11 ИНТЕРФЕЙСТЕРДІ ПАЙДАЛАНУ

11.1 Интерфейс ұғымы

"Интерфейс" сөзінің мағынасы көп, түрлі жағдайларда әр түрлі мағынада болуы мүмкін. Бағдарламалық және аппараттық интерфейс ұғымдары бар, бірақ көп жағдайларда интерфейс сөзі объекттер немесе процестер арасындағы байланысты анықтайды. Осы бөлімде интерфейс ұғымы қарастырылатын болады (интерфейс interface сөзінен кейін жазылады). Интерфейс – кластың дербес түрі.

Интерфейс ұғымы толығымен абстрактылы класты сипаттайды, ондағы әдістер толығымен абстрактылы болады.

Интерфейстің абстрактылы кластан айырмашылығы синтаксисіндегі және орындалу тәртібіндегі өзгешеліктерінде болады.

Синтаксистік айырмашылығы мынада: интерфейс әдістері қол жеткізу модификаторынсыз жарияланады.

Орындалу тәртібіндегі өзгешелігі – ұрпақтарға қойылатын талаптардың қатаң болуында. Интерфейсті мұраланатын класс (интерфейсті класс) интерфейстің барлық әдістерін толық жүзеге асыруы тиіс. Жоғарыда сипатталған барлық өзгешеліктер интерфейстің абстрактылы класты мұраланатын кластан айырмашылықтарын көрсетіп тұр. Абстрактылы кластың ұрпағы (класы) аталық абстрактылы кластың кейбір әдістерін ғана орындай алады.

Интерфейстік кластың қарапайым кластан айырмашылығы – бірнеше түпкі интерфейстерді мұралана алуында. Сонымен, C# тілінде интерфейстік кластарда ғана бірнеше рет мұралануға рұқсат берілген.

Класс атауы мен қос нүктеден кейін түпкі интерфейстер тізім бойынша бір бірден көрсетіледі:

```
    public interface INewClass: IInt1, IInt2, ...,
IIntN
    { . . .}
```

Осындай интерфейстік кластарда түпкі интерфейстердің барлық әдістерінің іске асырылуы болуы керек.

Ескерту, интерфейстік класс интерфейстермен қатар қарапайым бір класты (тек қана бір класты!) мұралануы мүмкін, қарапайым класты мұраланғанда ол әдеттегі мұрагер ретінде оның әдістерін өзгерте алады және өрістерді қоса алады т.б.

Бірнеше рет мұралануды қолданғанда атаулар арасындағы қателіктер немесе ортақ түпкі кластың болуына байланысты қателіктер кетуі мүмкін.

Атаулар арасындағы шиеленіс мына жағдайларда пайда болады: әр түрлі түпкі интерфейстерде синтаксистері бірдей аттас әдістер болуы мүмкін.

Интерфейстік класс түпкі интерфейстердің барлық әдістерін жүзеге асыруы керек болғандықтан коллизия пайда болады, оны шешудің бірнеше жолы бар.

Әдістерді желімдеу. Интерфейстік класс аттас әдістердің орындалуын бірдей деп қарастырады және түпкі кластардың барлық аттас әдістерін іске асыру үшін бір әдісті жариялайды.

Әдістерге басқа атау беру. Егер аттас әдістердің жүзеге асырылуы әр түрлі болуы керек болса, онда әдістердің атаулары өзгертіледі.

Интерфейсті абстрактылы кластан ерекшелендіретін тағы да бір маңызды қызметі бар. Абстрактылы класс класты жобалаудың алғашқы кезеңі болып табылады. Әрбір интерфейс класқа жаңа қосымша қасиеттерді қосады.

11.2 Интерфейс синтаксисі

Міндетті емес элементтері (олар квадрат жақшалармен бөлінген) бар интерфейстің жалпы сипаттамасының жазылу пішімі мына түрде болады:

```
    [ атрибуттар ] [ спецификаторлар ] interface
класс_атауы [ : түпкі кластар]
     { класс_денесі } ,
```

мұнда,

атрибуттар – класс туралы қосымша ақпаратты анықтайды;

спецификаторлар – кластың құрамдас бөліктеріне қол жеткізу шарттарын анықтайды;

түпкі кластар – класс мұраланатын аталық түпкі кластар;

класс денесі – интерфейстік кластың құрамын анықтайды.

