1 дәріс. Объектіге бағытталған программалау негіздері. Кластар, объектілер және әдістерге кіріспе. Параметрленген әдістер.

Мақсаты: студенттерде объектіге бағытталған программалаудың негізгі механизмдері және кластарды құру негіздері бойынша түсініктерін көрсетуге қабілет қалыптастыру.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Инкапсуляция, полиморфизм, мұралау идеялары бойынша түсініктерін көрсету;
- Өрістері мен әдістері бар класты жазу форматын білетінін көрсету;
- Параметрленген әдістерді пайдалану ерекшеліктері бойынша түсініктерін көрсету.

Объектіге бағытталған программалау

C# тілінде тілінің негізгі ұғымы болып *объектіге бағытталған программалау* (ОБП) түсінігі саналады. ОБП әдістемесін C# тілі тілінен бөліп алуға болмайды, сондықтан мұндағы барлық программамалар, ең төменгі дәрежеде дегеннің өзінде де, объектіге бағытталған болып табылады. Осыған орай, ең қарапайым программа жазу үшін де, ОБП-ның негізгі принциптерін қолдану өте маңызды, әрі пайдалы жұмыс болып саналады.

ОБП өзіне құрылымдық программалаудың ең жақсы идеяларын алып, оны жаңа түсініктермен толықтырды. Осының нәтижесінде программаларды ұйымдастырудың жаңа және үздік тәсілі жасалды. Ең жалпы түрде, программа екі тәсілдің біреуімен ұйымдастырылуы мүмкін: *кодтар айналасында* (яғни, бұл іс жүзінде орындалып жатқан әрекеттер маңында) немесе *мәліметтер айналасында* (яғни кодтардың әсеріне ұшырайтын деректер маңында) атқарылатын істерді жүзеге асыру арқылы орындау. Құрылымдық программалау әдістері арқылы жасалған программалар, әдетте, кодтар айналасында ұйымдастырылады. Мұндай тәсілді "мәліметтерге әсер ететін код" деп қарауға болады.

Объектіге бағытталған программалар тіпті басқаша жұмыс істейді. Олар берілген деректерді негізге ала отырып, "мәліметтер кодқа (ішкі программаға) қол жеткізуді басқарады" деген басты қағиданы ұстанып, мәліметтер айналасында ұйымдастырылған. Программалаудың объектіге бағытталған тілінде *мәліметтер* (деректер) және соларды өңдеуге, яғни өзгертуге, түрлендіруге рұқсат етілген *кодтар* анықталады. Демек, *мәліметтер типі* солармен *орындалатын операцияларды* дәлме-дәл *анықтай алады*.

ОБП-ның принциптерін сүйемелдеу үшін, барлық объектіге бағытталған программалау тілдерінің (соның ішінде C# тілі тілі де бар) үш түрлі жалпы қасиеттері болуы тиіс, олар: *инкапсуляция*, *полиморфизмом* және *мұралау* қасиеттері. Енді осы қасиеттерді әрқайсысын жеке-жеке қарастырып шығайық.

Инкапсуляция

Инкапсуляция – бұл өңделуге тиіс код пен оған берілетін мәліметтерді сырттың араласуынсыз біріктіретін механизм, ол, сонымен қатар, деректердің дұрыс пайдаланылуын да қадағалап отырады. Объектіге бағытталған тілде берілген мәлімет пен кодтың автономды *қара жәшік* түрінде біріктіріліп тұруы да мүмкін. Осындай жәшіктің ішінде барлық қажетті мәліметтер мен оның өңдейтін код бірге орналасады. Код пен берілген мәліметтер осылайша бір-бірімен байланысқаннан кейін барып, объект құрылады. Басқаша айтқанда, *объект* дегеніміз – инкапсуляцияны сүйемелдейтін элемент.

