8 КЛАСС ЭЛЕМЕНТТЕРІ

8.1 Конструкторлар

Конструктордың негізгі қызметі – объектті құру және осы объектің деректер элементтеріне белгілі бір бастапқы мәндерді меншіктеу. Кейде бұл процесс объект инициализациясы деп аталады.

Объектті құру конструктордың негізгі міндеті болып табылады.

Объекттің деректер элементтеріне кейбір мәндерді меншіктеу түрлі жолдармен орындалуы мүмкін. Объекттегі элементтер мәндерінің анықталуына қарай конструкторлардың мына түрлері болады:

– параметрсіз конструкторлар;

– параметрлері берілген конструкторлар;

– жиынтық (множественные) конструкторлар.

Параметрсіз конструкторлар бағдарламада параметрсіз шақырылады. Мысалы,

```
treyg t1 = new treyg();
```

Жоғарыдағы мысалда t1 объектісінің деректер элементтеріне белгіленген мәндер меншіктеледі (әдетте нөлдік мәндер).

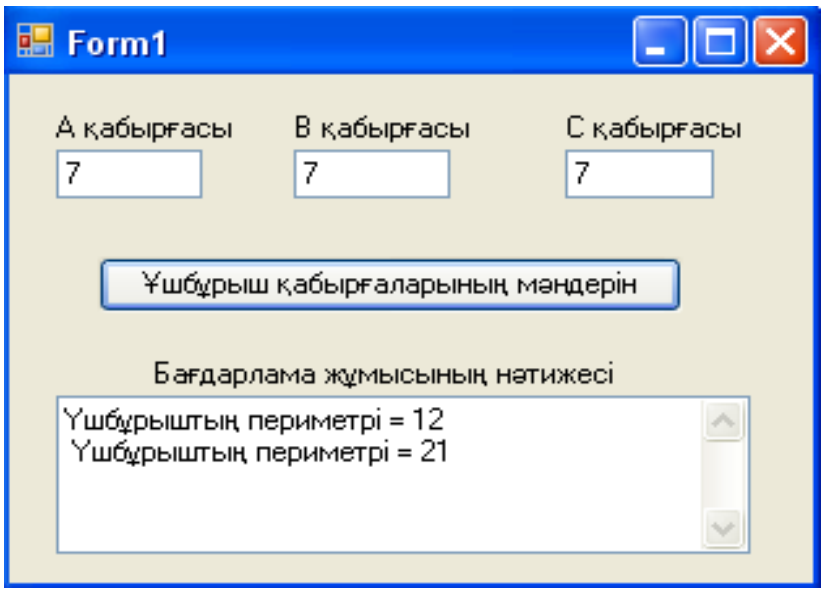
Осындай конструктор жүзеге асырылғанда деректер өрістеріне белгілі бір мәндер жиынын меншіктейтін операторлар қолданылуы мүмкін.

Мысалы, `treyg` класына параметрсіз конструкторды қосуға болады, конструктор үшбұрыш қабырғаларына белгіленген 3, 4 және 5 мәндерін меншіктейді. Осы жағдайды объект периметрін экранға шығарып тексеруге болады:

```
public treyg()
 {
 a = 3; b = 4; c = 5;
 }
 public void print0()
 {
  p = a + b + c;
  ss = "Үшбұрыштың периметрі = " + p.ToString();
 }
```

Объектті құру барысында объекттің деректер элементтерінің бастапқы мәндерін параметрлі конструкторлар анықтауға мүмкіндік береді. Осындай конструкторлардың класта жариялануын мына түрде көрсетуге болады:

```
public treyg(int sa, int sb, int sc)
 {
 a = sa; b = sb; c = sc;
 }
```



8.1-сурет – Қосымшада белгіленген бастапқы мәндері бар конструкторды қолдану

Көптеген бағдарламашылар конструкторды құрған кезде оған объекттердің элементтер мәндерінің дұрыстығын тексеруді қосады, ондай конструкторлар «ақылды» конструкторлар деп аталады. Мысалы, үшбұрыш класының конструкторын дайындаған кезде деректер элементтері бойынша мына тексеру жұмыстарын қосуға болады:

– үшбұрыштың барлық қабырғалары 0-ден үлкен болуы тиіс;
– үшбұрыштың кез келген екі қабырғасының қосындысы үшінші қабырғасынан үлкен болуы керек.