Егер интерфейстің жазылу пішіміне назар салып қарасақ, онда оның пішімі қарапайым кластың жазылу пішіміне ұқсас екенін байқауға болады. Өйткені интерфейс кластың бір түрі болып табылады.

.NET платформасының кітапханасында интерфейстердің көптеген түрлері бар, оларды класс мұраланған кезде қосымша қасиеттерге ие болады.

Мысалы, IComparable интерфейсін қосқан кезде біз объект деректерінінің өрістерін анықтаймыз (күйге келтіреміз), оларға объекттерді салыстыру әдістерін қосуға болады. Тек IComparable интерфейсін қосқаннан кейін ғана класс объекттерінде белгілі бір өріс бойынша сұрыптауды жүргізуге болады.

IEnumerable және IEnumerator интерфейстерінің іске қосылуы foreach конструкциясының көмегімен объект ішіндегісін қарап шығу мүмкіндігін, ал Icloneable интерфейсі объекттерді клондау мүмкіндігін береді.

Әрбір интерфейс класты белгілі бір жаңа мүмкіншіліктерді қосыды.

Мысалы, интерфейсті сауда үшін, яғни ағымдағы бағамға сәйкес валютаны сату-сатып алу үшін немесе жеңілдіктерді ескере отырып коммуналдық қызметтер бойынша түрлі есептеулер үшін құруға болады.

Мысал ретінде мына интерфейсті құрайық: интерфейс әдістерін жүзеге асырғанда кластарға музыкалық жазбаларды түрленіруге мүмкіндік береді – 7 нота мен дыбыс үзілісін 0 мен 7 аралығындағы цифрлерге түрлендіру және кері түрлендіруді орындау.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 interface ITextNoti
 {
  string Codirovanie();
  string Decodirovanie();
 }
 class MyzikText : ITextNoti
 {
  string text;
  static string[] codeTable =
 {
 "до", "ре", "ми", "фа", "соль", "ля", "си", "пауза"
 };
  //Конструктор
  public MyzikText(string txt)
  {
  text = txt;
  }
  //Интерфейстерді құру
  public string Codirovanie()
  {
  Boolean ok;
  string rez = "";
  string[] noti;
  //Төменгі регистрге түрлендіру
  string text1 = text.ToLower();
  // noti массивінің өлшемі Split әдісі қайтаратын массивтің
өлшеміне сай болады
  noti = text1.Split(' ');
  for (int i = 0; i < noti.Length; i++)
  {
   ok = false;
   for (int j = 0; j < 8; j++)
   if (noti[i] == codeTable[j])
   { ok = true; rez = rez + " " + j.ToString(); }
   if (ok == false) rez = rez + " ?";
  }
  return rez;
  }
```

```
 //нот кестесін қолданып text өрісінің шифрын анықтау
 public string Decodirovanie()
 {
 Boolean ok;
 string rez = "";
 string[] noti;
 // Төменгі регистрге түрлендіру
 string text1 = text.ToLower();
 noti = text1.Split(' ');
 for (int i = 0; i < noti.Length; i++)
 {
  ok = false;
  for (int j = 0; j < 8; j++)
  if (Convert.ToInt32(noti[i]) == j)
  { ok = true; rez = rez + " " + codeTable[j]; }
  if (ok == false) rez = rez + " ?";
 }
 return rez;
 }
}
public Form1()
{
 InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
 string a, b;
 a = textBox1.Text;
 MyzikText IcxodText = new MyzikText(a);
 b = IcxodText.Codirovanie();
 textBox3.AppendText(b + "\r\n");
}
private void button2_Click(object sender, EventArgs e)
{
 string a, b;
 a = textBox2.Text;
 MyzikText IcxodText1 = new MyzikText(a);
 b = IcxodText1.Decodirovanie();
 textBox3.AppendText(b + "\r\n");
}

