23. Делегаттар. Лямбда-өрнектер. Уақиғалар.

Делегаттар.

Делегат — бұл тәсілдерге сілтемелерді сақтауға арналған класс түрі. Делегатты кез кезген класс тәрізді параметр ретінде беруге, осыдан кейін ондағы инкапсуляцияланған тәсілді шақыруға болады.

Делегаттар оқиғаларды қолдау үшін, сонымен қатар, тілдің тәуелсіз конструкциясы ретінде қолданылады.

Делегаттың сипаттамасы оның көмегімен шақырылуы мүмкін тәсілдер сигнатурасын анықтайды:

[атрибуттар] [спецификаторлар] delegate тип аты([параметрлер])

Делегатты сипаттау мысалы:

```
public delegate void D ( int i );
```

Делегаттың базалық класы System.Delegate болып табылады

Делегаттар, негізінен, келесі жағдайлар үшін қолданылады:

1. шақырылған тәсілді компиляция кезінде емес, программаның орындалуы кезінде динамикалық түрде анықтауға мүмкіндік алу;
2. объектілер арасында «шығу көзі — бақылаушы» типі бойынша байланысты қамтамасыз ету;
3. оларға басқа тәсілдерді беруге болатындай универсалды тәсілдерді құру (кері шақырулар механизмін қолдау).

Делегатты параметрлер тізімі арқылы беру

```
namespace ConsoleApplication1 {
  public delegate double Fun( double x );      // делегатты жариялау
  class Class1 {
     public static void Table( Fun F, double x, double b )
     {   Console.WriteLine( " ----- X ----- Y -----" );

         while (x <= b)
         {  Console.WriteLine( "| {0,8} | {1,8} |", x, F(x));
           x += 1; }
      }
      public static double Simple( double x ) { return 1; }
      static void Main()
      { Table( Simple, 0, 3 );
        Table( Math.Sin, -2, 2 );                  // new Fun(Math.Sin)
        Table( delegate (double x ){ return 1; }, 0, 3 );
}}}
```

Лямбда-өрнектер.

Лямбда-өрнектер – бұл өрнек ағашын немесе делегаттар типін құруға қолданылатын анонимді функция. Лямбда-өрнегі арқылы локальді функция жазуға болады, оны кейін аргумент ретінде немесе мән қайтару үшін басқа функцияларға беруге болады. Лямюда-өрнектер әсіресе LINQ сұраныстарын жазғанда пайдаланылады. Лямбда өрнекті құруда кіріс параметрлерін сол жақтан және блок мағынасы немесе басқа бағытта нәтижесі көрсетіледі. Келесі мысалдағыдай:

```
delegate int del(int i);
static void Main(string[] args)
{
  del myDelegate = x => x * x;
  int j = myDelegate(5); //j = 25
```

}

Уақиғалар.

Оқиға — кластың басқа объектілерге (бақылаушыларға) өзінің күйінің өзгеруі туралы хабарламалар жіберуіне мүмкіндік беретін класс элементі.

Бақылаушы болу үшін объектінің оқиғалар өңдеушісі және ол оқиғалар өңдеушісін шығу көзі объектісінде тіркеу керек.

Оқиғалар механизмі

Оқиғалар делегаттар негізіде құрастырылған: делегаттардың көмегімен оқиғаларды өңдеуші тәсілдер шақырылады. Сондықтан класта *оқиғаны құру* келесі бөліндерден тұрады:

1. оқиғалар өңдеушілерінің сигнатурасын анықтайтын делегатты сипаттау;
2. оқиғаның сипаттау;

Оқиғаны тудыратын тәсілді (тәсілдерді) сипаттау.

Оқиғаның синтаксисі:

[атрибуттар] [спецификаторлар] event тип аты

Оқиғаларды өңдеу хабарламаны алушы кластарда орындалады. Ол үшін осы кластарда сигнатурасы делегат типіне сәйкес келетін оқиғаларды өңдеуші әдістер сипатталады. Хабарлама алғысы келетін әрбір объект (класс емес!) жіберуші объектіде осы тәсілді тіркеуі тиіс.