Егер шарттар орындалмайтын болса, хабарламаны шығарып, объект деректерінің элементтеріне белгіленген мәндерді меншіктеу керек немесе мәндерді қайта енгізуді орындау керек.

Осындай «ақылды» конструкторды жариялаудың қарапайым конструкторды жариялаудан айырмашылығы жоқ, ал оны құрған кезде класс деректерін енгізу әдісінде қажетті барлық тексерістер жүргізіледі, мысалы:

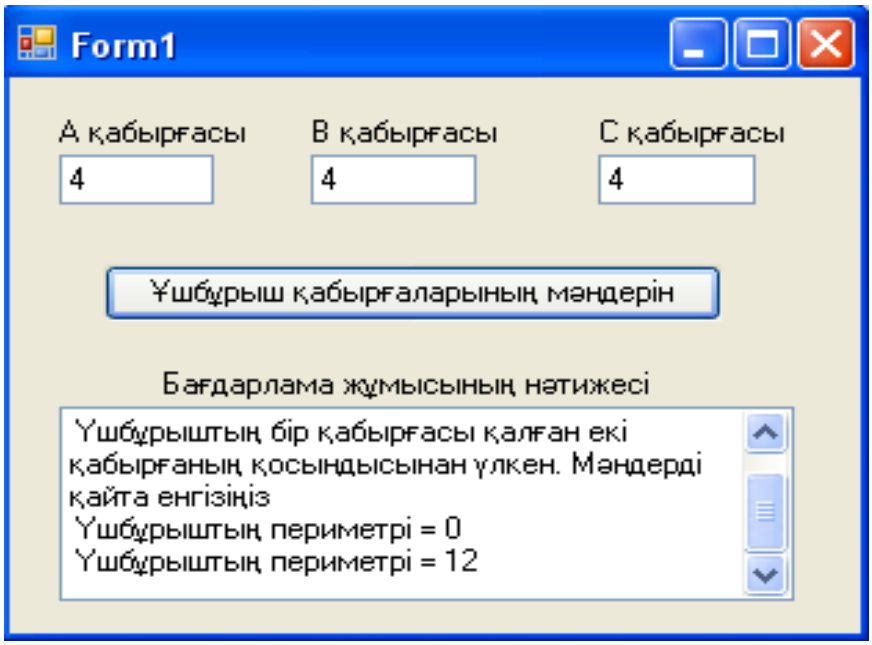
```
public treyg(int sa, int sb, int sc)
 {
 vvod(sa,sb,sc);
}
```

Оқиға өңдеуішінің бағдарлама кодын келесі түрде өзгертейік – конструкторға алдын ала үшбұрыш қабырғаларының қате мәндерін енгізейік:

```
private void button1_Click(object sender, EventArgs e)
 {
 int A, B, C;
 treyg t = new treyg(3,5,9);
 t.printO();
 A = Convert.ToInt32(textBox1.Text);
 B = Convert.ToInt32(textBox2.Text);
```

```
 C = Convert.ToInt32(textBox3.Text);
 t.vvod(A,B,C);
 textBox4.Text = t.ss;
}
```

Бағдарламаның жұмысы 8.2-суретінде көрсетілген.



8.2-сурет – Конструкторға мәндерді алдын-ала қате берген жағдайдағы қосымшаның жұмысы

Жиынтық конструкторлар түрлі типтегі аргументтерді өңдеген кезде қайта жүктелетін функцияны қолданады.

Осындай конструкторлар деректердің мәндері әртүрлі формада берілгенде қолданылады, мысалы, бүтін, нақты сандармен немесе жолмен берілсе.

Әдетте мұндай конструкторлар датаны өңдеу үшін қолданылады.

Қайта анықтау функциясын қолдана отырып класс сипаттамасына түрлі типтегі деректерді «түсінетін» бірнеше конструкторларды қосуға болады.

