Визуальное программирование

ЛЕКЦИЯ 5

Содержание лекции

01 Делегаты

02 Анонимные функции

03 Мультикаст делегаты

04 Стандартные делегаты

ДЕЛЕГАТЫ

Делегаты

Делегат — это тип, который представляет ссылки на методы с определенным списком параметров и типом возвращаемого значения.

При создании экземпляра делегата этот экземпляр можно связать с любым методом с совместимой сигнатурой и типом возвращаемого значения.

Метод можно вызвать (активировать) с помощью экземпляра делегата.

Делегаты используются для передачи методов в качестве аргументов к другим методам. Делегату можно назначить любой метод из любого доступного класса или структуры, соответствующей типу делегата.

Свойства делегатов

- позволяют обрабатывать методы в качестве аргумента;
- могут быть связаны вместе;
- несколько методов могут быть вызваны по одному событию;
- тип делегата определяется его именем;
- не зависит от класса объекта, на который ссылается;
- сигнатура метода должна совпадать с сигнатурой делегата.

Синтаксис

Объявление

```
modifier delegate ReturnType DelegateName ([Parameter_1]);
```

Инициализация

```
DelegateName DelegateObjectName = new DelegateName(MethodName);
```

Вызов

```
DelegateObjectName([Parameter_1]);
```

Вызов делегата

После того, как был создан экземпляр делегата, можно выполнить вызов этого делегата как обычного метода. Этот вызов будет передаваться (делегироваться) методу или нескольким методам, которые скрыты за этим делегатом.

Параметры, передаваемые делегату вызывающим кодом, передаются в метод(-ы), а возвращаемое методом значение (при его наличии) возвращается делегатом обратно в вызывающий код.

Пример

```
public delegate void MyDelegate(string customString);


public static void PrintMessage(string message)
{
	Console.WriteLine($"From {nameof(PrintMessage)}: {message}");
}


// Инициализируем делегат
MyDelegate delegateInstance = PrintMessage;

// Вызываем делегат
delegateInstance("Hello World!");
```

Ковариантность и контравариантность делегатов

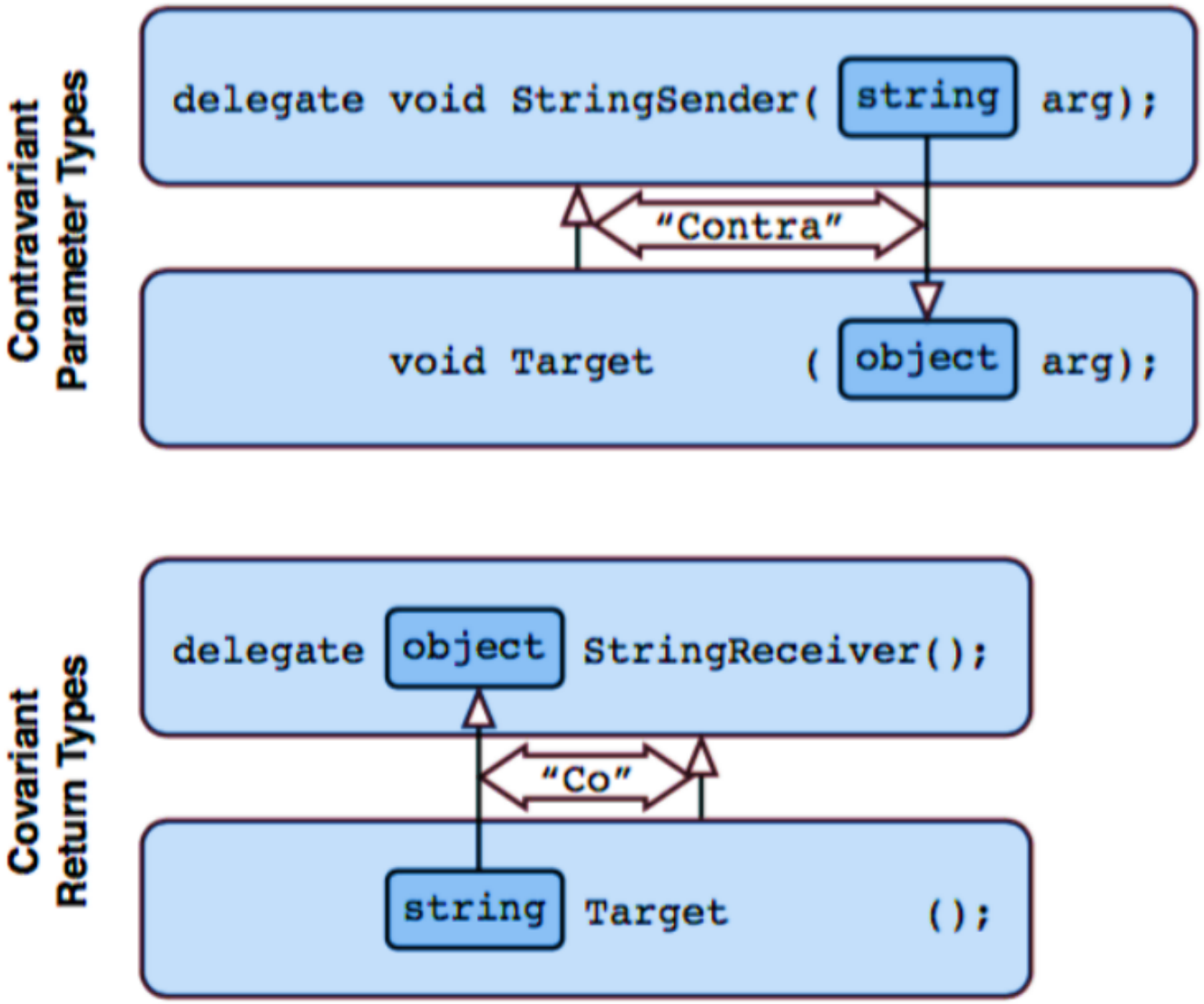
Делегаты могут быть ковариантными и контравариантными.

