3 дәріс. Конструкторлар және деструкторлар. Объектілерді әдістерге беру және әдістерден объектілерді қайтару.

Дәрістің мақсаты: студенттерде конструкторлар, деструкторлар қызметі және объектілермен әдістердің құрамында жұмыс істеу ерекшеліктері туралы түсініктерін көрсетуге қабілет қалыптастыру.

Осы дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- конструкторлар мен деструкторлардың қызметі және қолданылу форматы туралы түсініктерін көрсету;
- әдістерге объектілерді беру тәсілдері бойынша түсініктерін көрсету;
- әдістерге объектілерді беру тәсілдері бойынша түсініктерін көрсету;

Конструкторлар

Жоғарыдағы программа мысалдарында **Shenber** типіндегі әрбір объектідегі экземпляр айнымалыларын қолдап келесідей операторлар тізбегі арқылы инициалдауға тура келген еді.

```
shenber1.xcoord = 2;
shenber1.ycoord =2;
shenber1.radius = 3;
```

Мұндай тәсіл әдетте C# тілінде тілінде кәсіби түрде жазылған кодтарда қолданылмайды. Оның үстіне, мұнда қате де кетуі мүмкін (сіз өрістердің бірін инициалдауды ұмытып та кетуіңіз де мүмкін ғой). Дегенмен, осындай сәттерде көмектесетін жақсы бір тәсіл бар, ол: конструкторларды қолдану.

Конструктор объектіні құру кезінде бірден инициалдайды. Конструктордың аты оның класс атымен бірдей болып келеді де, синтаксис жағынан ол әдіске ұқсайды. Бірақ конструкторлардың тікелей көрсетіліп қайтарылатын типі жоқ.

Төменде конструктордың жапы формасы келтірілген.

```
қатынасу_типі класс_аты(параметрлер_тізімі) {
   // конструктор тұлғасы
}
```

Көбінесе конструктор класта анықталған экземпляр айнымалыларының бастапқы мәндерін беру үшін қолданылады немесе басқа да толық қалыптасқан объект жасау үшін қажет етілетін орнату процедураларын орындау үшін пайдаланылады. Одан қоса, конструкторлар көбінесе класта шақырылатындықтан, оның қатынасу типінің модификаторы **public** болып келеді. Ал *параметрлер_тізімі* кейде бос болады, ал көбінесе көрсетілген бір немесе бірнеше параметрлер тізімінен тұрады.

Программада анықтасаңыз да немесе анықтамасаңыз да, бәрі бір барлық кластарда конструкторлар болады, өйткені C# тілінде, алдын ала келісім бойынша, конструктор автоматты түрде құрылады, ол барлық экземпляр айнымалыларын берілетін мәндермен инициалдайды. Мәліметтердің көптеген типтері үшін алдын ала келісім бойынша берілетін мән нөл болады, **bool** типі үшін – **false** мәні алынса, ал сілтемелі типтер үшін – бос мән алынады. Бірақ егер сіз программада өз конструкторыңызды анықтайтын болсаңыз, онда келісім бойынша құрылатын конструктор пайдаланылмайды.

Мысал 7.3. Конструкторды қолданудың қарапайым мысалы.

```
using System;
class Shenber {
   public int xcoord;
   public int ycoord;
   public double radius;
   public Shenber()
   {
      xcoord = 2;
      ycoord = 2;
      radius = 1;
   }
}
class Program {
   static void Main() {
      Shenber shenber1 = new Shenber();
      Console.WriteLine("shenber centri: " + shenber1.xcoord + ", " +
shenber1.ycoord + ", shenber radiusy – " + shenber1.radius);
      Console.ReadKey();
   }
}
```

Бұл мысалдағы **Shenber** класының конструкторы мынадай түрде берілген.

```
public Shenber()
{
   xcoord = 2;
   ycoord = 2;
   radius = 1;
}
```

Мұндағы конструктордың **public** түрінде белгіленгеніне назар аударыңыз. Ол өз класы сыртындағы кодтардан да шақырылатын болуы тиіс. Ол объектіні құру кезінде **new** операторында шақырылады. Мысалы, келесі жолда:

```
Shenber shenber1 = new Shenber();
```

Конструкциялаудан кейін **shenber1** объектісінің айнымалылары сәйкесінше 2, 2, 3 мәндерін алады. Сонымен, жоғарыда келтірілген кодты орындау нәтижесі мынадай болады.

```
shenber centri: 2, 2, shenber radiusy – 1;
```

Алдыңғы мысалда параметрсіз конструктор пайдаланған болатын. Кейбір жағдайларда ол жеткілікті болғанмен, көбінесе конструктор бір немесе бірнеше параметрдерді қабылдауы тиіс. Конструкторға параметрлер әдістерге енгізілетін сияқты түрде енгізіледі. Ол үшін параметрлерді конструктор атынан кейін жай жақшалар ішінде жариялау жеткілікті.

