КӨПАҒЫНДЫҚ ПРОГРАММАЛАУ НЕГІЗДЕРІ. АҒЫНДЫ ҚҰРУ ЖӘНЕ ІСКЕ ҚОСУ. JOIN, SLEEP ӘДІСТЕРІ.

«ПРОГРАММАЛАУ ТЕХНОЛОГИЯЛАРЫ» ПӘНІ

ЛЕКТОР: АБДРАХМАНОВА М.Б.

МАҚСАТЫ ЖӘНЕ МІНДЕТТЕРІ

Дәрістің мақсаты: Студенттерде көпағындық программалау негіздерін түсіну дағдыларын қалыптастыру.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Көпағындық программалаудың қызметін түсіну;
- Ағынды құруға және іске қосуға арналған командаларды білетінін көрсету;
- Join, Sleep әдістерінің синтаксисі мен қызметін түсіну.

КӨПАҒЫНДЫҚ ПРОГРАММАЛАУ

- Көпағынды программа қатар орындалатын екі және одан да көп бөліктен тұрады. Мұндай программаның әрбір бөлігі ағын деп аталады командаларды орындаудың жеке жолын анықтайды.
- Көпағынды программалау C# тілінің өзінде қарастырылған бірқатар құралдарға, сондай-ақ .NET Framework ортасында анықталған кластарға сүйенеді.
- .NET Framework ортасының 4.0 нұсқасында көпағындық қосымшаларға қатысты 2 толықтыру енгізілді: TPL (Task Parallel Library - Тапсырмаларды параллельдеу кітапханасы) және PLINQ (Parallel LINQ - Интеграцияланған сұраулардың параллель тілі).

КӨПАҒЫНДЫҚ ПРОГРАММАЛАУ

TPL артықшылықтарына қарамастан, бастапқы көпағындық жүйені қолданудың себептері:

- Көпағынды өңдеудің алғашқы тәсілін пайдаланып жазылған біраз код мөлшері қолданыста бар.
- TPL негізіндегі дайындалған код құрамында алғашқы көпағындық жүйенің элементтері қолданылуы мүмкін, мысалы, синхронизациялау құралдары.
- TPL кітапханасы тапсырма абстракциясына сүйенетін болғанына қарамастан, оның түп негізінде ағындық құралдар жатыр.

КӨПАҒЫНДЫҚ ӨҢДЕУ

- Көптапсырмалықтың 2 түрі ажыратылады: процестер негізінде және ағындар негізінде.
- Процесс іс жүзінде атқарылатын программаны білдіреді. Сондықтан процестер негізіндегі көптапсырмалық – компьютерде екі немесе одан артық программаны параллель орындауға мүмкіндік беретін құрал. Бұл жағдайда тапсырмалар жоспарлаушы құралдың басқара алатын ең кіші код бірлігі программа болады.
- Ағын атқарылатын кодтың басқарылатын бірлігін білдіреді. Ағындар негізіндегі көптапсырмалықты ұйымдастыру жағдайында әрбір процестің кемінде бір ағыны болуы тиіс, ағындар одан артық болуы да мүмкін. Яғни, бір программада бір уақытта екі немесе одан артық тапсырма орындалуы мүмкін.

КӨПАҒЫНДЫҚ ӨҢДЕУ

- Көпағынды өңдеудің басты артықшылығы: программа жұмысы барысындағы бос тұру уақытын тиімді пайдалану.

Ағын күйлері:

- атқарылып жатқан ағым;
- орындалуға әзір – орталық процессор уақыты мен ресурстарын күтуде;
- тоқтатылған – уақытша атқарылмайтын;
- қайта жалғастырылған;
- бұғатталған – өзінің орындалуы үшін ресурстар күтуде;
- аяқталған – жұмысының орындалуы аяқталды және қайта жалғастырылуы мүмкін емес.