Оқиға — бұл программист үшін ыңғайлы абстракция. Шын мәнінде, ол ішінде делегат экземпляры құрылатын жабық статикалық кластан және өңдеушіні осы делегат тізіміне қосуға және одан жоюға арналған екі тәсілден тұрады.

Сыртқы код оқиғалармен бір сипатта ғана жұмыс істей алады: өңдеушілерді тізімге қосу немесе тізімнен өшіру, себебі кластан тыс тек += және -= операцияларын қолдануға болады. Бұл операциялар нәтижесінің типі — void, яғни ол арифметикалық типтер үшін күрделі меншіктеу операцияларынан өзгеше болып келеді. Өңдеушілер тізіміне басқаша қатынас құру тәсілдері жоқ.

23. Делегаттар. Лямбда-өрнектер. Уақиғалар. Делегаттар. Делегат — бұл тәсілдерге сілтемелерді сақтауға арналған класс түрі. Делегатты кез кезген класс тәрізді параметр ретінде беруге, осыдан кейін ондағы инкапсуляцияланған тәсілді шақыруға болады. Делегаттар оқиғаларды қолдау үшін, сонымен қатар, тілдің тәуелсіз конструкциясы ретінде қолданылады. Делегаттың сипаттамасы оның көмегімен шақырылуы мүмкін тәсілдер сигнатурасын анықтайды: [атрибуттар] [спецификаторлар] delegate тип аты([параметрлер]) Делегатты сипаттау мысалы: public delegate void D (int i); Делегаттың базалық класы System.Delegate болып табылады Делегаттар, негізінен, келесі жағдайлар үшін қолданылады: шақырылған тәсілді компиляция кезінде емес, программаның орындалуы кезінде динамикалық түрде анықтауға мүмкіндік алу; объектілер арасында «шығу көзі — бақылаушы» типі бойынша байланысты қамтамасыз ету; оларға басқа тәсілдерді беруге болатындай универсалды тәсілдерді құру (кері шақырулар механизмін қолдау). Делегатты параметрлер тізімі арқылы беру namespace ConsoleApplication1 { public delegate double Fun(double x); // делегатты жариялау class Class1 { public static void Table(Fun F, double x, double b) { Console.WriteLine(" ---- X ----- Y -----"); while (x <= b) { Console.WriteLine("| {0,8} | {1,8} |", x, F(x)); x += 1; } } public static double Simple(double x) { return 1; } static void Main() { Table(Simple, 0, 3); Table(Math.Sin, -2, 2); // new Fun(Math.Sin) Table(delegate (double x){ return 1; }, 0, 3); }}} Лямбда-өрнектер. Лямбда-өрнектер – бұл өрнек ағашын немесе делегаттар типін құруға қолданылатын анонимді функция. Лямбда-өрнегі арқылы локальді функция жазуға болады, оны кейін аргумент ретінде немесе мән қайтару үшін басқа функцияларға беруге болады. Лямюда-өрнектер әсіресе LINQ сұраныстарын жазғанда пайдаланылады. Лямбда өрнекті құруда кіріс параметрлерін сол жақтан және блок мағынасы немесе басқа бағытта нәтижесі көрсетіледі. Келесі мысалдағыдай: delegate int del(int i); static void Main(string[] args) { del myDelegate = x => x * x; int j = myDelegate(5); //j = 25 } Уақиғалар. Оқиға — кластың басқа объектілерге (бақылаушыларға) өзінің күйінің өзгеруі туралы хабарламалар жіберуіне мүмкіндік беретін класс элементі. Бақылаушы болу үшін объектінің оқиғалар өңдеушісі және ол оқиғалар өңдеушісін шығу көзі объектісінде тіркеу керек. Оқиғалар механизмі Оқиғалар делегаттар негізіде құрастырылған: делегаттардың көмегімен оқиғаларды өңдеуші тәсілдер шақырылады. Сондықтан класта оқиғаны құру келесі бөлімдерден тұрады: оқиғалар өңдеушілерінің сигнатурасын анықтайтын делегатты сипаттау; оқиғаның сипаттау; Оқиғаны тудыратын тәсілді (тәсілдерді) сипаттау. Оқиғаның синтаксисі: [атрибуттар] [спецификаторлар] event тип аты Оқиғаларды өңдеу хабарламаны алушы кластарда орындалады. Ол үшін осы кластарда сигнатурасы делегат типіне сәйкес келетін оқиғаларды өңдеуші әдістер сипатталады. Хабарлама алғысы келетін әрбір объект (класс емес!) жіберуші объектіде осы тәсілді тіркеуі