Объектіде код пен мәліметтер немесе солардың әрқайсысы жеке-жеке *жабық* немесе *ашық* түрде орналасуы мүмкін. Жабық мәліметтер немесе код сол объектінің ішіндегі соның қалған бөлігіне ғана белгілі, әрі қол жетімді болады. Бұл жабық мәліметтердің немесе кодтың сол объектіден тысқары жерде қолданылуына жол бермейді, яғни оларға сырттан қол жеткізуге болмайды. Егерде мәліметтер немесе оның коды ашық болатын болса, онда олар объект ішінде анықталғанмен, программаның басқа бөліктерінде де пайдаланыла

береді, яғни оған қол жеткізуге болады. Көбінесе, объектінің ашық бөліктері оның жабық бөліктерімен басқарылатын интерфейс ұйымдастыру үшін қызмет етеді.

C# тілінде тіліндегі инкапсуляцияның негізгі бірлігі объект формасын анықтайтын *класс* болып табылады. Ол мәліметтерді және де солармен жұмыс істейтін кодты сипаттайды. C# тілінде тілі ортасындағы *класс сипаттамасы* класс экземплярлары болып табылатын объектілерді құру үшін қолданылады. Демек, класс дегеніміз, негізінде, объектіні құрудың бірсыпыра схемалық сипаттауларын беру тәсілі болып табылады.

Полиморфизм

Полиморфизм, бұл грекше "формалар жиыны" дегенді білдіреді, бұл – бір интерфейстің ұқсас іс-әрекеттердің бірнеше тобын (класын) біріктіріп, біртіндеп басқаруына мүмкіндік беретін қасиет.

Көбінесе полиморфизм ұғымының жалпы мағынасы "бір интерфейс – көптеген әдістер" түрінде айтылады. Бұл өзара байланысқан іс-әрекеттер тобы үшін жалпы бір интерфейс жасауға болады дегенді білдіреді. Полиморфизм *әрекеттердің жалпы бір класын сипаттау* үшін бір ғана интерфейс жасау арқылы программаны қарапайым түрде қысқартып жазу мүмкіндігін береді. Әрбір жеке жағдайда, белгілі бір нақты әрекетті (яғни әдісті) таңдап алу бұл – компилятордың міндеті. Мұны программалаушының өзінің істеуі қажет емес. Оған жалпы интерфейсті есте сақтай отырып, оны дұрыс пайдалана білу жеткілікті болып табылады.

Мұралау

Мұрагерлік дегеніміз жұмыс барысында бір объектінің басқа бір объектінің қасиеттерін алу (иелену) процесі болып табылады. Бұл өте маңызды процесс, өйткені ол иерархиялық жіктеу принципін қамтамасыз етеді. Егер тереңірек ойланатын болсақ, онда білімдеріміздің басым бөлігі иерархиялық жіктеу бойынша жалқыдан жалпыға қарай топтарға бөлініп жүйеленеді екен.

Егер объектілердің иерархиялық (сатылы түрдегі) ортақ қасиеттерін есепке алмасақ, онда әрбір объект үшін оның барлық қасиеттерін анықтауға тура келер еді. Ал егер мұралауды пайдалансақ, онда бұл объектінің осы кластағы басқа объектілерден ерекше болып келген қасиеттерін ғана көрсету жеткілікті болады. Ол сондай-ақ, өз класының барлық объектілеріне де ортақ жалпы қасиеттерін де (тегінің қасиеттерін) мұралай алады. Демек, мұрагерлік қасиеттерді есепке алу арқасында бір объект жалпы кластың белгілерін алған жеке бір экземплярына (данасына) айналады.

