9 дәріс. Виртуалды әдістер және оларды қайта анықтау. Абстрактілі кластар.

Дәрістің мақсаты: студенттерде виртуалды әдістер мен абстрактілі кластарды пайдалану ерекшеліктері туралы түсініктерін көрсетуге қабілет қалыптастыру.

Осы дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Виртуалды әдістердің қызметі бойынша түсініктерін көрсету;
- Әдісті қайта анықтау механизмі туралы түсініктерін көрсету;
- Абстрактілі кластарды пайдалану ерекшеліктері бойынша түсініктерін көрсету.

Виртуалды әдіс деп базалық класта **virtual** ретінде жарияланатын және туынды кластарда қайта анықталуы мүмкін әдісті атайды. Ендеше, әрбір туынды класта виртуалды әдістің өзіндік нұсқасы болуы мүмкін. C# тілінде виртуалды әдісті шақыру кезінде сілтеме бойынша шақыру орындалып жатқан объектінің типіне байланысты виртуалды әдістің қай нұсқасы шақырылуы керектігі анықталады. Яғни, виртуалды әдістің орындалатын нұсқасы объектіге сілтеменің типіне емес, объектінің өз типіне байланысты таңдалады. Осылайша, егер базалық класта виртуалды әдіс анықталған болса, базалық класқа сілтеме арқылы әртүрлі типтегі объектілерге қатынасу кезінде осы виртуалды әдістің әртүрлі нұсқалары орындалады.

Виртуалды әдіс базалық класта **virtual** кілттік сөзінің көмегімен жарияланады. Туынды класта виртуалды әдісті қайта анықтау **override** модификаторының көмегімен орындалады. Әдісті қайта анықтау кезінде оның аты, қайтаратын типі және сигнатурасы дәл базалық кластағы нұсқамен сәйкес келуі тиіс.

1-мысал. Виртуалды әдістерді құру және қайта анықтау

```
using System;
class Onim {
   public virtual void Shygaru()
   {
      Console.WriteLine("Onim klasymen zhumys");
   }
}
class Tauar : Onim {
   public override void Shygaru()
   {
      Console.WriteLine("Tauar klasymen zhumys");
   }
   // Берілген тауардың жалпы бағасы
}
class Program {
   static void Main() {
      Onim onim001 = new Onim();
      Tauar tauar001 = new Tauar();
      Onim Silteme;
      Silteme = onim001; Silteme.Shygaru();
      Silteme = tauar001;
      Silteme.Shygaru();
      Console.ReadKey();
   }
}
```

Бұл программаның орындалу нәтижесі келесідей болады:

```
Onim klasymen zhumys
Tauar klasymen zhumys
```

Мысалда берілген **Shygaru()** виртуалды әдісі туынды класта қайта анықталған. **Program** класының ішінде базалық кластың **Silteme** сілтемелік типті айнымалысына алдымен базалық класс типіндегі объектіге сілтеме меншіктеліп, **Shygaru()** әдісі шақырылады, содан кейін **Silteme** айнымалысына туынды класс типіндегі объектіге сілтеме меншіктеледі, бұл жағдайда да **Shygaru()** әдісі шақырылады. **Shygaru()** әдісін бірінші шақырған кезде базалық класта анықталған виртуалды әдіс, ал екінші шақыру кезінде туынды класта қайта анықталған әдіс орындалады.

Виртуалды әдісті туынды класта қайта анықтау міндетті емес, бұл жағдайда туынды класс типіндегі объект арқылы базалық класта анықталған әдіс нұсқасы шақырылады. Әдістерді қайта анықтаудың көмегімен C# тілінде полиморфизм принципі қолдау табады.

Абстрактілі кластар

Кейбір жағдайларда барлық туынды кластар үшін барынша жалпы формасы ғана анықталатын базалық класты құру қажет болады, мұндай кластың мазмұндылығын толықтыру туынды кластардың еншісіне беріледі. Базалық класс мазмұны нақты болмаған жағдайда оның құрамында анықталған әдістердің туынды кластарда міндетті түрде қайта анықталғаны жөн. Бұл әрекет абстрактылы әдістерді пайдалану арқылы орындалады.

Абстрактылы әдіс **abstract** типіндегі модификатор көмегімен құрылады. Абстрактылы әдістің тұлғасы болмайды, сондықтан ол базалық класта жүзеге асырылмайды және туынды класта міндетті түрде қайта анықталу керек. Абстрактылы әдіс автоматты түрде виртуалды болып саналады.

