }
}
```

События

События используются для выполнения определенного когда при наступлении некоторого события. Это лишь обертки над делегатами, но очень удобные

Мы можем подписывать на событие неограниченное количество обработчиков, при помощи +=, при необходимости, мы можем отписать обработчик, используя -=

```
delegate void SampleDelegaet();
event SampleDelegate SampleEvent();
//имеется метод
void SomeAction()
{ Console.WriteLine("Some");}
//наш код
MyType v = new MyType();
v.SampleEvent += SomeAction;
//Теперь, когда будет необходимо, метод SomeAction будет вызван, и нам не надо про это думать.
```

Анонимные методы

При подписке на событие или передаче делегата не всегда удобно писать непосредственный код в отдельных методах. Мы можем написать необходимый код прямо на месте

В скобках необходимо указывать имена принимаемых делегатом параметров, далее в фигурных скобках непосредственно исполняемый код

```
v.SampleEvent += delegate()
{
    Console.WriteLine("Anon method");
}
```

Лямбды

- Лямбда-выражения представляют упрощенную запись анонимных методов. Лямбда-выражения позволяют создать емкие лаконичные методы, которые могут возвращать некоторое значение и которые можно передать в качестве параметров в другие методы.
- Ламбда-выражения имеют следующий синтаксис: слева от лямбда-оператора => определяется список параметров, а справа блок выражений, использующий эти параметры: (список_параметров) => выражение.

ПРИМЕР ПРОСТОЙ ЛЯМБДЫ

```
class Program
{
    delegate int Square(int x); // объявляем делегат,принимающий int и возвращающий int
    static void Main(string[] args)
    {
        Square squareInt = i => i * i; // объекту делегата присваивается лямбда-выражение

        int z = squareInt(6); // используем делегат
        Console.WriteLine(z); // выводит число 36
        Console.Read();
    }
}
```

Action, Func

- В C# имеются уже определенные обобщенные делегаты, которые мы можем использовать, не прибегая к написанию собственных.
- Так, Делегат Action<T> определяет делегат, которые ничего не возвращает, но принимает параметр типа T. Параметром может быть несколько Action<T1, T2>
- А делегат Func используется для возвращения некоторого значения. Тип возвращаемого значения указывается последним. Func<TResult> ничего не принимает, возвращает TResult. А Func<T1, T2, TResult> принимает T1 и T2, и возвращает TResult.

МНОГОПОТОЧНОСТЬ