13 дәріс. Делегаттар. Топтық адрестеу. Оқиғалар.

Дәрістің мақсаты: студенттерде делегаттар мен оқиғаларды пайдалану ерекшеліктері туралы түсініктерін көрсетуге қабілет қалыптастыру.

Осы дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Делегаттарды пайдалану ерекшеліктері туралы түсініктерін көрсету;
- Оқиғаларды пайдалану ерекшеліктері туралы түсініктерін көрсету;

Делегат түсінігінің анықтамасынан бастайық. Қарапайым тілмен айтқанда, *делегат әдіске сілтеме жасайтын объект* болып саналады. Сондықтан, делегат құру нәтижесінде әдіске сілтеме жасайтын объект пайда болады.

Сонымен бірге, әдісті осы сілтеме арқылы шақыруға болады. Басқаша айтқанда, делегат өзі сілтеме жасап тұрған әдісті шақыруға мүмкіндік береді. Төменде осындай қағиданың әрекеттер орындауға қаншалықты ыңғайлы болатыны көрсетіледі.

Программаның орындалу барысында бір ғана делегаттың өзі әртүрлі әдістерді шақыруға қолданыла алады, ол үшін делегат сілтеме жасап тұрған әдісті өзгерту жеткілікті. Сонымен, делегат шақыратын әдіс компиляция кезінде емес, орындалу кезеңінде анықталады. Сондықтан делегаттың басты артықшылығы да осында.

ЕСКЕРТУ

Егерде сіздің C/C++ тілдерінде программалаудан тәжірибеңіз бар болса, онда C# тіліндегі делегат C/C++ тіліндегі функцияға нұсқауышқа ұқсас болып келеді.

Делегат типі delegate түйінді сөзі арқылы жарияланады. Төменде делегаттың жалпы жазылу формасы көрсетілген:

```
delegate қайтарылатын_типі аты(параметрлер_тізімі);
```

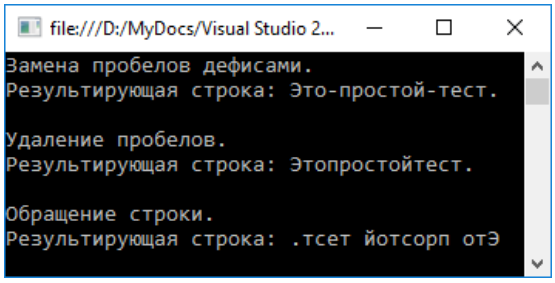
мұндағы *қайтарылатын_типі* – делегатпен шақырылатын әдістермен қайтарылатын мәннің типін білдіреді; *аты* – делегаттың нақты аты; *параметрлер_тізімі* – делегат арқылы шақырылатын әдіс үшін қажетті параметрлер. Делегаттың экземпляры құрылғаннан соң, ол бірден әдістерді шақырып сілтеме жасай алады, мұндағы әдістердің қайтаратын типі мен параметрлері делегаттың жариялануында көрсетілген мәндермен сәйкес болуы тиіс.

Ең бастысы, делегатты қайтарылатын типі мен сигнатурасы сәйкес келетін кез келген әдісті шақыру үшін қолдануға болады. Сонымен бірге, шақырылатын әдіс жеке объектімен байланысқан экземпляр әдісі немесе нақты класпен байланысқан статикалық әдіс болуы мүмкін. Бір ғана шартты қанағаттандырса болғаны: қайтарылатын тип пен әдістің сигнатурасы делегатты жариялау кезінде көрсетілген мәндермен сәйкес болуы керек.

