

13-ma'ruza. Delegatlar. Lyambda ifodalar

- ❖ Delegatlarni aniqlash
- ❖ Metodga havolani o'zlashtirish
- ❖ Metodlarni delegatga qo'shish
- ❖ Delegatlarni birlashtirish
- ❖ Delegatga murojaat
- ❖ To'ldirilgan delegatlar

Kalit so'zlar: *DateTime.Now.Hour, delegate, delegatga murojaat, delegatlarni birlashtirish, havola, int, ko'rsatgich, metod qo'shish, metod, modifikator, murojaat, null, out, o'zgaruvchi, parametr, ref, tip*

Delegatlar metodlarni ko'rsatadigan ob'yektlarni ifodalaydi. Ya'ni delegatlar – metodlarga ko'rsatkich bo'lib, ular yordamida metod ma'lumotlariga murojaat qilish mumkin.

Delegatlarni aniqlash

Delegatlarni e'lon qilish uchun delegate kalit so'zi qo'llaniladi, ushbu kalit so'zdan so'ng qaytariladigan tip, nom va parameter keltiriladi. Masalan:

```
delegate void Message();
```

Message delegati qaytariluvchi tip sifatida void (ya'ni hech narsa qaytarmaydigan) tipga ega va hech qanday parametr qabul qilmaydi. Bu shuni anglatadiki, bu delegat hech qanday parametr qabul qilmaydigan va hech narsa qaytarmaydigan istalgan metodni ko'rsatishi mumkin.

Ushbu delegatning qo'llanilishini qarab chiqaylik:

```
class Program
{
    delegate void Message(); // 1. Delegatni e'lon qilamiz

    static void Main(string[] args)
    {
        Message mes; // 2. Delegat o'zgaruvchisini yaratamiz
        if (DateTime.Now.Hour < 12)
        {
            mes = GoodMorning; // 3. Bu o'zgaruvchiga metod manzilini o'zlashtiramiz
        }
        else
        {
            mes = GoodEvening;
        }
    }
}
```

```

    mes(); // 4. Metodni chaqiramiz
    Console.ReadKey();
}
private static void GoodMorning()
{
    Console.WriteLine("Good Morning");
}
private static void GoodEvening()
{
    Console.WriteLine("Good Evening");
}
}

```

Bu yerda biz dastlab delegatni aniqlab olamiz:

```
delegate void Message(); // 1. Delegatni e'lon qilamiz
```

Bu holatda delegat sinf ichida aniqlanmoqda, lekin delegatni sinfdan tashqarida nomlar sohasi ichida ham aniqlash mumkin.

Delegatdan foydalanish uchun ushbu delegatning o'zgaruvchisi e'lon qilinadi:

```
Message mes; // 2. Delegat o'zgaruvchisini yaratamiz
```

DateTime.Now.Hour xossasi yordamida joriy soatni aniqlab olamiz. Vaqt bilan bog'liq holda delegatga ma'lum bir metod manzili uzatiladi. E'tibor qiling, metodlar ham delegatlar qaytaradigan qiymat va parametrlar to'plami (bu holatda parametrlar mavjud emasligi) ga ega bo'ladi.

```
mes = GoodMorning; // 3. Bu o'zgaruvchiga metod manzilini o'zlashtiramiz
```

So'ngra delegat orqali shu delegat yo'naltirilgan metod chaqiriladi:

```
mes(); // 4. Metodni chaqiramiz
```

Delegatga murojaat metodga murojaat kabi bo'ladi.

Boshqa delegatga misolni qarab chiqaylik:

```

class Program
{
    delegate int Operation(int x, int y);

    static void Main(string[] args)
    {
        // metod manzilini konstruktor orqali o'zlashtirish
        Operation del = Add; // delegat Add metodini ko'rsatmoqda
        int result = del(4, 5); // Add(4, 5)
        Console.WriteLine(result);
    }
}

```

```

        del = Multiply; // delegat Multiply metodini ko'rsatmoqda
        result = del(4, 5); // Multiply(4, 5)
        Console.WriteLine(result);
        Console.Read();
    }
    private static int Add(int x, int y)
    {
        return x+y;
    }
    private static int Multiply (int x, int y)
    {
        return x * y;
    }
}

```

Mazkur holatda Operation delegati int tipidagi qiymat qaytaradi va ikkita int tipidagi parametrga ega. Shu sababli bu delegatga int tipini qaytaruvchi va int tipidagi ikkita parametrga ixtiyoriy metod mos keladi. Ushbu holatda bular Add va Multiply metodlari. Ya'ni delegat o'zgaruvchisiga bu metodlardan ixtiyoriy bittasini o'zlashtirishimiz va chaqirishimiz mumkin.

