Дәріс 5. Оқиғалар негізінде синхронизациялау. Interlocked класы.

Дәрістің мақсаты: Студенттерде көпағындық программаларда оқиғалар негізінде синхронизациялауды түсіну.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Синхронизациялауға байланысты оқиға түрлерін ажырату;
- Interlocked класының қызметін түсіну.

Оқиғаларды пайдалану

C# бағдарламасында үндестіру үшін нысанның тағы бір түрі қарастырылған: оқиға. Бастапқы күйге қолмен және автоматты түрде орнатылатын оқиғалардың екі түрі бар. Олар тиісінше ManualResetEvent және AutoResetEvent сыныптарында ұсталады. Бұл сыныптар сыныптар иерархиясының жоғарғы деңгейіндегі EventWaitHandle класының туындылары болып табылады және бір ағын басқа ағымда кейбір оқиғаның пайда болуын күтетін жағдайларда қолданылады. Мұндай оқиға пайда болған соң, екінші ағын ол туралы бірінші ағынды хабардар етеді, сөйтіп оны орындауды қайта бастауға мүмкіндік береді.

Төменде ManualResetEvent және AutoResetEvent сыныптарының құрастырушылары келтірілген.

public ManualResetEvent(bool initialState)

public AutoResetEvent(bool initialState)

Егер екі пішінде де initialState параметрінің логикалық мәні true болса, онда оқиға туралы бастапқыда хабарланады. Ал егер оның логикалық мәні false болса, онда оқиға туралы бастапқыда хабарланбайды. Оқиғалар өте қарапайым қолданылады. Мысалы, түріндегі оқиға үшін қолдану тәртібі мыналар ManualResetEvent. Кейбір оқиғаны күтетін ағын осы оқиғаны көрсететін оқиға нысаны үшін WaitOne() әдісін тудырады. Егер оқиға объектісі сигналдық күйде болса, онда дереу әдісінен қайтару. Олай болмаған жағдайда шақырушы ағынды орындау оқиға туралы хабарлама алынғанға дейін тоқтатыла тұрады. Оқиға басқа ағымда болғаннан кейін, бұл ағын Set() әдісін шақырып, оқиға нысанын дабыл күйіне орнатады. Сондықтан Set() әдісін оқиғаның болғанын хабарлаушы ретінде қарағаған жөн. Оқиға объектісі сигналдық күйге орнатылғаннан кейін WaitOne() әдісінен дереу қайтарылады және бірінші ағын өзінің орындалуын қайта бастайды. Ал Reset() әдісін шақыру нәтижесінде оқиға нысаны дыбыстық емес күйге қайтарылады.

AutoResetEvent түріндегі оқиға түрінен ерекшеленеді ManualResetEvent тек бастапқы күйге орнату тәсілімен. Егер түрдегі оқиға үшін Егер оқиға нысаны Reset() әдісі шақырылғанша дабылды күйде қалса, онда оқиға түрі үшін оқиға оқиғаны күтіп тұрған ағын ол туралы хабар алып, оны орындауды қайта бастағаннан кейін автоматты түрде дыбыстық емес күйге ауысады. Сондықтан, егер AutoResetEvent түріндегі оқиға қолданылса, онда әдісті шақыру Reset() міндетті емес.

Төменде келтірілген бағдарлама мысалында ManualResetEvent түрі оқиғасының қолданылуы көрсетіледі.

```
using System;
using System.Threading;
// Келесі ағын оның конструкторына оқиға берілгенін хабарлайды.
class MyThread {
public Thread Thrd;
ManualResetEvent mre;
```

```
public MyThread(string name, ManualResetEvent evt) {
Thrd = new Thread(this.Run);
Thrd.Name = name;
mre = evt;
Thrd.Start();
}
// Ағынға кіру нүктесі.
void Run() {
Console.WriteLine(Thrd.Name + " ағынында");
for(int i=0; i<5; i++) {
Console.WriteLine(Thrd.Name);
Thread.Sleep(500);
}
Console.WriteLine(Thrd.Name + " аяқталды!");
// Оқиға жөнінде хабарлау.
mre.Set();
}
}
class ManualEventDemo {
static void Main() {
ManualResetEvent evtObj = new ManualResetEvent(false);
MyThread mt1 = new MyThread("1-ші оқиғалық ағын", evtObj);
Console.WriteLine("Негізгі ағын оқиғаны күтуде.");
// Оқиға туралы хабарламаны күту.
evtObj.WaitOne();
Console.WriteLine("Негізгі ағын бірінші ағыннан оқиға туралы хабарлама алды.");
// Оқиғалық объектіні бастапқы күйге орнату.
evtObj.Reset();
mt1 = new MyThread("2-ші оқиғалық ағын", evtObj);
// Оқиға туралы хабарламаны күту.
evtObj.WaitOne();
Console.WriteLine("Негізгі ағын екінші ағыннан оқиға туралы хабарлама алды.");
}
}
```

Interlocked класы

Синхрондаумен байланысты тағы бір сынып Interlocked класы болып табылады. Бұл сынып тек жалпы айнымалының мәнін өзгерту қажет болғанда басқа үндестіру құралдарына балама ретінде қызмет етеді. Interlocked класында қол жетімді әдістер олардың әрекеті біртұтас, үздіксіз операция ретінде орындалатынына кепілдік береді. Бұл бұл жағдайда ешқандай синхрондау мүлдем қажет емес дегенді білдіреді. Interlocked класында екі бүтін мәнді қосу, бүтін мәнді инкременттеу және декременттеу, объектінің мәндерін салыстыру және орнату, объектілермен алмасу және 64-разрядты мәнді алу үшін статикалық әдістер ұсынылады. Бұл операциялардың барлығы үзіліссіз орындалады.

Төменде келтірілген бағдарламаның мысалында Interlocked класының Increment() және Decrement() сыныптарынан екі әдісті қолдану көрсетіледі. Бұл ретте екі әдістің де мынадай нысандары пайдаланылады:

public static int Increment(ref int location)

public static int Decrement(ref int location)

мұнда location - бұл инкременттеуге немесе декременттеуге жататын айнымалы.

```
using System;
using System.Threading;
// Ортақ ресурс.
class SharedRes {
public static int Count = 0;
}
// Бұл ағында SharedRes.Count айнымалысы инкременттеледі.
class IncThread {
public Thread Thrd;
public IncThread(string name) {
Thrd = new Thread(this.Run);
Thrd.Name = name;
Thrd.Start();
}
// Ағынға кіру нүктесі.
void Run() {
for(int i=0; i<5; i++) {
Interlocked.Increment(ref SharedRes.Count);
Console.WriteLine(Thrd.Name + " Count = " + SharedRes.Count);
}
}
}
// Бұл ағында SharedRes.Count айнымалысы декременттеледі.
class DecThread {
public Thread Thrd;
public DecThread(string name) {
Thrd = new Thread(this.Run);
Thrd.Name = name;
Thrd.Start();
}
// Ағынға кіру нүктесі.
void Run() {
for(int i=0; i<5; i++) {
Interlocked.Decrement(ref SharedRes.Count);
Console.WriteLine(Thrd.Name + " Count = " + SharedRes.Count);
}
}
}
class InterlockedDemo {
static void Main() {
// Екі ағынды құру.
IncThread mt1 = new IncThread("Инкременттеуші Ағын");
DecThread mt2 = new DecThread("Декременттеуші Ағын");
mt1.Thrd.Join();
mt2.Thrd.Join();
}
}
```