Делегаттың жұмысын көрсету үшін алдымен оның қолданылуынан қарапайым мысал қарастырайық.

```
// Делегатты пайдаланудың қарапайым мысалы.

using System;

// Делегат типін жариялау.
delegate string StrMod(string str);

class DelegateTest
{
  // Босорындарды дефиспен алмастыру.
```

```
  static string ReplaceSpaces(string s) {
    Console.WriteLine("Замена пробелов дефисами.");
    return s.Replace(' ', '-');
    }

    // Босорындарды өшіру.
    static string RemoveSpaces(string s) {
      string temp = "";
      int i;

      Console.WriteLine("Удаление пробелов.");
      for (i = 0; i < s.Length; i++)
         if (s[i] != ' ') temp += s[i];
      return temp;
    }


    // Тіркесті кері жазу.
    static string Reverse(string s) {
      string temp = "";
      int i, j;

      Console.WriteLine("Обращение строки. ");
      for (j = 0, i = s.Length - 1; i >= 0; i--, j++)
         temp += s[i];
      return temp;
    }

    static void Main()
    {
      // Делегатты конструкциялау.
      StrMod strOp = new StrMod(ReplaceSpaces);
      string str;

      // Делегат арқылы әдісті шақыру.
      str = strOp("Это простой тест.");
      Console.WriteLine("Результирующая строка: " + str);
      Console.WriteLine();

      strOp = new StrMod(RemoveSpaces);
      str = strOp("Это простой тест.");
      Console.WriteLine("Результирующая строка: " + str);
      Console.WriteLine();

      strOp = new StrMod(Reverse);
      str = strOp("Это простой тест.");
      Console.WriteLine("Результирующая строка: " + str);
      Console.ReadKey();
  }
}
```



Бұл кодты орындау келесідей нәтиже береді.

Замена пробелов дефисами.

Результирующая строка: Это-простой-тест.

Удаление пробелов.
Результирующая строка: Этопростойтест.

Обращение строки.
Результирующая строка: .тсет йотсорп отЭ

Топтық адрестеу

Делегаттың ең тамаша қасиеттерінің бірі топтық адрестеуді сүйемелдеу болып табылады. Қысқаша айтқанда, *топтық адрестеу* – бұл делегатты пайдалану кезінде, автоматты түрде шақырылатын әдістер үшін жаңа *тізім* немесе *шақырулар тізбегін* жасау деген сөз. Мұндай тізім құру қиын емес. Ол үшін бір делегат экземплярын жасап алып, сонан соң + немесе += операторы көмегімен сол тізімге әдістер косу жеткілікті. Тізімдегі бір әдісті өшіру үшін - (азайту) немесе -= операторы қолданылады. Егерде делегат мән қайтаратын болса, онда ол шақырулар тізіміндегі ең соңғы әдіс қайтаратын мән болып табылады. Сондықтан, топтық адрестеу қолданылатын делегаттың қайтарылатын типі көбінесе void болады.

Төменде топтық адрестеу мысалы келтірілген. Бұл алдыңғы мысалдардың қайта өзгертілген нұсқасы, мұнда тіркестерді өңдейтін әдістер қайтаратын мәндердің типі void болып өзгертілген, ал өңделген сөз тіркесін кодтың шақыратын бөлігіне қайтару үшін ref типіндегі параметр қолданылады. Осыған орай бұл әдістер топтық адрестеуге бейімделген түрде болады.

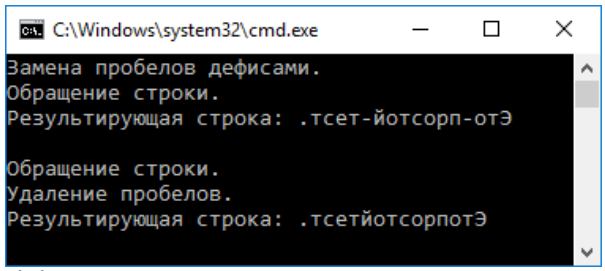
```
// Топтық  адрестеуді көрсету.

using System;

// Делегат типін жариялау
delegate void StrMod(ref string str);
class MultiCastDemo
{
    // Дефистерді бос орындармен алмастыру
    static void ReplaceSpaces(ref string s)
    {
        Console.WriteLine("Замена пробелов дефисами.");
        s = s.Replace(' ', '-');
    }
    // Бос орындарды өшіру
    static void RemoveSpaces(ref string s)
    {
        string temp = "";
        int i;

        Console.WriteLine("Удаление пробелов.");
        for (i = 0; i < s.Length; i++)
            if (s[i] != ' ') temp += s[i];
        s = temp;
    }

    // Тіркесті кері жазу.
    static void Reverse(ref string s)
    {
```

```
        string temp = "";
        int i, j;

        Console.WriteLine("Обращение строки.");
        for (j = 0, i = s.Length - 1; i >= 0; i--, j++)
            temp += s[i];

        s = temp;
    }

    static void Main()
    {
      //Делегатты конструкциялау.
      StrMod strOp;
      StrMod replaceSp = ReplaceSpaces;
      StrMod removeSp = RemoveSpaces;
      StrMod reverseStr = Reverse;
      string str = "Это простой тест.";

      // Топтық адрестеуді ұйымдастыру
      strOp = replaceSp;
      strOp += reverseStr;
      //Делегатқа топтық адрестеуді қолдану.
      strOp(ref str);
      Console.WriteLine("Результирующая строка: " + str);
      Console.WriteLine();
      //Босорындарды алмастыру әдісін өшіріп,босорындарды өшіретін әдіс қосу.
      strOp -= replaceSp;
      strOp += removeSp;
      str = "Это простой тест.";  // бастапқы жолды қалпына келтіру

      //  Топтық адрестеуі бар делегатты пайдалану.
      strOp(ref str);
      Console.WriteLine("Результирующая строка: " + str);
      Console.WriteLine();
    }
}
```