Абстрактылы әдісті анықтау формасы төмендегідей:

```
abstract тип аты  {параметрлер_тізімі );
```

Көріп отырғандарыңыздай, абстрактылы әдістің тұлғасы болмайды. Құрамында бір немесе бірнеше абстрактылы әдіс анықталған класс та абстрактылы болып жариялануы тиіс, ол үшін класс жариялануындағы **class** кілттік сөзінің алдында **abstract** модификаторы көрсетіледі. Абстрактылы кластың қызметі толық анықталмайтын болғандықтан, оның объектілерін құру мүмкін емес.

Абстрактылы кластан мұралайтын туынды класта базалық кластың барлық абстрактылы әдістері қайта анықталуы міндетті. Олай болмаған жағдайда туынды класс та абстрактылы болып жариялануы керек.

2-мысал. Абстрактылы класс мысалы

```
using System;
abstract class Progressiya {
   int a0;
   int qadam;
   // Келісім бойынша конструктор
   public Progressiya()
   {
      a0 = 1; qadam = 2;
   }
   // Параметрленген конструктор
   public Progressiya(int a, int q)
   {
      a0 = a; qadam = q;
   }
   // Қасиеттер
   public int A0
   {
      get; set;
   }
   public int Qadam
   {
      get { return qadam; }
      set { if (value > 0) qadam = value;
```

```
            else qadam = 1; }
   }
   public void Shygaru()
   {
      Console.Write("Progressiyanyng algashqy elementi – "
                    + a0 + ", qadamy – " + qadam);
   }
   public abstract void Tip();
   // прогрессияның n-ші мүшесінің мәні
   public abstract double an(int n);
   // прогрессияның n-шы қосындысы
   public abstract double qosyndy(int n);
}
// Арифметикалық прогрессия туынды класы
class Arifm : Progressiya {
   public Arifm() { }
   public Arifm(int a, int q) : base(a, q) { }
   // Класс үшін n-мүшесінің мәнін есептеу әдісін қайта анықтау
   public override double an(int n)
   {
      return A0 + n * Qadam;
   }
   public override double qosyndy(int n)
   {
      return (n + 1) * (2*A0 + n * Qadam)/ 2;
   }
   // Прогрессия түрін көрсету
   public override void Tip()
   {
      Console.WriteLine(", arifmetikalyq progressiya ");
   }
}
// Геометриялық прогрессия туынды класы
class Geom : Progressiya {
   public Geom() { }
   public Geom(int a, int q) : base(a, q) { }
   // Класс үшін n-мүшесінің мәнін есептеу әдісін қайта анықтау
   public override double an(int n)
   {
      return A0 * Math.Pow(Qadam, n);
   }
   public override double qosyndy(int n)
   {
      double An = an(n);
      return (A0 - An * n) / (1 - n);
   }
   // Прогрессия түрін көрсету
   public override void Tip()
   {
      Console.WriteLine(", geometriyalyq progressiya ");
   }
}
class Program {
   static void Main() {
      Progressiya[] pro = new Progressiya[4];
      int n = 5;
      pro[0] = new Arifm(1, 5); pro[1] = new Arifm();
      pro[2] = new Geom(3, 4); pro[3] = new Geom();
      for (int i = 0; i < pro.Length; i++)
      {
         pro[i].Shygaru(); pro[i].Tip();
         Console.WriteLine("Progressiyanyng " + n + "-
              mushesining mani = " + pro[i].an(n) + ", al " + n
              + "-qosyndysynyng mani = " + pro[i].qosyndy(n));
```

```
            }
            Console.ReadKey();
        }
    }
```

Программаның нәтижесі:

```
    Progressiysnyng algashqy elementi – 1, qadamy – 5,
arifmetikalyq progressiya
    Progressiyanyng 5-mushesining mani = 25, al 5-
qosyndysynyng mani = 75
    Progressiysnyng algashqy elementi – 1, qadamy – 2,
arifmetikalyq progressiya
    Progressiyanyng 5-mushesining mani = 10, al 5-
qosyndysynyng mani = 30
    Progressiysnyng algashqy elementi – 3, qadamy – 4,
geometriyalyq progressiya
    Progressiyanyng 5-mushesining mani = 3072, al 5-
qosyndysynyng mani = 3840
    Progressiysnyng algashqy elementi – 1, qadamy – 2,
geometriyalyq progressiya
    Progressiyanyng 5-mushesining mani = 32, al 5-
qosyndysynyng mani = 40
```

Осылайша, абстрактылы кластың барынша пайдалы қасиеттерінің бірі – кез келген туынды кластарының объектілерімен цикл құрамында жұмыс жасауға мүмкіндік беруі.