Delegat int tipidagi ikkita parametr qabul qilganligi sababli, uni chaqirishda bu parametrlarga qiymat uzatish lozim: `del(4, 5)`.

Delegatlar albatta delegat o'garuvchisi aniqlangan sinfda aniqlangan metodni ko'rsatishi shart emas. Ular boshqa sinf va struktura metodlari ham bo'lishi mumkin.

```

class Math
{
    public int Sum(int x, int y) { return x + y; }
}
class Program
{
    delegate int Operation(int x, int y);

    static void Main(string[] args)
    {
        Math math = new Math();
    }
}

```

```
        Operation del = math.Sum;
        int result = del(4, 5);           // math.Sum(4, 5)
        Console.WriteLine(result);      // 9
        Console.Read();
    }
}
```

Metodga havolani o‘zlashtirish

Yuqorida delegat o‘zgaruvchisiga metod to‘g‘ridan-to‘g‘ri o‘lashtirilgan. Yana bir usul mavjud – konstruktor yordamida kerakli metod uzatiladigan delegat ob’yektini yaratish usuli:

```
class Program
{
    delegate int Operation(int x, int y);

    static void Main(string[] args)
    {
        Operation del = Add;
        Operation del2 = new Operation(Add);

        Console.Read();
    }
    private static int Add(int x, int y) { return x + y; }
}
}
```

Ikkala usul ham bir xil qiymatga ega.

Metodlarning delegatlarga mosligi

Yuqorida yozilganidek, agar qaytaruvchi qiymatlari bir xil tipda bo‘lsa va parametrlar to‘plami bir xil bo‘lsa, metodlar delegatga mos bo‘ladi. Lekin shuni hisobga olish keraakki, `ref` va `out` modifikatorlari ham qo‘llaniladi. Masalan, bizga quyidagi delegat berilgan:

```
delegate void SomeDel(int a, double b);
```

Ushbu delegatga quyidagi metod mos keladi:

```
void SomeMethod1(int g, double n) { }
```

Quyidagi metodlar esa mos kelmaydi:

```
int SomeMethod2(int g, double n) { }  
void SomeMethod3(double n, int g) { }  
void SomeMethod4(ref int g, double n) { }  
void SomeMethod5(out int g, double n) { g = 6; }
```

Bu yerda `SomeMethod2` metodi delegatda farqli qaytariluvchi tipga ega. `SomeMethod3` boshqa parametrlar to'plamiga ega. `SomeMethod4` va `SomeMethod5` metodlarining parametrlari ham delegat parametrlaridan farq qiladi, chunki `ref` va `out` modifikatorlariga ega.

Metodlarni delegatga qo'shish

Yuqoridagi misollarda delegat o'zgaruvchisi bitta metodni ko'rsatadi. Aslida delegat bir xil signatura va qaytaruvchi tipga ega bir nechta metodlarni ko'rsatishi mumkin. Delegatdagi barcha metodlar maxsus ro'yxat – murojaat ro'yxati yoki `invocation list` ga tushadi. Delegat chaqirilganida barcha metodlar bu ro'yxatdan ketma-ket chaqiriladi. Biz bu ro'yxatga bitta emas, balki bir nechta metodlarni qo'shishimiz mumkin:

```
class Program  
{  
    delegate void Message();  
    static void Main(string[] args)  
    {  
        Message mes1 = Hello;  
        mes1 += HowAreYou; // endi mes1 ikkita metodni ko'rsatmoqda  
        mes1(); // ikkita metod - Hello va HowAreYou chaqiriladi  
        Console.Read();  
    }  
    private static void Hello()  
    {  
        Console.WriteLine("Hello");  
    }  
    private static void HowAreYou()  
    {  
        Console.WriteLine("How are you?");  
    }  
}
```

Mazkur holatda mes1 delegati murojaat ro'yxatiga ikkita – Hello va HowAreYou metodlari qo'shilgan. mes1 ga murojaat qilinganida ikkala metod ham chaqiriladi.