Мысалы, ағымдағы датаны түрлі тәсілдермен енгізуге болады. Мысалы, датаны енгізудің үш нұсқасы бар:

- бүтін сандармен (айы, күні, жылы – 23 12 07);
- сөздермен (23 желтоқсан 2007 жыл);
- қосымша символдармен қосып жазу (20.10.07 немесе 17/09/2007).

```
class date
 {
 . . .
 date() { . . . }
 date(int mm, int dd, int gg) { . . . }
 date (string tekct) { . . . }
 . . .
```

```
};
```

Объектіні инициализациялау кезінде кез келген ұсынылған нұсқаны қолдануға болады.

8.2 Деструкторлар

C# тілінде кластарда арнайы әдістер – деструкторлар болуы мүмкін. Қызметі жағынан қарастырғанда олар конструктордың әрекеттеріне қарама-қарсы әрекеттерді орындауы керек. Бірақ C# тіліндегі деструкторлардың ерекшелігі – олар сәйкес объекттердің конструкторына бөлінген жадыны босатуды орындамайды (оны қоқыс жинаушы бақылайды), бірақ объектке бөлінген ресурстарды босатады, мысалы, деректер базасының сервермен, басқа компьютермен байланысты тоқтатады.

Конструктор сияқты деструктордың атауы кластың атауымен бірдей болады, бірақ деструктор атауының алдында «~» – тильда символы қойылады. Мысалы, егер treyg класында деструктор болса, онда оның жазылуы мына түрде болады:

```
public ~treyg()
 { деструктор_денесі }
```

Деструкторды бағдарламада тікелей шақыруға болмайды, оны үйіндіден объектті жойған кезде автоматты түрде «қоқыс» жинаушысы шақырады. Іс жүзінде объектке бөлінген ресурстардың қажеттілігі жоқ болғаннан кейін ресурстарды босату автоматты түрде орындалады. Сондықтан деструкторлар класс құрылымында конструктивті түрде болады, бірақ әдетте құрылмайды, әдепкідей қолданылады.

8.3 Қасиеттер

Объекті-бағытталған бағдарламалау принциптерінің бірі – инкапсуляция. Инкапсуляция дегеніміз – деректерді қосымшадан тікелей қол жеткізуден қорғау мақсатында класс өрістері мен әдістерінің бірлесуі. Объект өрістері объект интерфейсі - қол жеткізу ережелер жинағы немесе қасиеттер арқылы қолданылады. Инкапсуляция («капсула» сөзінен) – объект өрістерін жасыру қасиеттер арқылы жүзеге асырылады.

Қасиеттер get() және set() арнайы екі әдістерінен және объект өрісінен тұрады. Get() әдісі объекттің сәйкес өрісінің ағымдағы мәнін қайтарады, ал set() әдісі объект өрісіне жаңа мәнді енгізеді.

Қосымша «қасиеттің» мәнін алғанда немесе өзгерткенде осы әдістер автоматты түрде шақырылады. Синтаксисі бойынша қасиеттердің өрістерден өзгешелігі жоқ, олар меншіктеу операторының сол жағында орналаса алады немесе оператордың оң жағында өрнектің мүшесі ретінде жазыла алады. Мысалы, int a жабық өрісіне бүтін типтегі келесі Aa қасиетін жазуға болады.

```
public class treyg
```

```
{
private int a, b, c, p;
public int Aa
{
get { return a;}
set { a = value;}
}
public string ss;
. . .
```

Қосымшада t объектісін құрғаннан кейін a өрісіне Aa қасиетінінің көмегімен жаңа мәнді меншіктеуге болады:

```
int A, B, C;
 treyg t = new treyg();
 A = Convert.ToInt32(textBox1.Text);
 B = Convert.ToInt32(textBox2.Text);
 C = Convert.ToInt32(textBox3.Text);
t.Aa = A; ,
```

немесе A айнымалысына t объектісінің `a` өрісінің мәнін меншіктеуді мына түрде жазуға болады:

```
A = t.Aa; ,
```

мұнда t.Aa – treyg класының қасиеті.

Бұл мысалда t.Aa қасиеті арқылы біз `treyg` класының жабық a өрісін ашық өріске айналдырдық. Бұны орындаудың жеңіл жолы – класта private спецификаторының орнына public спецификатор жариялау.

Әлбетте қасиеттер кластың жабық өрістеріне қол жеткізуге арналған.