Ковариантность делегата предполагает, что возвращаемым типом может быть производный тип.

Контравариантность делегата предполагает, что типом параметра может быть более универсальный тип.

Обобщенные делегаты также могут быть ковариантными (ключевое слово out) и контравариантными (ключевое слово in).

Ковариантность и контравариантность



АНОНИМНЫЕ ФУНКЦИИ

Анонимные функции

Анонимная функция — это "встроенный" оператор или выражение, которое может использоваться, когда тип делегата неизвестен. Ее можно использовать для инициализации именованного делегата или передать вместо типа именованного делегата в качестве параметра метода.

Для создания анонимной функции можно использовать лямбда-выражение или анонимный метод.

Лямбда-выражения

Лямбда-выражения могут записываться в одной из следующих форм:

- (входные параметры) => выражение, когда требуется выполнить только одну строчку кода
- (входные параметры) => { блок инструкций }, когда надо выполнить несколько действий

Лямбда-выражения

```
delegate int Operation(int x, int y);
Operation sum = (x, y) => x + y;
int result = sum(5, 10);
//C# v.10
var mul = (int x, int y) => x * y;

Operation comp = (x, y) =>
{
if (x > y) return 1;
else if (x == y) return 0;
else return -1;
};
```

Лямбда-выражения

```
delegate int UnaryOperation(int x);
UnaryOperation square = x => x * x;

delegate int MethodWithoutArguments();
MethodWithoutArguments getYear = () => DateTime.Now.Year;
```

Анонимные методы

Для создания анонимного метода используется оператор delegate.

```
Operation sum = delegate (int x, int y) { return x + y; };
UnaryOperation square = delegate (int x) { return x * x; };
MethodWithoutArguments getYear = delegate () { return DateTime.Now.Year; };
```

МУЛЬТИКАСТ-ДЕЛЕГАТЫ

Мультикаст-делегаты

Делегаты, включающие в себя более одного метода, называются мультикаст-делегатами. При вызове они выполняют каждый метод в заданном порядке, позволяя таким образом связывать несколько методов в цепочку.

Для работы мультикаст-делегатов те не должны возвращать какой-либо результат. В противном случае обработается результат последнего метода цепочки.

Мультикаст-делегаты

Выражение	Результат
null + d1	d1
d1 + null	d1
d1 + d2	[d1, d2]
d1 + [d2, d3]	[d1, d2, d3]
[d1, d2] + [d2, d3]	[d1, d2, d2, d3]
[d1, d2] – d2	d1
[d1, d2] – d1	d2
[d1, d2, d1] – d1	[d1, d2]
[d1, d2, d3] – [d1, d2]	d3
[d1, d2, d3] – [d2, d1]	[d1, d2, d3]
[d1, d2, d3, d1, d2] – [d1, d2]	[d1, d2, d3]
[d1, d2] – [d1, d2]	null

System.MulticastDelegate

Когда компилятор C# обрабатывает тип делегата, он автоматически генерирует запечатанный класс, унаследованный от System.MulticastDelegate.

Этот класс (в сочетании с его базовым классом System.Delegate) предоставляет необходимую инфраструктуру для делегата, чтобы хранить список методов, подлежащих вызову в более позднее время.

System.MulticastDelegate

Сгенерированный компилятором класс определяет три общедоступных метода.

Invoke() —используется для синхронного вызова каждого из методов, поддерживаемых объектом делегата; это означает, что вызывающий код должен ожидать завершения вызова, прежде чем продолжить свою работу.

Методы BeginInvoke() и EndInvoke() предлагают возможность вызова текущего метода асинхронным образом, в отдельном потоке выполнения.

СТАНДАРТНЫЕ ДЕЛЕГАТЫ

Action<T> и Func<T>

Тип делегата Action является стандартным делегатом действия без возвращаемого результата, следовательно, типом возвращаемого значения методов, на которые может ссылаться делегат такого типа, является void. Может принимать до 16 входных параметров.

Тип делегата Action<T> можно было бы переписать в следующем представлении:

```
public delegate void Action<T>(T value);
```

Action<T> и Func<T>

Тип делегата Func предусматривает обязательное выходное значение. Может принимать до 16 входных параметров. Тип делегата Func<T, TResult> можно было бы переписать в следующем представлении:

```
public delegate TResult Func<T,TResult>(T value);
```

Тип, который указывается последним в угловых скобках, является типом возвращаемого значения для данного делегата.

Action<T> и Func<T>

Когда можно и нужно создавать собственные типы делегатов:

- количество входных параметров более 16;
- ключевые слова out, ref, params и значения по умолчанию для входных параметров;
- обозначение смысловой нагрузки и переиспользование данного типа делегатов.

Predicate<T>

Также, в .NET предопределен тип делегата Predicate<T>, который применяется в некоторых стандартных методах, например Array.Find. Однако, он редко применяется в промышленной разработке.

ВОПРОСЫ ПО ЛЕКЦИИ