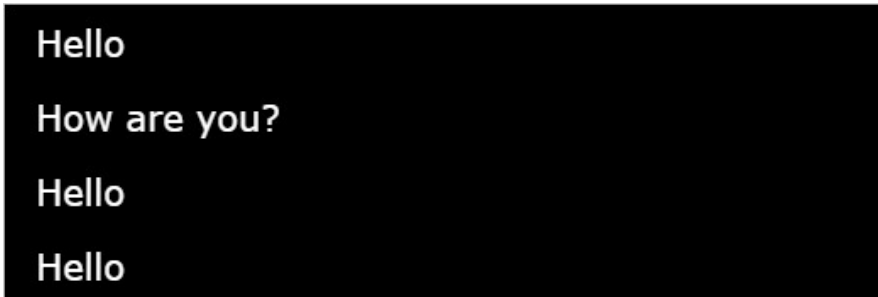
Delegatlarni qo'shishda += amali qo'llaniladi. Aslida delegatning yangi ob'yektini yaratish sodir bo'ladi va delegatning yangi yaratilgan ob'yektiga mes1 o'zgaruvchisi o'zlashtiriladi.

Delegatlarni qo'shishda bitta metodga havolani bir necha marta qo'shish mumkinligini ham e'tiborga olish lozim. Mos ravishda delegatga murojaat bo'lganida qo'shilgan metod necha marta qo'shilgan bo'lsa, shuncha marta chaqiriladi:

```
Message mes1 = Hello;
mes1 += HowAreYou;
mes1 += Hello;
mes1 += Hello;

mes1();
```

Konsoldagi natija:



```
Hello
How are you?
Hello
Hello
```

Shu tariqa delegatdan metodni o'chirish uchun -= amali qo'llaniladi:

```
static void Main(string[] args)
{
    Message mes1 = Hello;
    mes1 += HowAreYou;
    mes1(); // mes1 dan barcha metodlar chaqiriladi
    mes1 -= HowAreYou; // HowAreYou metodi o'chiriladi
    mes1(); // Hello metodi chaqiriladi
    Console.Read();
}
```

Delegatdan metodni o'chirishda metodlarga murojaat ro'yxatida bitta metod kam bo'lgan yangi delegat yaratiladi.

O‘chirishda shuni hisobga olish kerakki, agar delegat bitta metodga bir nechta havolaga ega bo‘lsa, -= amali delegatga murojaat ro‘yxatining oxiridan qidiruvni boshlaydi va faqat birinchi topilgan moslikni o‘chiradi. Agar delegatning murojaat ro‘yxatida bunday metod mavjud bo‘lmasa, -= amali hech qanday effektga ega bo‘lmaydi.

Delegatlarni birlashtirish

Delegatlarni boshqa delegatga birlashtirish mumkin. Masalan:

```
class Program
{
    delegate void Message();
    static void Main(string[] args)
    {
        Message mes1 = Hello;
        Message mes2 = HowAreYou;
        Message mes3 = mes1 + mes2; // delegatlarni birlashtirish
        mes3(); // mes1 va mes2 dan barcha metodlar chaqiriladi
        Console.Read();
    }
    private static void Hello()
    {
        Console.WriteLine("Hello");
    }
    private static void HowAreYou()
    {
        Console.WriteLine("How are you?");
    }
}
```

Ushbu holatda mes3 ob'yekti mes1 va mes2 delegatlari birlashmasini ifodalaydi. Delegatlarni birlashtirish mes3 delegati murojaat ro‘yxatiga mes1 va mes2 delegatlarining barcha metodlari tushishini bildiradi. mes3 delegatiga murojaat qilinganida ushbu metodlar ham chaqiriladi.

Delegatga murojaat

Yuqoridagi misollarda delegat oddiy metod chaqirilgani kabi chaqirildi. Agar delegat parameter qabul qilgan bo‘lsa, u holda unga murojaat qilinganida parametrlar uchun kerakli qiymatlar uzatildi:

```
class Program
```

```
{
    delegate int Operation(int x, int y);
    delegate void Message();
    static void Main(string[] args)
    {
        Message mes = Hello;
        mes();
        Operation op = Add;
        op(3, 4);
        Console.Read();
    }
    private static void Hello()
    { Console.WriteLine("Hello"); }
    private static int Add(int x, int y) { return x + y; }
}
```

Delegatni chaqirishning boshqa bir usuli Invoke() metodini tasvirlaydi:

```
class Program
```

```
{
    delegate int Operation(int x, int y);
    delegate void Message();
    static void Main(string[] args)
    {
        Message mes = Hello;
        mes.Invoke();
        Operation op = Add;
        op.Invoke(3, 4);
        Console.Read();
    }
    private static void Hello()
    { Console.WriteLine("Hello"); }
    private static int Add(int x, int y) { return x + y; }
}
```

Agar delegat parametrlar qabul qilsa, u holda Invoke metodiga bu parametrlar uchun qiymatlar uzatiladi.