Мысал 7.4. Параметрленген **Shenber** конструкторын қолдану мысалы.

```
using System;
class Shenber {
   public int xcoord;
   public int ycoord;
   public double radius;
   public Shenber(int x, int y, int r)
   {
      xcoord = x;
      ycoord = y;
      radius = r;
   }
}
class Program {
```

```
        static void Main()
        {
           Shenber shenber1 = new Shenber(2,2,1);
           Console.WriteLine("shenber centri: " + shenber1.xcoord + ", " +
shenber1.ycoord + ", shenber radiusy – " + shenber1.radius);
            Console.ReadKey();
        }
    }
```

Бұл кодты орындау нәтижесі мынадай болады:

```
shenber centri: 2, 2, shenber radiusy – 1;
```

Деструкторлар

Жоғарыда көрсетілгендей, new операторын қолдану кезінде жасалатын объектілер үшін компьютер жедел жадынан алынған бос орын көлемі (буфердегі) динамикалық түрде бөлінеді. Әрине, жедел жады көлемі шексіз емес, ол ерте ме, кеш пе, әйтеуір бітеді. Сондай кездерде, керекті объект құру үшін бос жады мөлшерінің болмай қалуы **new** операторының дұрыс орындалмауына алып келеді. Міне, сондықтан да компьютер жадын динамикалық түрде бөлудің кез келген схемасының басты функциясы пайдаланылмайтын объектілерден жадыны босатып, оны келесі жұмыстарға қолдануға дайындау болып табылады.

C# тілінде "*қоқысты жинау*" жүйесінің өзі компьютер жадын пайдаланылып болған объектілерден байқаусыз және программалаушының қатысуынсыз автоматты түрде босатады. "Қоқысты жинау" былай жүргізіледі. Егер объектіге сілтеме болмаса, онда ондай объект керексіз деп есептеледі де, ол орналасқан жады босатылып, буферге қосылады. Осылай "тазартылған" жады басқа объектілерге бөлініп берілетін болады. "Қоқысты жинау" программаны орындау барысында, анда-санда жүргізіліп отырады. Ол жұмыстан босаған бір-екі объект үшін орындала салмайды, өз кезегін күтеді. Сонымен, "қоқысты жинау" қай кезде болатынын алдын ала білуге немесе болжауға болмайды.

C# тілінде "қоқысты жинау" жүйесі арқылы қажет болмайтын объектіні жою алдында орындалатын әдісті шақыруды анықтау мүмкіндігі бар. Мұндай әдіс деструктор деп аталады және ол объектімен жұмыс істеу мерзімін аяқтауға кепілдік беретін ерекше жағдайларда пайдаланыла алады. Мысалы, деструктор босатылуға тиіс объект алып тұрған жүйелік ресурсты кепілді түрде босату үшін қолданылады. Бірақ деструкторлар тек сирек туындайтын ерекше жағдайларда ғана қолданылатын құралдар болып табылатынын айта кету керек.

Төменде деструктордың жалпы жазылу формасы келтірілген:

```
~класс_аты() {
   // деструктор коды
}
```

мұндағы *класс_аты* – нақты бір кластың аты. Деструктор конструктор тәрізді жарияланады, тек оның аты алдына "тильда" (~) белгісі қойылады. Деструктордың қайтарылатын типі мен оған берілетін аргументтері болмайды.

Деструкторды бір класқа қосу үшін, оны класс құрамына мүше етіп енгізу жеткілікті. Ол өз класындағы белгілі бір объектіні жою керек болған сайын іске қосылады. Деструкторда объектіні жою алдында орындалуға тиіс әрекеттерді көрсетуге болады.

Мысал 7.5. Төменде деструкторды қолдануды көрсететін программа мысалы келтірілген. Ол программада көптеген объектілер құрылады және өшіріледі. Ол процестің орындалуы барысындағы кейбір сәттерде "қоқысты жинау" іске қосылып, қажетсіз объектілерді өшіру үшін деструкторлар шақырылып жатады.

```
using System;
class Destruct {
   public int x;
   public Destruct(int i) {
      x = i;
   }
   // Объектіні өшіру кезінде шақырылады
   ~Destruct() {
      Console.WriteLine("Joiylady " + x);
   }
   // Объектіні жасайды да және оны бірден жояды
   public void Generator(int i) {
      Destruct o = new Destruct (i);
   }
}
class Program {
   static void Main() {
       int count;
       Destruct ob = new Destruct(0);
       /* Ал енді көптеген объектілер жасалады.
Белгілі бір сәттерде "қоқыс жинау" орын алады.
Ескерту: "қоқыс жинауды" екпінді ету үшін жасалатын
объектілер санын көбейту керек шығар */
       for (count=1; count < 100000; count++)
       ob.Generator(count);
       Console.WriteLine("Dayin!");
   }
}
```