 }
}
```

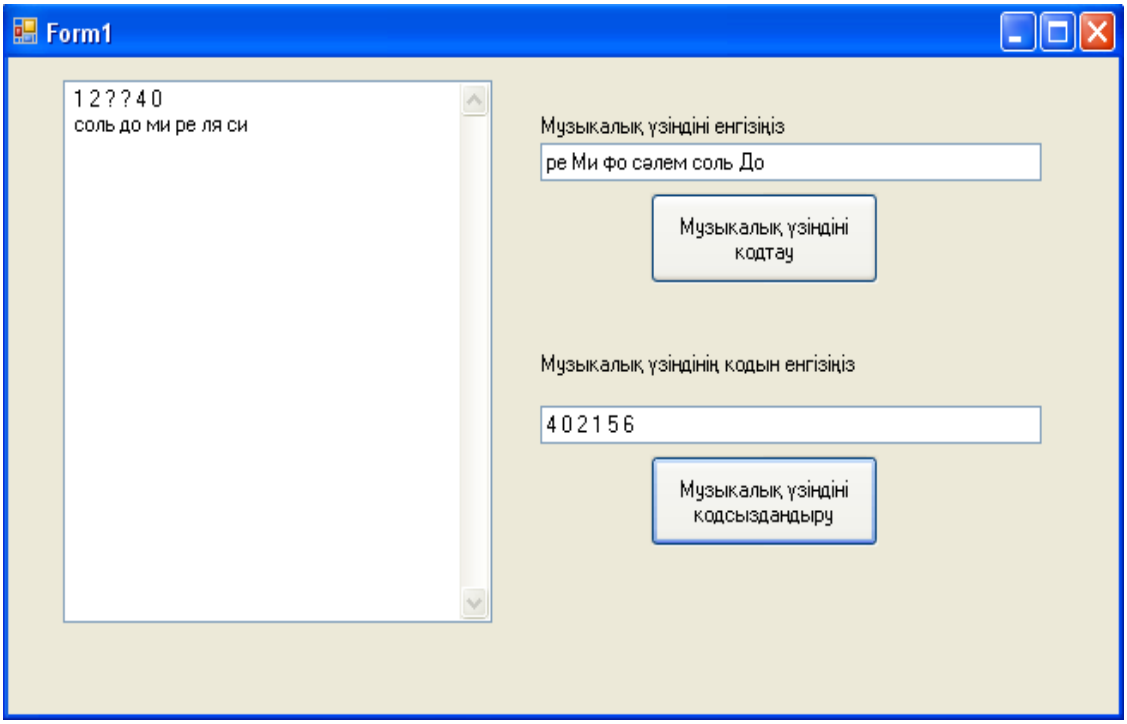
ItextNoti интерфейсін жариялаймыз, оның екі әдісі бар: кодтау (нот мәтіні 0 мен 7 аралығындағы сандармен алмастырылады) және декодтау (0 мен 7 аралығындағы сандармен берілген мәтін нот мәтінімен алмастырылады).

```
interface ITextNoti
    {
        string Codirovanie();
```

```
        string Decodirovanie();
    }
```

Одан кейін интерфейстік класты жариялаймыз, ол интерфейсті мұраланады және оның әдістерін орындайды. Қол жетімді интерфейс әдістерін іске асырамыз. Қосымшаның кодында интерфейс әдістерінің орындалу алгоритміне түсіндірме берілген, сондықтан қосымша түсініктемені беруді қажет етпейді.

Бағдарлама жұмысы 11.1-суретте көрсетілген.



11.1-сурет – Интерфейстік класты қолдану

Қарастырылған мысалда интерфейс пен интерфейстік класты құру және қолдану технологиясы көрсетілген.

11.3 IEnumerable стандартты интерфейсін қолдану

Бір қарағанда интерфейстік класты енгізудің артықшылықтары жоқ болып көрінеді – кодтау, декодтау әдістерін тікелей MyzikText класына орналастыруға болады.

.NET платформасының кітапханасындағы кластарда әр түлі интерфейстердің интерфейстік әдістерінің көптеген саны бар, оларды класс мұраланып қосымша қасиеттерге ие болады (әдістердің орындалуы арқылы емес, олардың атаулары арқылы). Әдетте әрбір интерфейстік класс интерфейстік әдістерді жүзеге асыруын өзі орындауы керек.

Мысалы, егер класта foreach циклінің көмегімен кейбір массив түрінде берілген тізімдеулі объекттерді қарап шығуды ұйымдастыру керек болса,

онда біздің класс IEnumerable (тізімдеулі, перечислимый) интерфейсінің мұрагері болуы керек. Осы интерфейстің бір ғана GetEnumerator() әдісі болады және осы әдіс Enumerator (тізімдеуші, перечислитель) типіндегі объекті қайтарады. GetEnumerator() әдісінің жазылу пішімі келесі түрде болады:

```
IEnumerator GetEnumerator();
```

Сонымен, осы клас IEnumerable және IEnumerator интерфейстерінің мұрагері болуы керек.

IEnumerator интерфейсінің тізімделген кезекті объекті қайтаратын Object Current{get;} деген бір қасиеті және екі әдісі болады. Bool MoveNext() әдісі тізімдеушіні келесі тізімделген объектке жылжытады, void Reset() әдісі тізімдеушіні бірінші тізімделген объектке орнатады.

Жоғарыда келтірілген қасиет пен екі әдіс Foreach циклінің көмегімен массив объекттерін қарап шығу процессін ұйымдастыруға мүмкіндік береді. Осы интерфейстердің әдістері виртуалды коллекциямен жұмыс жасайды және осы жағдай олардың әмбебаптығын анықтайды.

Егер класта объекттерді салыстыру керек болса, мысалы, объекттерді сұрыптау керек болса, онда ондай класты IComparable интерфейсінің мұрагері етіп жариялау керек. Осы интерфейстің бір ғана әдісі болады - CompareTo(object obj), ол әдіс "үлкен", "кіші" немесе "тең" қатынастарының орындалуына байланысты оң, теріс немесе нөлге тең бүтін санды қайтарады.

IEnumerable және IEnumerator интерфейстерімен жұмыс жасауды қарастырайық. Осы мысал бойынша дүкендегі тауарларды қарап шығуды ұйымдастыру керек. Tovar класының екі өрісі болады – тауардың атауы және оның бағасы.

```
class Tovar
 {
  public string Naz; // Тауардың атауы мен бағасы
  public int Cena;
  public Tovar(string n, int c) // Конструкор
  {
  Naz = n; Cena = c;
  }
 }
```

Tovar типіндегі объекттерді сақтау үшін Cklad класын қолданамыз, құрылымы мынандай:

