ЕРЕКШЕ ЖАҒДАЙЛАРДЫ АЛДЫН АЛУ

14.1 Ерекше жағдайларды алдын алу туралы түсініктер

Алдынғы бөлімде көрсетілген бағдарлама операторларларында мәндерді диалог режімінде дұрыс енгізу керек, яғни сандық мәндерді символдық мәндермен, ал нақты сандарда үтірді нүктемен шатастыруға болмайды.

Ал нақты жағдайларда қате енгізілген мәндер үшін «ерекше жағдайлар» туындайды, ол бағдарламаның жұмысын уақытынан бұрын тоқтатады (бағдарлама тоқтаталады немесе «тұрып қалады»).

Күрделі жобаларды дайындау барысында қателерді болдырмау мүмкін емес, олар әрқашан көріне бермейді, бірақ нәтижелердің қате болуына немесе бағдарламаның жаңылысуына әкеледі. Осындай қателіктер «ерекше жағдайлардың» себебінен болады, мысалы: санды нөлге бөлу, жоқ файлды ашу, массив индексінің белгіленген аралықтан асып кетуі.

Ары қарай жұмысты дұрыс орындай алмайтын болғандықтан бағдарлама өз жұмысын үзетін болса, онда ондай жағдайды ерекше жағдай деп атайды.

C# тілінде ерекше жағдайларды өңдейтін арнайы механизм қарастырылған. Ерекше жағдай орын алған уақытта `Exception` класының немесе оған тиісті кластың арнайы объектісі құрылады (ол ерекше жағдайларды алдын алу деп аталады). `Exception` класы пайдаланушыға орын алған мәселе бойынша хабар береді немесе ол туралы толығырақ мәлімет береді (T типіндегі айнымалының мәнін құрады немесе мәтіндік хабарламаны шығарада).

Ерекше жағдайларды алдын алу бағдарлама жұмысындағы қателіктер туралы хабарлаудың қалыпты тәсілі болып табылады. Бағдарлама жұмысындағы орын алған қателіктер сәйкес типтегі ерекше жағдайды тудырады, осының салдарынан бағдарлама орындалуының қалыпты барысы тоқтатылады және басқару стандартты ерекше жағдай өңдеуішісіне немесе бағдарламаның өзінде қарастырылған ерекше жағдай өңдеуішісіне беріледі.

Ерекше жағдайлардың стандартты өңдеуішісі операциялық жүйеде қарастырылған , ол сәйкес хабарламаны бере отырып бағдарлама жұмысын аяқтайды. Туындаған ерекше жағдай бойынша стандартты сипаттама пайдаланушыға түсініксіз болуы мүмкін.

C# тілінде ерекше жағдайларды өңдеу құралдары бар, яғни бағдарламаның өзінде қарастырылған құралдар. Осындай өңдеу жұмыстарының мақсаты: пайдаланушыға нәтиженің дәйексіздігі туралы хабарлау, қателіктер салдарын жою әрекеті, анық нәтижеге қол жеткізу.

Ерекше жағдайды өңдеуді орындау үшін ерекше жағдай туындайтын мүмкін бөліктің орнына қорғалған блок жазылады. Қорғалған блокта ерекше жағдай туындаған кезде басқару белгілі бір ерекше жағдайды алдын алу өңдеуішісіне тапсырылады, ол пайдаланушыға бағдарламаның қате жұмысын және қателіктер салдарын жоюға әрекет етеді.

Қорғалған блокты жазу пішімі мына түрде жазылады:

```
try
{  // Қорғалған блок }
```

```
catch (T1 e1)
{// Ерекше жағдайдың өңдеуіші. Егер ерекше жағдайдың типі T1 болса іске қосылады.}
...
catch(Tk ek)
{// Ерекше жағдайдың өңдеуіші. Егер ерекше жағдайдың типі Тк болса іске қосылады. }
finally
{// Аяқталу блогы. Кез келген жағдайды іске қосылады. }
```

мұнда `try, catch, finally` — қызметтік сөздер.

Try кілттік сөзі алдында жазылған блок бағдарлама кодының қорғалған блогы немесе try-блогы деп аталады. Бағдарламашы осы блокқа бағдарлама жұмысының қате орындалуына себеп болатын, яғни ерекше жағдайды тудыратын кодты орналастырады.

Exception ұрпақтары нақты ерекше жағдайларды өңдеуге арналған. Мысалы, System атаулар кеңістігінде ArgumentOutOfRangeException, IndexOutOfRangeException, StackOverlowException және т.б. маңызды кластар анықталған.

Ерекше жағдайды не туындатады? Оны көбінесе операциялық жүйенің ядросы орындайды, бірақ C#-та оны бағдарламаның өзі throw қызметтік сөзімен орындай алады.

Сонымен, try-блогында T типіндегі ерекше жағдай орын алуы мүмкін.

C# тілінде ерекше жағдайды өңдеу үшін арнайы `catch`-блоктары қарастырылған. Әрбір `catch`-блогында `Exception` класының немесе оның ұрпағының формалды параметрі бар және ол ерекше жағдайда «дұрыс» әрекет жасауға арналған.