Программа орындалуының нәтижесі:

```
    ...
    Joiylady 4403
    Joiylady 4402
    Joiylady 4401
    Joiylady 4400
    Joiylady 4399
    Joiylady 4398
    Joiylady 4397
    Joiylady 4396
    Joiylady 4395
    Joiylady 4394
    Joiylady 4393
    ...
```

Бұл программа былай жұмыс істейді. Конструктор **x** айнымалысын белгілі бір мәнмен инициалдайды. Бұл мысалда **x** айнымалысы объект идентификаторы рөлін атқарады. Ал деструктор объект жұмысын бітірген соң, **x** айнымалысының мәнін шығарады. **Destruct** типіндегі объектіні жасап және бірден жоятын **Generator()** әдісі ерекше қызығушылық туғызады. Алдымен **Program** класында **Destruct** типіндегі **ob** бастапқы объектісі жасалады, ал сонан соң біртіндеп кезекпен 100 мың объект жасалып өшіріледі. Осы процестің әртүрлі сәттерінде "қоқысты жинау" орын алады. Оның қаншалықты жиі орындалуы – бірнеше факторларға байланысты болады, оның ішінде бастапқы бос жады көлемі, пайдаланылатын операциялық жүйе типі және т.с.с. бар. Дегенмен белгілі бір сәтте деструктор арқылы қалыптасатын хабарламалар пайда бола бастайды. Егер де олар программа жұмысының соңына дейін, яғни "Дайын!" деген хабарлама шыққанша пайда болмаса, онда жасалынатын объектілердің шектеулі санын **for** цикліндегі қадамдар санын көбейте отырып арттыру керек.

Тағы бір маңызды ескерту: **~Destruct()** деструкторында **WriteLine()** әдісі тек осы мысалдың көрнекті болып көрінуі үшін ғана шақырылады. Көбінесе деструктор тек өз класында анықталған экземпляр айнымалыларына ғана әсер етуі керек.

Деструкторларды шақыру реттілігі дәлме-дәл анықталмағандықтан, оларды программаның орындалуы барысындағы белгілі бір сәтте жүзеге асатын әрекеттерді орындау үшін қолдануға болмайды. Дегенмен, "қоқысты жинауды" қолдап инициалдау көптеген жағдайларда ұсынылмайды, өйткені бұл программаның тиімділігін төмендетуге әкеліп соқтыруы мүмкін. Оның үстіне, "қоқысты жинау" жүйесінің өз ерекшеліктері бар – егер тіпті "қоқысты жинауды" тікелей орындауды сұратсақ та, нақты объектінің қашан жойылатынын бәрі бір алдын ала білуге болмайды.

Мысал 7.6. Кесінді класын құрыңыз. Класс кесіндінің екі ұшының координаталарын анықтайтын 4 өрістен тұрады. Келесі әдістерді жүзеге асырыңыз:

1. кесіндінің ұзындығын анықтау;
2. өрістерді инициалдауға арналған конструкторлар құру.

```
using System;
class Kesindi {
   public int x1, y1, x2, y2;
   public Kesindi() { x1 = y1 = 0; x2 = y2 = 1; }
   public Kesindi(int a, int b, int c, int d) {
      x1 = a; y1 = b; x2 = c; y2 = d;
   }
   private double uzyndygy() { return Math.Sqrt
          (Math.Pow(x2 - x1, 2) + Math.Pow(y2 - y1, 2)); }
   public void Shygaru()
   {
      Console.Write("kesindi koordinatalary: ({0}, {1}) zhane
          ({2}, {3}), onyn uzyndygy - {4:#.##}", x1, y1, x2,
          y2, uzyndygy());
   }
}
class Program {
   static void Main() {
      Kesindi A = new Kesindi();
      Kesindi B = new Kesindi(2, 2, 7, 8);
      Console.Write("A ");
      A.Shygaru();
      Console.WriteLine();
      Console.Write("B ");
      B.Shygaru();
      Console.WriteLine();
      Console.ReadKey();
   }
}
```

Программаның орындалу нәтижесі:

```
    A kesindi koordinatalary: (0, 0) zhane (1, 1),
onyn uzyndygy – 1,41
    B kesindi koordinatalary: (2, 2) zhane (7, 8),
onyn uzyndygy – 7,81
```

Объектілерді әдістерге беру

Бұған дейін әдістерге параметрлер бергенде **int** немесе **double** типтері қолданылды. Бірақ әдістерге сілтемелік типтегі параметрлерді де беруге болады, бұл үрдіс ОБП-да кең таралған. Келесі мысалда объектілерді сілтеме бойынша әдіске беру қарастырылады.