Shuni hisobga olish kerakki, agar delegat bo'sh bo'lsa, ya'ni uning murojaat ro'yxatida biror bir metodga havola mavjud bo'lmasa (delegat Null ga teng), u holda bunday delegatga murojaat qilinganida istisno yuzaga keladi, masalan quyidagi holatda:

```
Message mes = null;
//mes();    //! Xatolik: delegat null ga teng

Operation op = Add;
op -= Add;    // op delegate bo'sh
op(3, 4);    //!Xatolik:: delegat null ga teng
```

Shu sababli delegatga murojaat vaqtida har doim uning null emasligini tekshirish ma'qulroq. Yoki Invoke metodi va null ga tekshirish operatorini qo'llash mumkin:

```
Message mes = null;
mes?.Invoke();    //xatolik yo'q, shunchaki, delegat chaqirilmagan
Operation op = Add;
op -= Add;    // op delegat bo'sh
op?.Invoke(3, 4);    //xatolik yo'q, shunchaki, delegat chaqirilmagan
```

Agar delegat qandaydir qiymat qaytarsa, u holda murojaat ro'yxatidagi oxirgi metod qiymati qaytariladi (agar murojaat ro'yxatida bir nechta metod bo'lsa). Masalan:

```
class Program
{
    delegate int Operation(int x, int y);
    static void Main(string[] args)
    {
        Operation op = Subtract;
        op += Multiply;
        op += Add;
        Console.WriteLine(op(7, 2));    // Add(7,2) = 9
        Console.Read();
    }
}
```

```
private static int Add(int x, int y)
{ return x + y; }
private static int Subtract(int x, int y)
{ return x - y; }
private static int Multiply(int x, int y)
{ return x * y; }
}
```

Delegatlar metod parametri sifatida

Delegatlar, shuningdek, metodlar parametri bo'lishi ham mumkin:

```
class Program
{
    delegate void GetMessage();
    static void Main(string[] args)
    {
        if (DateTime.Now.Hour < 12)
        {
            Show_Message(GoodMorning);
        }
        else
        {
            Show_Message(GoodEvening);
        }
        Console.ReadLine();
    }
    private static void Show_Message(GetMessage _del)
    {
        _del?.Invoke();
    }
    private static void GoodMorning()
    {
        Console.WriteLine("Good Morning");
    }
    private static void GoodEvening()
    {
        Console.WriteLine("Good Evening");
    }
}
```

To'ldirilgan delegatlar

Delegatlar to'ldirilgan bo'lishi ham mumkin, masalan:

```
delegate T Operation<T, K>(K val);
```

```
class Program
{
    static void Main(string[] args)
    {
        Operation<decimal, int> op = Square;
        Console.WriteLine(op(5));
        Console.Read();
    }
    static decimal Square(int n)
    { return n * n; }
}
```

Nazorat uchun savollar:

1. Delegat nima?
2. Delegatlar qanday aniqlanadi?
3. Metodga havola qanday o‘zlashtiriladi?
4. Metodlar delegatga qanday qo‘shiladi?
5. Delegatlarni birlashtirish qanday amalga oshiriladi?
6. Delegatlarga qanday murojaat qilinadi?
7. To‘ldirilgan delegatlarga misol keltiring

Foydalanish uchun tavsiya etiladigan adabiyotlar

1. Троелсен Эндрю, Джепикс Филипп. Язык программирования C# 7 и платформы .NET и .NET Core. Вильямс. 2018
2. Албахари Бен, Албахари Джозеф. C# 7.0. Справочник. Полное описание языка. Пер. с англ.-СПб: “Альфа-книга”, 2018, -1024 с.
3. Ю.С. Магда C#. Язык программирования Си Шарп. – Изд. ДМК Пресс, 2013, 190 с.
4. Лабор В.В. C#: Создание приложение для Windows. – Мн.: Харвест, 2003, 384 с.
5. <https://metanit.com/sharp/tutorial/3.13.php>