Қарастырылған мысалда класс өрісіне жаңа мәнді жазу қасиетіне біз мына шарттарды қоса аламыз: мәннің 0-ден үлкен болу және өріске тек бір рет жазуды орандау мүмкіндігі, мысалы:

```
set { if (a == 0 && value > 0) a = value;}
```

Қасиеттің кез келген әдістерінің бірі болмауы мүмкін (бірақ, екеуі емес). Ондай жағдайда бізде тек қана оқуға немесе тек қана жазуға арналған қасиеттер болады.

8.4 Сілтемесі параметрі - This

Арнайы нұсқағыш өрісті (this) қарастырайық, ол объект құрылған кезде автоматты түрде құрылады және осы объекттің адресін сақтайды.

Іс жүзінде объект өрістері мен класс әдістерінің байланысы this параметрі арқылы орнатылады. Кластың әрбір әдісі ағымдағы объект элементтерімен жұмыс жасауы үшін this параметрін тікелей қолдана алады. `This` мәні үнемі ағымдағы объектке (қосымшадағы ағымдағы объект) сәйкес болғандықтан, класс әдістері ағымдағы объект элементтерімен жұмыс жасайды.

Көптеген конструкторларда объект өрістерін инициализациялау үшін this параметрі қолданылады. Авторлар класс конструкторын сипаттаған кезде формалды параметрлердің атаулары ретінде класс өрістерінің атауларын қолданады. Сондықтан класс өрістері мен формалды параметрлердің атауларын ажырату үшін класс өрістері атауларының алдына this параметрі жазылады. Мысалы, treyg конструкторы мына түрде жазылады:

```
public treyg(int a, int b, int c)
 {
 this.a = a; this.b = b; this.c = c;
 }
```

Атауларға байланысты түсінбеушіліктерді жоюдың бір жолы – түрлі формалды параметрлерге басқа атауларды беру (бағдарламада осы тәсіл қолданылады).

Кластың статикалық элементтерін қолданған кезде this параметрін пайдалануға болмайды, өйткені олар нақты бір объектке емес жалпы класқа тиісті болады.

C# тілінде base сілтемесі бойынша тағы да бір параметр қоланылады, ол базалық объектпен жұмыс істеу үшін пайдаланылады. Егер Form1.Designer.cs файлының кодын мұқият қарасаңыз, void Dispose(bool disposing) әдісінде жазылған кодтың соңғы жолдарында base.Dispose(disposing); жазбасы бар. Бұл жазба бойынша жадыдан объект жойылған кезде, базалық объект те жойылады.

8.5 Класс оқиғалары

ОББ маңызды құрамдас бөлігі – класта жүзеге асырылған оқиғалар механизмі. Осы механизм арқылы бір объект (оқиға көзі) басқа объектке (оқиғаны қабылдаушы) өз жағдайының өзгергені жөнінде хабарлай алады.

Әдетте оқиға тетігі көпағынды процестердегі синхронизация кезінде қолданылады, бұл дегеніміз - осы ағындар жұмысының кезектілігін реттеу.

Бірақ осы механизмді Windows-қосымшаларда қолдануға болады, онда батырма, жалаушалар, т.б. элементтер пайдаланушының онымен байланысы туралы ақпаратты шығарады. Мысалы, барлық объекттерді – батырмаларды (Button класының) тышқанмен шерткен кезде олар OnClick оқиғасын туындатады.

Формада орналасқан әрбір басқару элементінің белгілі оқиғалар жиыны болады, оқиғалардың «бос» өңдеуіштерін элементтің қасиеттер терезесі арқылы ашуға болады. Алайда бағдарламашылар өздерінің арнайы оқиғалар өңдеуіштерін жаза алады. Оқиғалар механизмін жүзеге асыру үшін, яғни белгілі бір кластың клиенттеріне класс оқиғасының пайда болу жайлы хабарлау үшін оқиға көзі мыналарды орындауы керек:

– оқиғаны класс мүшесі ретінде жариялау (өрістермен, әдістермен, қасиеттермен бірге) – жариялау үшін Event қызметтік сөзі қолданылады;

– керекті мезетте класс клиенттеріне (оқиғаны қабылдаушы) орын алған оқиға туралы ақпаратты жеткізу, қажетті параметрлерді көрсету;

– клиенттен жауапты алу, оны талдау, оқиғаға байланысты әрекеттерді орындау.