Бұл кодтың орындалуы келесі нәтижеге алып келеді.

Замена пробелов дефисами.
Обращение строки.
Результирующая строка: .тсет-йотсорп-отЭ

Обращение строки.
Удаление пробелов.
Результирующая строка: .тсетйотсорпотЭ



Қарастырылып отырған кодта Main() әдісінде делегаттың төрт экземпляры құрылады. Олардың біріншісі – strOp бос болып құрылады, ал қалған үшеуі сөз тіркестерін өзгертетін нақты әдістерге сілтеме жасайды. Содан кейін RemoveSpaces() и Reverse() әдістерін шақыру үшін топтық адрестеу ұйымдастырылады.

Делегаттардың атқаратын қызметі

Алдыңғы мысалдарда делегаттар қызметінің ішкі механизмі көрсетілген болатын, бірақ бұл мысалдар олардың шынайы атқаратын қызметін айқын бере алмайды.

Көбінесе, делегаттар екі түрлі себеппен қолданылады. Біріншіден, бұдан бұрын айтылғандай, делегаттар оқиғаларды сүйемелдейді. Екіншіден, делегаттар программаның орындалуы кезінде компиляция барысында, олар туралы ешнәрсені анық білмесе де, әдістерді шақыра алады. Бұл жекелеген программалық компоненттерді қосуға мүмкіндік беретін базалық конструкцияны құру кезінде өте қолайлы.

Мысал ретінде стандартты Windows Paint сервистік программасына ұқсас графикалық программаны қарастырайық. Делегат көмегімен қолданушыға арнайы түрлі түсті фильтрлер мен бейнелер анализаторын қосатын мүмкіндіктер беруге болады. Сонымен қатар, қолданушы осы фильтрлер мен анализаторларды пайдалана отырып, олардан бүтін бір тізбекті бейнелер құрастыра алады. Делегаттарды қолдану арқылы программалардың осы сияқты мүмкіндіктерін қамтамасыз ету қиын емес.

Оқиғалар

C# тіліндегі делегаттарға негізделіп жасалған тағы бір құрал – оқиға. Оқиға, негізінде, қандай да бір әрекеттің болғанын автоматты түрде хабарлайды. Оқиғалар келесі принцип бойынша әрекет жасайды: оқиғаға қызығушылық білдіретін объект, сол оқиғаның өңдеуішін тіркеуге алады. Оқиға орындалғанда, сол оқиғаның барлық тіркелген өңдеуіштері шақырылады. Оқиға өңдеуіштері әдетте делегаттар түрінде беріледі. Оқиғалар кластың мүшелері болып табылады да, event түйінді сөзі көмегімен жарияланады. Көбінесе олар мынадай формада қолданылады:

event *оқиға_делегаты оқиға_аты;*

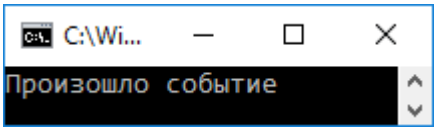
мұндағы *оқиға_делегаты* оқиғаны сүйемелдеу үшін қолданылатын делегат атын белгілейді, ал *оқиға_аты* — жарияланатын оқиғаның нақты бір объектісі.

