匿名函数和委托有着密切的关系

什么是委托：（委托就相当于创建一个类，将到时候需要使用的函数存入其中。将相当于你去银行存钱，你就把钱委托给了柜员让他们去执行存钱的这个函数。）

委托是一种引用类型。在C#中，委托是一种可以把引用存储为函数的类型。委托可以引用实例和静态方法，而函数指针只能引用静态方法。

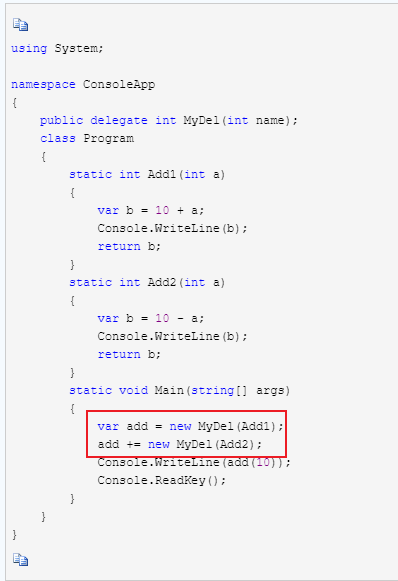
在定义了委托后，就可以声明该委托类型的变量，

然后可以将这个变量初始化为与该委托有相同签名（包含参数列表和一个返回类型）的函数进行引用，

随后使用委托变量调用该函数。（委托本质就是一个类，他的目的就是将方法可以作为其他方法的参数传递。）

当delegate绑定实例方法的时候，delegate的target属性就被设置为指向这个实例方法所属类型的一个实例对象。当绑定静态方法时，target属性就个null。

多播委托（一个委托可以看成是一个类，所以当有多个调用方法时，就可以使用多播委托）这个委托执行后，其中的每个方法都会调用



这里的 += 不再是之前的+=赋值运算符，而是一个添加引用委托方法的重载。

1. 为什么要使用匿名函数，什么时候用？

当用常规的委托调用方法时使代码混乱或者不简洁的时候，可以使用匿名方法（内联代码块定义）。就是说当我们用委托调用方法时，是根据方法名调用的，当需要待用的方法很多时，我们创建很多的方法，但是很多方法又不常用，所以我们可以采用匿名方法。（当某些方法比较简单时就可以直接携程匿名函数，而不需要在去创建一个函数在传递给委托。）

1. 匿名函数总是以一个delegate关键词开始，后面跟着用在方法和方法体本身中的参数。用户不需要确定匿名方法的返回类型。它（返回类型）由方法体中的return语句推断而来。.NET CLR不能执行像匿名方法一样的自由流（free flowing）代码块。CLR要求：它执行的每个方法是一个类型的一部分，并且应该是一个静态（static）方法或实例（instance）方法。因此当你在一个类的代码中写匿名方法并编译这个代码时，C#编译程序默默地在你定义匿名方法的相同的类中创建了一个静态或实例方法。所以匿名方法只是一个在类中定义你自己方法以传递到委托的方便的语法。

若一个方法声明中含有static修饰符，则称该方法为静态方法。若没有static修饰符时，则称该方法为实例方法。静态方法不对特定实例进行操作，在静态方法中引用this是编译时错误。实例方法对类的某个给定的实例进行操作，而且可以用this来访问该实例。

闭包：大概是当一个函数中（外部函数）调用了另一个函数（内部函数）时，当内部函数使用了外部函数中的变量时，这样就可能形成闭包。

当委托的方法没有参数，使用匿名方法来构造函数实例时就很方便。具体代码如下：



虽然我们使用了匿名方法来省略委托参数，但是对于编译器而言，它还是会调用委托的构造函数来实例化委托（委托实例方法时），所以如果匿名方法能转换为多个委托类型时，此时如果省略了委托函数，编译器就不知道把匿名方法转化为哪个具体的委托类型，此时就会出现编译错误。

进入timer类中可以看到Elapsed就是一个Eventhandler（事件处理器）。即事件委托，表示用于处理不具有事件数据的事件方法。(将一个事件与处理事件的方法联系到一起的机制)

C#中的事件基于委托模型。**委托遵循观察者设计模式，是订阅者（接收或处理事件的类）能够向提供方（发送或引发事件的类）注册并接收相关通知**。

相当于两个箱子，一个发产品，一个接收产品。

首先声明一个委托：

// 这些委托没有返回类型值，并且接受两个参数（事件源的对象和事件数据的对象）。

public delegate void NameEventHandler(object sender, NameEventArgs e);

然后声明一个事件，并指定事件的委托类型：

// event关键字代表事件，返回类型为委托(事件与委托绑定)；

public event EventHandler Name（事件名）;

同时，创建引发事件的方法，一般命名为：OnEventName：

// 通常为了引发事件，会添加一个标记为protected和virtual的方法，以允许派生类重写引发事件的逻辑。

protected virtual void OnName(EventArgs e)

{

EventHandler handler = ThresholdReached;

if (handler != null)

{

handler(this, e);

}

}

在定义一个事件处理程序方法，处理事件

static void c\_ThresholdReached(object sender, EventArgs e)

{

Console.WriteLine("The threshold was reached.");

}

最后使用加法赋值 运算法（+=）来为事件附加事件处理程序。+=表示添加事件，-=表示删除事件：

// Counter类中需要拥有一个名为ThresholdReached的事件

Counter c = new Counter();

c.Name += c\_Name;

// 以下是老的C#1.0的语法

// c.ThresholdReached += new ThresholdReachedEventHandler(c\_ThresholdReached);



EventHandler就是等待某事件发生，一旦事件发生则执行设置的方法。

在匿名方法中捕捉变量

一个变量被捕捉后，被匿名方法捕捉到的是真的变量，而不是创建委托实例时该变量的值，并且被匿名方法中捕捉到的变量会延长生命周期。