Мысал 1. Әдістерге сілтеме бойынша объектілер беру мысалы.

```
using System;
class Shenber {
   public int xcoord;
   public int ycoord;
   public double radius;
   public Shenber(int x, int y, int r)
   {  xcoord = x;
      ycoord = y;
      radius = r;
```

```
    }
    public void Shygaru() { Console.WriteLine("shenber centrinin
         koordinatalary: (" + xcoord + ", " + ycoord + "),
         shenber radiusy = " + radius);
    public bool qiylysu(Shenber ob) {
       double d = Math.Sqrt(Math.Pow(xcoord - ob.xcoord, 2) +
           Math.Pow(ycoord - ob.ycoord, 2));
       if (d < radius + ob.radius) return true;
       return false;
    }
}
class Program {
    static void Main() {
       Shenber shenber1 = new Shenber(2,2,1);
       Shenber shenber2 = new Shenber(6,6,2);
       Console.WriteLine("birinshi shenber parametrleri: ");
       shenber1.Shygaru();
       Console.WriteLine("ekinshi shenber parametrleri: ");
       Shenber2.Shygaru();
       bool b = shenber1.qiylysu(shenber2);
       if(b) Console.WriteLine("berilgen shenberler qiylysady");
       else
          Console.WriteLine("berilgen shenberler qiylyspaidy");
       Console.ReadKey();
    }
}
```

Программа орындалу нәтижесі :

```
    birinshi shenber parametrleri:
    shenber centrinin koordinatalary: (2, 2), shenber radiusy
= 1
    ekinshi shenber parametrleri:
    shenber centrinin koordinatalary: (6, 6), shenber radiusy
= 2
    berilgen shenberler qiylyspaidy
```

Бұл программадағы **qiylysu()** әдісі аргумент ретінде **Shenber** типіндегі объектіге сілтеме алады. Бұл әдіс шақырған объекті мен **ob** параметрі арқылы берілетін объектінің қиылысатынын тексереді. Егер екі объекті қиылысатын болса, бұл әдіс **true** мәнін қайтарады.

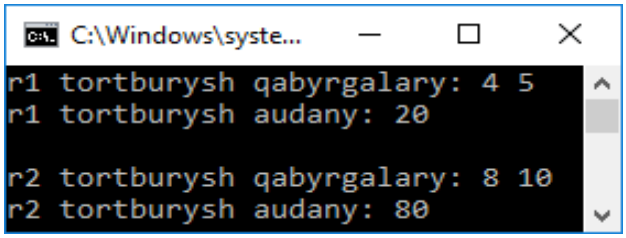
// Әдістен объектіні қайтару.

```
using System;

class Rect {
    int width;
    int height;

    public Rect(int w, int h) {
       width = w;
       height = h;
    }

    public int Area() {
       return width * height;
    }

    public void Show() {
```

```
            Console.WriteLine(width + " " + height);
        }


        /* Әдіс шақырылған объектідегі төртбұрышпен салыстырып
             қарағанда, қабырғалары берілген  коэффициентке
             пропорционал үлкейтілген төртбұрышты қайтарады. */
        public Rect Enlarge(int factor) {
            return new Rect(width * factor, height * factor);
        }
    }

    class RetObj {
    static void Main() {
        Rect r1 = new Rect(4, 5);

        Console.Write("r1 tortburysh qabyrgalary: ");
        r1.Show();
        Console.WriteLine("r1 tortburysh audany: " +  r1.Area());

        Console.WriteLine();

        // r1 tortburyshynan eki ese ulken tortburysh turgyzu.
        Rect r2 = r1.Enlarge(2);
        Console.Write("r2 tortburysh qabyrgalary : ");
        r2.Show();
        Console.WriteLine("r2 tortburysh audany : " + r2.Area());
    }
}
```

Бұл программа нәтижесі:

```
r1 tortburysh qabyrgalary:  4  5
r1 tortburysh audany : 20

r2 tortburysh qabyrgalary:  8  10
r2 tortburysh audany : 80
```



Әдіс объектіні қайтарғанда, объект оған сілтеме болғанша жұмыс істей береді. Сонан кейін объект "қоқыс" ретінде жойылады. Сол себепті объектіні құрған әдіс аяқталғанмен, объект жойылмайды. Объект типіндегі қайтарылатын мәліметтердің практикалық мысалы ретінде *класс фабрикасы* қолданылады. Ол өз класындағы объектілер тұрғызуға арналған әдіс болып табылады.