КӨПАҒЫНДЫҚ ӨҢДЕУ

- .NET Framework ортасында ағынның екі түрі анықталған: басымдылықты (келісім бойынша) және фондық. Фондық ағынның процесіндегі барлық басымдылықты ағындар тоқтатылған болса, фондық ағын да автоматты түрде аяқталады.
- Барлық процестердің құрамында кемінде бір ағын болады, ол негізгі деп аталады. Программаның орындалуы осы ағыннан басталады. Негізгі ағыннан басқа ағындарды құруға болады.
- Көпағындық программалауды қолдайтын кластар System.Threading атаулар кеңістігінде анықталған.
- `using System.Threading;`

THREAD КЛАСЫ

- Көпағынды өңдеу жүйесі орындау Thread класына негізделеді, бұл класс атқару ағынын инкапсулациялайды. Бұл класты мұралау мүмкіндігі жоқ. Класс құрамында ағындарды басқаруға арналған әдістер мен қасиеттер анықталған.
- Ағынды құру үшін Thread типті объектінің экземплярын алу жеткілікті. Thread класы конструкторының формасы:
- `public Thread (ThreadStart іске_қосу)`
- мұнда *іске_қосу* - ағынды атқаруды бастау мақсатында шақырылатын әдістің аты, ал ThreadStart - .NET Framework ортасында анықталған делегат.
- `public delegate void ThreadStart()`

THREAD КЛАСЫ (МЫСАЛ 1)

- Жаңадан құрылған ағынның атқарылуы оның Start() әдісі шақырылғанға дейін басталмайды. Start() әдісі Thread класында анықталған. Бұл әдісті жариялаудың 2 формасы бар.
- *public void Start()*
- Атқарылып бастаған ағын *іске_қосу* арқылы көрсетілген әдістен қайту орындалғанша жалғасады. Атқарылуы басталып қойған ағынға Start() әдісін шақыруға тырыссаңыз, ThreadStateException аластамасы туындайды.

SLEEP() ӘДІСІ

- Sleep() статикалық әдісі қай ағын құрамынан шақырылса, сол ағынның жұмысын миллисекундпен көрсетілген уақыт мөлшеріне тоқтатады. Бір ағын жұмысы тоқтатылған кезде басқа ағын жұмысын жалғастыра алады.
- **`public static void Sleep(int `*`миллисекундпен_бөгелу`*`)`**
- Егер бөгелу уақыты 0 миллисекунд болса, онда шақырушы ағын өз кезегін күтіп тұрған басқа ағынның жұмыс атқаруына мүмкіндік беру үшін ғана бөгеледі.
- Көпағындық программалауда программаның жұмысын негізгі ағын аяқтау ережесі қабылданған. Бұл ережені ұстану міндетті болмағанымен, осы арқылы программаның ақырғы нүктесін айқын белгілеуге болады.

КӨПАҒЫНДЫҚ ПРОГРАММАНЫ ЖЕТІЛДІРУ (МЫСАЛ 2)

- Ағын құрылғаннан кейін бірден атқарылуын бастау: жұмыс істеп отырған класс конструкторында Thread типті объект экземплярын алу;
- Ағын атын бейнелеу үшін Thread класының Name қасиетін пайдалану:
- `public string Name { get; set; }`

БІРНЕШЕ АҒЫНДЫ ҚҰРУ

- Бірнеше ағынды құру мысалын қарастыру: мысал 3

АҒЫННЫҢ АЯҚТАЛУЫ (МЫСАЛ 4)

- Ағынның аяқталуын бақылау үшін келесі қасиетті пайдалануға болады:
- ***public bool IsAlive { get; }*** // *t*rue – ағын әлі де орындалуда.

АҒЫННЫҢ АЯҚТАЛУЫ (МЫСАЛ 5)

- Ағынның аяқталуын бақылаудың тағы бір жолы – Join() әдісін пайдалану.
- `public void Join()`
- Шақырушы ағын шақырылған әдіске қосылғанша күтеді. Егер ағын басталмаған болса, онда ThreadStateException аластамасы генерацияланады. Join() құрамында ағынның аяқталуын күтудің максимал уақытын көрсетуге болады.

Назарларыңызға рахмет!