Соңғы екі операция (клиентпен ақпарат алмасу) әдетте делегаттар арқылы жүзеге асырылады, оларды біз келесі бөлімдерде қарастырамыз.

8.6 Класс операцияларын қайта анықтау

Кейбір кластарда жазылатын код үзіндісінің (блоктар) тақырыбында класс атауынан (бірақ конструктор емес) кейін, бірақ дөңгелек жақшаларда жазылатын формалды параметрлерге дейін `operator` қызметтік сөзі қолданылады. `Operator` қызметтік сөзі арқылы операцияларды қайта анықтау механизмі жазылады, операцияларды қайта анықтау қарапайым математикалық өрнектерде класс типіндегі айнымалыларды қолдануға мүмкіндік береді. Мысалы, «студент» класы үшін қосу операциясы қайта анықталған болса (operator+) және бағдарламада ct1, ct2 – «студент» типіндегі екі объект құрылған болса, онда ct1 + ct2 өрнегін жазуға болады.

Екі объектіні қосу нені білдіреді? Мүмкін біз ондағы аттарды біріктірген немесе жас шамаларын қосқан болармыз?

Міне, операцияларды қайта анықтау осы мақсатта қолданылады, онда екі объектіні бір-біріне қосқанда нені қосу керектігі анық жазылады.

Operator қызметтік сөзі операцияға қайта анықтау орындалатынын, ал « + » операциясы болса, қосу операциясының қайта анықталатынын көрсетеді.

Operator сөзін пайдаланып кластың меншікті операцияларын анықтау операцияларды қайта анықтау деп аталады. Операцияларды қайта анықтау әдетте математикалық, физикалық ұғымдарды сипаттайтын кластар үшін қолданылады.

Кластың операциясын анықтау арнайы әдістердің (функциялар - операциялар) түрлері арқылы сипатталады.

Операцияны қайта жүктеудің жазылу пішімі мына түрде болады:

```
[спецификаторлар] класс атауы operator операция атауы (формалды параметрлер) {денесі }
```

Спецификатор ретінде әдетте public және static қызметтік сөздері бірге қолданылады. Сонымен қатар, операцияны сыртқы операция түрінде жариялауға болады (extern).

Операцияны өрнектерде қолданғанда орындалатын әрекеттер операция денесінде (басқа әдістердің денесі сияқты блок) анықталады.

Мысалы:

```
public static int operator+ (Stydent S1, Stydent S2)
  { return S1.Ocenka + S2.Ocenka; }
```

Егер біз белгілі бір класс үшін операцияны қайта анықтауды енгізетін болсақ, онда оның әрекеттері тек осы кластың деректеріне ғана қолданылады.

Ескерту, операцияны қайта анықтау жаңа операцияны құрмайды, тек операцияларды класс типіндегі деректерге бейімдейді.

Операцияны қайта анықтауды сипаттаған кезде келесі ережелерді сақтау керек:

– операция кластың ашық статикалық әдісі түрінде сипатталуы керек (public static спецификаторлары);

– формалды параметрлерді операцияға мәндері бойынша жіберілуі тиіс (яғни ref немесе out қызметтік сөздері қолданылмайды);

– кластың барлық операцияларының жазылу форматтары әр түрлі болуы керек.