тиіс. Оқиға — бұл программист үшін ыңғайлы абстракция. Шын мәнінде, ол ішінде делегат экземпляры құрылатын жабық статикалық кластан және өңдеушіні осы делегат тізіміне қосуға және одан жоюға арналған екі тәсілден тұрады. Сыртқы код оқиғалармен бір сипатта ғана жұмыс істей алады: өңдеушілерді тізімге қосу немесе тізімнен өшіру, себебі кластан тыс тек += және -= операцияларын қолдануға болады. Бұл операциялар нәтижесінің типі — void, яғни ол арифметикалық типтер үшін күрделі меншіктеу операцияларынан өзгеше болып келеді. Өңдеушілер тізіміне басқаша қатынас құру тәсілдері жоқ. 24. Сөз тіркесін құру. Жүйелі өрнектер. Сөз тіркесін құру. String – көптеген пайдалы әдістерді жүзеге асыратын аса мықты класс болып есептелінеді. Алайда String классының жеткіліксіздігі оның өзгертілмейтіндігі , яғни бір кез инициализацияланған жолды объект өзгертіле алмайды. StringBuilder классы екі басты сипаттамадан тұрады: 1)Length объекттегі дәл сол уақыттағы бар жолдың ұзындығын көрсетеді 2)Capacity белгіленген объектің жадысына орналаса алатын жолдың максималды ұзындығын көрсетеді. Кез келген жолдың модификациясы StringBuilder экземплярына бөлінген блок ішіндегі жадыда орындалады. Бұл жол бөлімдері мен инвидуалды жол символдарының ауыстыруларын өте әсерлі етеді. Егер StringBuilder құрамын String түрінде алу үшін ToString() әдісін қолдану керек. Көп жағдайда StringBuilder –ді көптеген жолдарды манипуляциялау қажет болғанда қолданған дұрыс. Алайда қарапайым зат жасайтын кезде,мысалы , екі жолды біріктіру үшін System.String-ті қолданған ыңғайлы. Жүйелі өрнектер. Жүйелік өрнектер- бұл, программаның үлкен диапазонында кең қолданылатын , бірақ жасап шығарушыларда аз қолданылатын технологиялық облыстың кішігірім бөлігі болып есептеледі. Жүйелік өрнектерді бір спецификалық мақсаты бар программалаудың кішігірім тілі деп елестетуге болады. Бұл жаңа технология емес, бастапқыда ол UNIX ортасында пайда болды және әдетте Perl программалау тілінде қолданылады. Microsoft жасап шығарушылары оны Windows-қа көшірді. Қазір жүйелік өрнектер .NET-гі System.Text.RegularExpressions атаулы аймақтағы көптеген класстармен жұмыс істейді . Жүйелік өрнекті қолданған жағдайда ортаның көптеген аймағында кездестіруге болады. Төмендегі мысалда көрсетілгендей жүйелі сөйлемдер көмегімен жолдарға қиын және жоғары деңгейлі әрекет жасауға болады. Жолдағы барлық ұқсас сөздерді теңестіру Барлық сөздің бірінші әрібін бас әріппен жазу Үш сөзден көп болатын сөйлемдегі бірінші сөздердің бірінші әрібін өзгерту. Сөйлемдердің дұрыс капиталдануын қадағалау URI – дағы әр түрлі элементтерді белгілеу