Алдымен қарапайым мысал қарастырайық.

```
// Оқиға жұмысын көрсететін өте қарапайым мысал

using System;

// Оқиға үшін делегат типін анықтау.
delegate void MyEventHandler();

// Оқиғаны қамтитын классты жариялау
class MyEvent {
  public event MyEventHandler SomeEvent;
  // бұл әдіс оқиғаны іске қосу үшін қолданылады
  public void OnSomeEvent() {
      if (SomeEvent != null)
          SomeEvent();
  }
}
class EventDemo {
  // Оқиғаны өңдеуіш
  static void Handler() {
      Console.WriteLine("Произошло событие");
  }

  static void Main() {
    MyEvent evt = new MyEvent();
    //  Handler()  әдісін оқиға тізіміне қосу,
```

```
    evt.SomeEvent += Handler;
    // Оқиғаны жіберу
    evt.OnSomeEvent();
  }
}
```

Кодты орындағанда шығатын нәтиже:



Произошло событие

Өзінің қарапайымдылығына қарамастан, берілген мысалда оқиғаны өңдеуге қажет барлық негізгі элементтер бар. Ол төменде көрсетілгендей, оқиғаны өңдеуіш үшін қажетті делегаттың типін жариялаудан басталады.

```
delegate void MyEventHandler();
```

Барлық оқиғалар делегаттар көмегімен іске асырылады. Сондықтан, оқиға делегатының типі сол оқиға үшін қайтарылатын тип пен сигнатураны анықтайды. Мұндағы жағдайда оқиғаның параметрлері жоқ, бірақ оларды көрсетуге рұқсат етіледі.

Ары қарай MyEvent оқиға класы құрылады. Бұл класта келесі жолдағы SomeEvent оқиғасы жарияланады.

```
public event MyEventHandler SomeEvent;
```

Бұл жарияланудың синтаксисіне назар аударыңыз. Event түйінді сөзі компиляторға оқиға жарияланатыны туралы мәлімет береді.

Сонымен қатар, MyEvent класында оқиғаның іске қосылатыны жайлы сигнал беру үшін шақырылатын OnSomeEvent () әдісі жарияланады. Бұл оның оқиға орындалуы кезінде шақырылатынын білдіреді. OnSomeEvent () әдісінде SomeEvent делегаты көмегімен оқиға өңдеуіші шақырылады.

```
if (SomeEvent != null)
SomeEvent();
```

Көріп тұрғаныңыздай, SomeEvent оқиғасы бос болмаған жағдайда ғана оқиға өңдеуіші шақырылады. Ал оқиғаға қызығушылық программаның басқа бөліктерінде тіркелуі керек болғандықтан, ол туралы мәлімет алу үшін, кез келген оқиға өңдеуіші тіркелгенге дейін OnSomeEvent () әдісі шақырылуы керек. Бірақ бос сілтеме бойынша шақыруды болдырмас үшін, оқиға делегатының бос емес екендігіне көз жеткізу мақсатында, оқиға делегаты тексерілуі тиіс.

EventDemo класында Handler() оқиға өңдеуіші құрылады. Берілген осы қарапайым мысалда оқиға өңдеуіші жай ғана хабарлама шығарады, бірақ басқа өңдеуіштер мұнан гөрі күрделірек функцияларды орындай алады. Ары қарай, Main() әдісінде MyEvent оқиға класының объектісі құрылады, ал Handler() тізімге қосылатын осы оқиғаның өңдеуіші ретінде тіркеледі.

```
MyEvent evt = new MyEvent();
//  Handler() әдісін оқиға тізіміне қосу.

evt.SomeEvent += Handler;
```

Өңдеуіштің += операторы арқылы тізімге қосылатынына назар аударыңыздар. Оқиғаларды тек += және -= операторлары ғана сүйемелдейді. Мұндағы жағдайда Handler()

статикалық әдіс болып табылады, бірақ оқиға өңдеуіштері ретінде экземпляр әдістері де қолданыла алады.

Сонымен, ең соңында оқиға төменде көрсетілгендей түрде іске қосылады.

```
// Оқиғаны іске қосу
evt.OnSomeEvent();
```

OnSomeEvent() әдісін шақыру өңдеуіш тіркеген барлық оқиғаларды шақыруға әкеп соқтырады. Мұндағы жағдайда тек бір өңдеуіш тіркелген, бірақ олар, келесі бөлімде көрсетілетіндей, бірнешеу болулары да мүмкін.