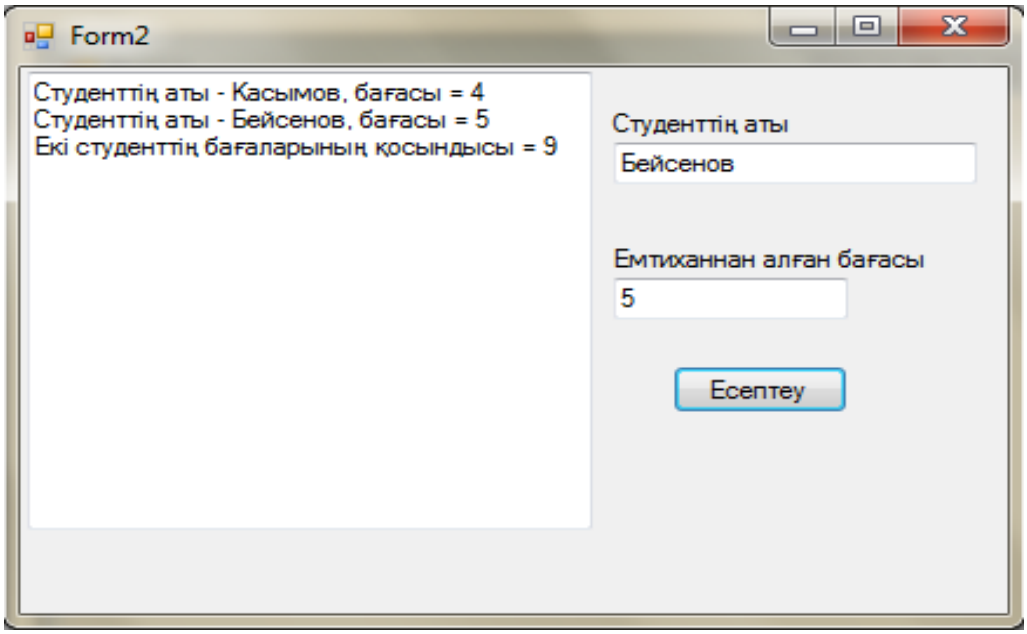
Әдетте операцияны қайта анықтау пайдаланушы анықтайтын типтерге арналған өрнектер синтаксисінің көрнекілігін қамтамасыз ету үшін қолданылады.

Бірінші тұрған екі студенттің бағалар қосындысын есептеу үшін операцияны қайта анықтауды қолдану мысалы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
 public partial class Form2 : Form
 {
 public class Stydent
 { int Ocenka;
  string Name;
  public string ss;
  public int Aa
  {
  get { return Ocenka; }
  set { if (value >= 2 && value <= 5) Ocenka = value; else
ss = "Бағаның мәнін дұрыс енгізіңіз \r\n"; }
  }
  public void Vvod(string name)
  {
  Name = name;
  }
 public static int operator +(Stydent S1, Stydent S2)
  {
  return S1.Ocenka + S2.Ocenka;
  }
 }
 public Stydent[] styd = new Stydent[5];
 public static int n = 0;
```

```
        public Form2()
        {
         InitializeComponent();
         textBox1.Text = "";
         n = 0;
        }
        private void button1_Click(object sender, EventArgs e)
        { int Oc;
         string fio, In;
         Stydent st = new Stydent();
         fio = textBox3.Text;
         st.Vvod(fio);
         In = textBox2.Text;
         Oc = Convert.ToInt32(In);
         st.Aa = Oc;
         styd[n] = st;
         textBox1.AppendText("Студенттің аты - " + fio + ", бағасы
= " + styd[n].Aa.ToString() + "\r\n");
         if (st.Aa != 0) n++;
         if (n == 2) textBox1.AppendText("Екі студенттің
бағаларының қосындысы = " + (styd[0] + styd[1]).ToString() +
"\r\n");
        }
        }
      }
```

Бағдарлама жұмысы 8.3-суретте көрсетілген.



8.3-сурет – Қосу операциясын қайта анықтау мысалы

8.7 Өзін-өзі тексеру сұрақтары

1 Параметрсіз конструктор не үшін қолданылады?

2 Параметрлері берілген конструктор не үшін қолданылады?

3 Атаулары бірдей болатын әдістерді анықтау процесін қалай атайды?

4 Бір кластың бірнеше конструкторлары қалай аталады?

5 tka класының деструкторы қашан шақырылады?

6 Егер қайтарылатын мәнінің типі void болып жарияланса, әдіс қалай аталады?

7 C# тілінде типі void емес әдіс қалай аяқталуы тиіс?

8 Әдістің қандай формалды параметрлері сілтемелік параметрлер деп аталады?

9 Өрістер мен оларды өңдеу әдістерінің бір құрылымда бірігуі қалай аталады?.

10 Кластың жабық өрістеріне қол жеткізу үшін кластарда қандай механизм қолданылады.