Бірінші, формалды параметрі T типімен келісілген (T типінде немесе оның ұрпағынан) `catch`-блогында ерекше жағдайды өзі өңдейді, сондықтан `catch`-блоктарының жазылу тәртібі өте маңызды. Алдымен арнайы өңдеушілер жазылуы тиіс. Әмбебап өңдеуіш болып `Exception` класының формалды параметрлері бар `catch`-блогы табылады және ол кез келген T типімен сәйкестендірілген. Әмбебап өңдеуіш ең соңында орналастырылады, өйткені ол кез келген типтегі ерекше жағдайды қарастыра алады. `Catch`-блогы `switch` операторы сияқты орындалады, ал одан кейін басқару қорғалған бөлікке арналған ортақ іс-әрекеттер орындалатын `finally` блогына беріледі. `Finally` блогына қорғалған бөлікте ерекше жағдай туындамаса да басқару беріледі, яғни осы блок кез келген жағдайда орындалады. Онда, мысалы, ашық файлдар жабылуы немесе қорғалған бөлікке бөлінген ресурстардың босатылуы мүмкін.

Catch() өңдеушісінің кез келген санын жазуға рұқсат берілген, сонымен қатар finally блогын қолданбауға болады. Осы кластың ең маңызды қасиеттері 14.1-кестеде көрсетілген.

14.1-кесте - Exception класының қасиеттері

Қасиеттер	Сипаты

HilpLink	Қатені сипаттайтын анықтамалық URI файлын қайтарады
Message	Қатенің мәтіндік сипаттамасын қайтарады
Source	Ерекше жағдайды тудыратын объектті немесе қосымшаның атауын қайтарады
StackTrace	Ерекше жағдай туындаған уақытта стек күйін қайтарады
InnerExcepton	Ағымдағы ерекше жағдай бойынша мәліметтерді сақтау үшін қолданады

14.2 Ерекше жағдайларды алдын алу бойынша мысал

5 кездейсоқ бүтін саннан тұратын массивті құру керек, сандар 0-ден 100 дейінгі аралықта болады. Массив индекстері 0-ден 8-ге дейінгі аралықта кездейсоқ құрылады. Егер массив индексі берілген аралықтан асып кетсе, онда ерекше жағдайды өңдеуді қарастыру керек.

Сонымен, индекстің рұқсат етілген мәні 0-ден бастап 4-ке дейінгі аралықта болады. Индекстің басқа мәндерінде бағдарлама жұмысын тоқтатуға әкелетін ерекше жағдай туындайды. Бағдарламаның қорғалған блогында индекс құрылады және ол рұқсат етілген аралықтан асып кетсе, онда сәйкес хабарлама шығады, бірақ бағдарлама жұмысы тоқтатылмайды.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main(string[] args)
 {
  int i, j;
  int[] a = new int[5];
  Random rnd = new Random();
  Console.WriteLine("Kezdeisok bytin sandar massivi: ");
  i = 0;
  while (i < 5)
  {
  j = 0;
  try
  {
   i++;
   j = rnd.Next(8);
   a[j] = rnd.Next(100);
   Console.WriteLine("a[{0}] = {1} ", j, a[j]);
  }
  catch (IndexOutOfRangeException)
```

```
   {
    i--;
    Console.WriteLine("Massivtin indeksi diapozon sheginen
tiskari {0}", j);
   }
   }
   Console.WriteLine("Bagdarlamanin zhymisinin soni");
   Console.ReadLine();
  }
  }
}
```

Бағдарлама жұмысы:

Kezdeisok bytin sandar massivi:

a[1] = 31

Massivtin indeksi diapozon sheginen tiskari 6

a[2] = 36

a[2] = 38

Massivtin indeksi diapozon sheginen tiskari 7

Massivtin indeksi diapozon sheginen tiskari 6

a[3] = 68

a[1] = 63

Bagdarlamanin zhymisinin soni

Қарастырылған мысалда тек бір ғана ерекше жағдай қарастырылған - массив индексінің рұқсат етілген аралықтан асып кетуі. Берілген мысалда бұл бір ғана мүмкін ерекше жағдай, сондықтан бір ғана `catch` блогы қолданылған.

14.3 Өзін-өзі тексеру сұрақтары

1 Ерекше жағдай ұғымы.

2 Ерекше жағдайды алдын алу ұғымы.

3 Бағдарламада ерекше жағдайларды өңдеу механизмі (әдісі) неге негізделген?

4 Операциялық жүйеде қарастырылған ерекше жағдай бойынша стандартты өңдеуіш қалай жауап береді?

5 Бағдарламаның өзінде алдын ала қарастырылған ерекше жағдай бойынша өңдеуіш не үшін қолданылады?

6 Бағдарламаны қорғалған блогы не үшін құрылады?

7 Бағдарламаның қорғалған блогы қандай қызметтік сөзден басталады?

8 Бағдарламада қарастырылған ерекше жағдай бойынша өңдеуіш ерекше жағдайға қалай жауап береді?

9 `Catch`-блоктар не үшін қолданылады?

10 Бір қорғалған блокқа қанша `catch`-блок сәйкес бола алады?