```
class Cklad
{
 public Tovar[] tovar; // тауарлар массиві
 public Cklad() // қойма конструкторы
 {
 tovar = new Tovar[4];
 }
}
```

Қоймада сақтауға болатын объекттердің максималь саны 4-ке тең.

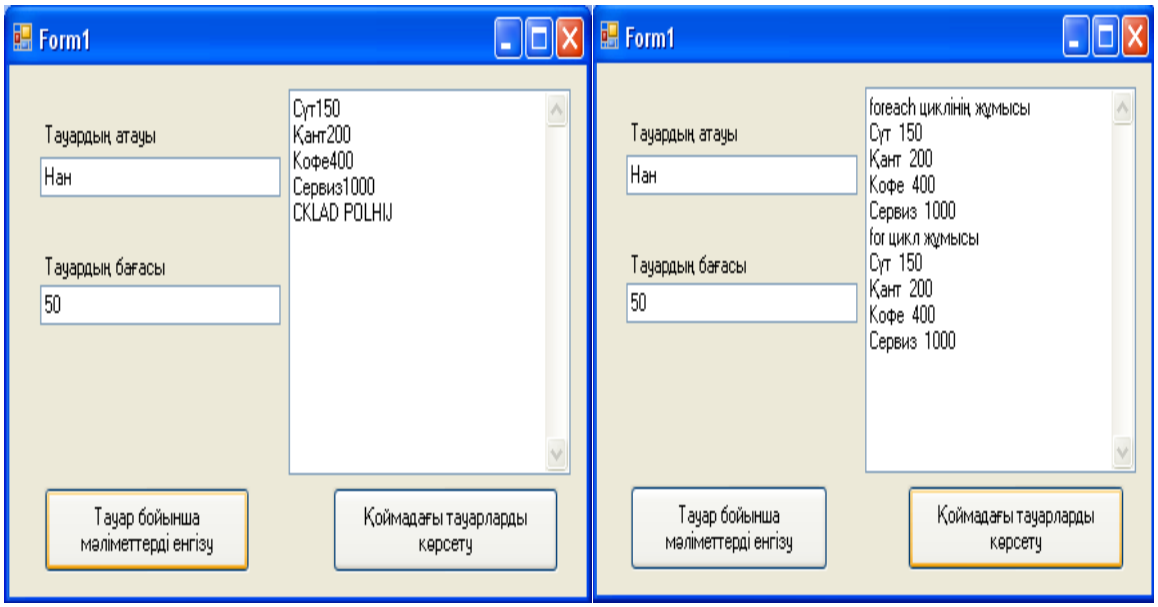
Қосымшада IEnumerable интерфейсін қолданбаймыз.

Form1.cs файлының коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public static string s;
 public static int kol;
 class Tovar
 {
  public string Naz; // Тауардың атауы мен бағасы
  public int Cena;
  public Tovar(string n, int c) // Конструкор
  {Naz = n; Cena = c;
  }
 }
 class Cklad
 {
  public Tovar[] tovar; // Тауар бойынша массив
  public Cklad() // қойма конструкторы
  {
  tovar = new Tovar[4];
  }
 }
 public Form1()
 { InitializeComponent();
  kol = 0;
  s = "";
 }
 Cklad ckl = new Cklad();
 private void button2_Click(object sender, EventArgs e)
 {
  if (kol < 4)
  {
  ckl.tovar[kol] = new Tovar(textBox1.Text,
Convert.ToInt32(textBox2.Text));
  s = s + textBox1.Text + textBox2.Text + "\r\n";
  }
  else { s = s + "Қойма толды" + "\r\n"; kol--; }
  kol++;
  textBox3.Text = s;
 }
 private void button1_Click(object sender, EventArgs e)
 {
  s = "";
  s = "foreach циклінің жұмысы" + "\r\n";
  foreach (Tovar t in ckl.tovar)
  {
```

```
        s = s + t.Naz + " " + t.Cena.ToString() + "\r\n";
        }
        s = s + "for цикл жұмысы" + "\r\n";
        for (int i = 0; i < kol; i++)
        {
        s = s + ckl.tovar[i].Naz + " " +
ckl.tovar[i].Cena.ToString() + "\r\n";
        }
        textBox3.Text = s;
       }
       }
     }
```

Бағдарлама жұмысы 11.2-суретте көрсетілген.



11.2-сурет – Қосымшаның интерфейстерсіз жұмысы

Айта кететін жәйт, қосымшада foreach циклі массив типіндегі ckl.tovar айнымалысы үшін ғана қолданылады, айнымалының кіріктірілген интерфейсі бар.

Бірақ, егер біз foreach циклін Cklad класс объектісінің tovar массиві үшін емес, Tovar класының объекттері үшін қолданатын болсақ, мысалы

```
foreach (Tovar t in ckl)
 {
 s = s + t.Naz + " " + t.Cena.ToString() + "\r\n";
```

```
},
```

онда, компилятор қосымшада кеткен қателік туралы хабарлама шығарады:

```
«foreach statement cannot operate on variables of type '
WindowsFormsApplication1.Form1.Cklad ' because 'Books' does not
contain a public definition for 'GetEnumerator'»
```

(**foreach** операторын 'WindowsFormsApplication1.Form1.Cklad' типіндегі айнымалыларда қолдануға болмайды, өйткені осы кластың айнымалыларында **'GetEnumerator'** әдісінің айқын анықтамасы жоқ).

