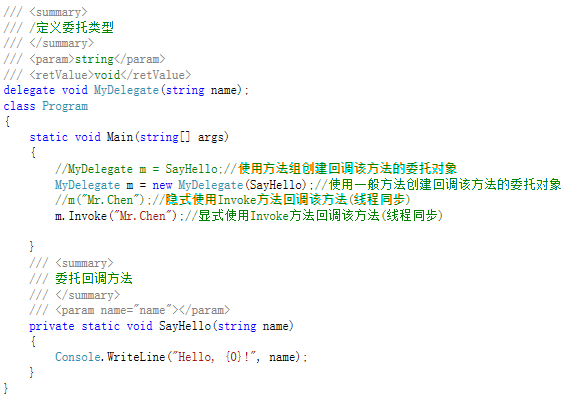
**delegate委托**

**delegate委托用于自下而上的回调callback，即从程序中创建的对象内部回调程序中定义的方法，其经典的案例是event事件回调。**

* **区分回调callback和调用invoke机制：**

|  |  |  |
| --- | --- | --- |
| 类型 | 方向 | 案例 |
| 回调callback | 至下而上 | 委托事件event |
| 调用invoke | 至上而下、同一级别 | 调用对象的方法、类的静态方法 |

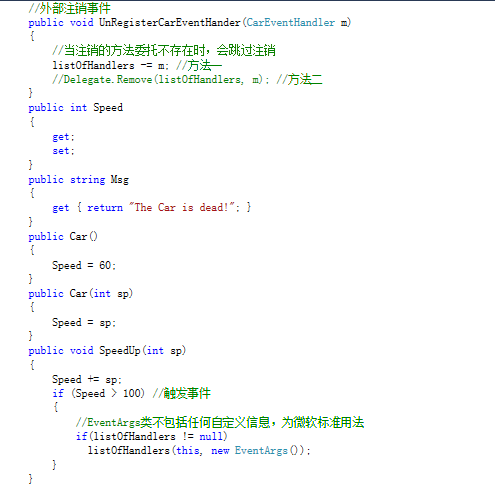
* **委托的最基本用法**

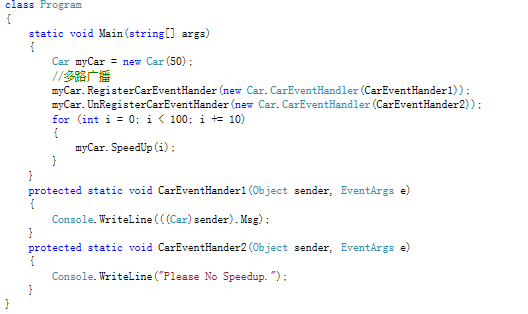
****

* **多路广播**

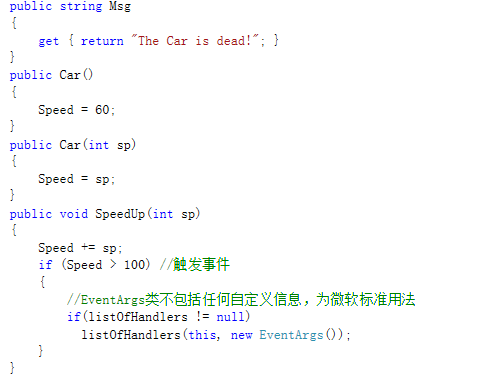
多路广播是指一个委托对象可以同时回调多个不同方法，而且回

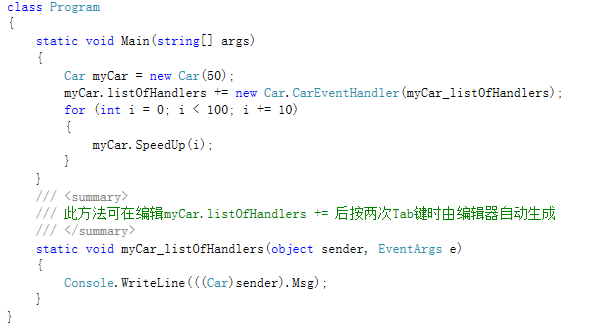
调时将按注册顺序执行这些方法。

****



* **event关键字简化委托事件**





* **C#匿名方法**

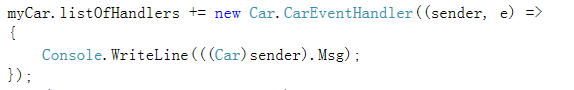
以上代码中，可用匿名方法（内联）代替回调方法的定义。



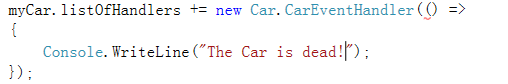
**注意：匿名方法中的变量不可和外部变量同名，但可以与对象属性同名。同时，匿名方法可以访问外部变量。**

* **Lambda表达式**

Lambda表达式是匿名方法的进一步简化。



若无形参，则Lambda表达式写为 ()=>{ 方法体 }。

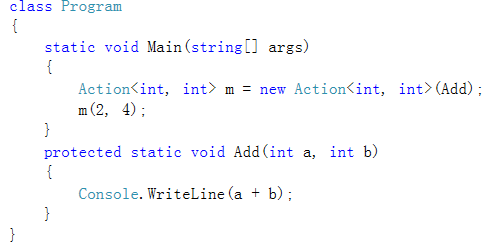


同时，Lambda表达式常用于Linq查询。

* **常用泛型委托 Func<>、Action<>、EventHander<>**

**Func<paramsT, retValT >**中，最后一个参数为返回值

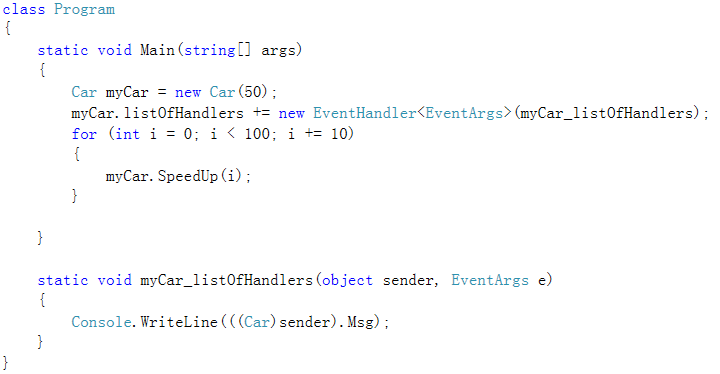
**Action<paramsT>**无返回值



**EventHander< EventArgs >**泛型可以用于event事件委托的简化。

之所以这样做，是因为普遍的事件委托定义以**(Object sender, EventArgs e)** 为形参，而Object sender一般固定，因此以上代码可以简写为：





泛型可以省略对委托类型的定义，即可以省略以下代码：