Бағдарламаға керекті интерфейсті қосайық, ол үшін бағдарламаға қосымша атаулар кеңістігін қосу керек:

```
using System.Collections;
```

Cklad класы IEnumerable интерфейсін мұралануы керек:

```
class Cklad : IEnumerable
```

Cklad класының денесіне GetEnumerator әдісін қосу керек:

```
public IEnumerator GetEnumerator()
 {
 for (int i = 0; i < 4; i++) yield return tovar[i];
}
```

Т.А.Павловскаяның кітабынан алынған кейбір түсініктемелерді бере кетейік [2].

«Сонымен, егер класс элементтерін бір бірден қарастыру үшін foreach циклі қолданылған болса, онда GetEnumerator, Current, MoveNext, Reset төрт әдісін орындау керек. Мысалы, егер кластың ішкі элементтері массив түрінде ұйымдастырылған болса, онда кластың жабық өрісін сипаттау керек болады, онда ағымдағы индекс сақталады. MoveNext әдісінде индекстің өзгері 1-ге тең, массив шекарасынан асып кету шарты тексеріледі. Current әдісінде массив элементі ағындағы индекс бойынша қайтарылады және т.б. »

Бұл қызықты жұмыс емес, бірақ оны жиі орындауға тура келеді, сондықтан 2.0 версияда объекте бір бірден қарастыруды жеңілдететін құралдар – итераторлар енгізілген.

Итератор дегеніміз – код блогы, онда объект элементтерін бір-бірден қарап шығу реттілігі анықталады. Foreach циклінің әрбір қадамында итератордың бір қадамы орындалады, осының нәтижесінде yield қызметтік сөзі арқылы кезекті мән қайтарылады.

2.0 версияда бір-бірден қарап шығуды орындау үшін – класта IEnumerable интерфейсі орындалатының көрсету және итераторды

сипаттау керек. Оған қол жеткізу IEnumerator интерфейсінің MoveNext және Current әдістері арқылы орындалады.

foreach циклінің әрбір қадамында итератор үшін «қоршам» - қызметтік объект әзірленеді, ол объект итератордың ағымдағы жағдайын сақтайды. Басқаша айтқанда, итераторды құраушы код үзіліссіз түрде орындалмайды, ол жеке итерацияларға бөлінген және осы аралықтардың арасында итератордың күйі сақталады.

Бағдарламада келтірілген екі өзгеріс foreach циклін қолдануға мүмкіндік береді, алдынғы қарастырған жағдайда бұл циклды қолданған кезде қате кеткені туралы хабарламаны шығаратын.

Бағдарламаның «тұрып қалуына» себеп болатын foreach циклінің жұмыс ерекшелігін ескере кету керек. Егер біздің бағдарламада тауардың бірнеше объекттерін енгізгеннен кейін, бірақ массив толық толмай тұрып, қоймадағы тауарларды қарап шығу режімін қосатын болсақ, онда бағдарлама жоқ мәндерді шығару кезінде «тұрып қалады», сондықтан foreach цикліндегі мәндерді бақылау керек.

For циклінде мұндай кемшіліктер жоқ, өйткені ондағы ақырғы мән ағымдағы ауқымды kol айнымалысының мәнімен анықталады.

```
for (int i = 0; i < kol; i++)
{
    s = s + ckl.tovar[i].Naz + " " +
ckl.tovar[i].Cena.ToString() + "\r\n";
}
```

11.4 Өзін-өзі тексеру сұрақтары

1 Интерфейс ұғымы?

2 Интерфейстік класты мұралану және қарапайым класты мұралану арасындағы айырмашылық неде?

3 Интерфейстік класс және қарапайым класс арасындағы айырмашылық неде?

4 Интерфейстік кластарда атаулар арасында қарама-қайшылықтардың туындауы неліктен?

5 Бірнеше рет мұралану кезінде атаулар арасында болатын қарама-қайшылықты қалай шешуге болады?

6 Әдістерді желімдеудің мәнісі неде?

7 Әдістердің атауын өзгертудің мәнісі неде?

8 Интерфейстердің артықшылықтары неде?

9 Интерфейстер қандай типте болуы мүмкін?

10 Интерфейсті жариялағанда қандай қызметтік сөз қолданылады?