Кластар мен әдістер

Класс объектінің формасын анықтайтын үлгі (шаблон) болып табылады, оны программада қолданылатын типтердің күрделі түрі деп те атауға болады. Онда керекті *мәліметтер* және соларды өңдейтін, яғни *түрлендіретін код* екеуі қатар көрсетіледі. Егер ең қарапайым кластар тек қана кодтан немесе мәліметтерден тұратын болса, нағыз толыққанды кластарда бұл екеуі де орын алады. C# тілінде объектілерді құруға арналған класс спецификациясы қолданылады, оларды класс даналары (экземплярлары) деп атайды. Сондықтан класты, негізінде, объект құруға арналған тізбектелген амалдар сипаттамасы деп қарауға болады. Класс оның объектісі құрылғаннан кейін ғана барып, компьютер жадынан белгілі бір орын алатын болады.

Мәліметтер (айнымалылар) класта анықталған *мәлімет-мүшелерде*, ал код (программа) – *функция-мүшелерде* орналасады. *Мәлімет-мүшелерге* (немесе өрістерге) жеке *даналар айнымалылары* және *статикалық айнымалылар* жатса, ал *функция-мүшелерге – әдістер* (функциялар), *конструкторлар*, *деструкторлар*, *индексаторлар*, *оқиғалар*, *операторлар* және *қасиеттер* жатады.

Әдістер класта анықталған мәліметтерді өңдеуге арналған ішкі программалар болып табылады, ал көптеген жағдайларда олар сол мәліметтерге қол жеткізіп, есептеулерде пайдалануға мүмкіндік береді. Көбінесе, программаның басқа бөліктері класпен оның осы әдістері арқылы әрекеттеседі.

Әдіс бір немесе бірнеше операторлардан тұрады. C# тіліндегі сауатты жазылған кодта әрбір әдіс тек бір ғана функцияны орындайды. Әрбір әдістің оны шақыруды жүзеге асыратын арнайы аты болады. Жалпы, әдіске кез келген бір идентификаторды ат ретінде беруге болады. Дегенмен, **Main()** идентификаторы программаны бастайтын әдіске берілетін ат ретінде бекітілген. Кез келген әдістер аты ретінде C# тілінде тілінің түйінді сөздерін қолдануға да болмайды.

Төменде әдісті анықтаудың жалпы формасы келтірілген:

```
қатынасу_түрі қайтарылатын_тип аты(параметрлер_тізімі)  {
    // әдіс тұлғасы
}
```

мұндағы *қатынасу_түрі* – бұл қатынасу модификаторы, ол осы әдісті программаның қандай бөліктерінен шақыруға болатынын анықтайды. Келесі *қайтарылатын_тип* әдістің қайтаратын мәліметінің типін анықтайды. Бұл тип құрылатын кластың типін де анықтайды. Егер әдіс мән қайтармайтын болса, онда қайтарылатын типті **void** деп көрсету керек. Келесі *аты* әдіске берілетін нақты атауды көрсетеді. Аты ретінде жариялаудың осы аймағында қолдануға болатын кез келген сөзді таңдаймыз. Ең соңғы *параметрлер_тізімі* – бұл үтірмен бөліне отырып, типі мен идентификаторы қатар көрсетілген, екі сөзден тұратын анықтаулар тізбегі. Параметрлер әдісті шақыру кезінде оған берілетін *аргументтер* мәндеріне сәйкес айнымалылар тізбегі. Егер әдістің параметрлері болмаса, онда тізім бос болады.

Класс **class** деген түйінді сөз арқылы жасалады. Төменде бірнеше экземплярлар айнымалылары мен әдістерден тұратын ең қарапайым кластың жалпы құрастырылу бейнесі келтірілген.

```
class класс_аты {
   // Экземпляр айнымалыларын жариялау
   қатынас_түрі типі 1-айнымалы;
   қатынас_түрі типі 2-айнымалы;
   //...
   қатынас_түрі типі N-айнымалы;

   // Әдістерді жариялау
   қатынас_түрі қайтару_типі 1-әдіс (параметрлері) {
      // әдіс операторлары
   }
   қатынас_түрі қайтару_типі 2-әдіс (параметрлері) {
      // әдіс операторлары
   }
   //...
   қатынас_түрі қайтару_типі N-әдіс (параметрлері) {
      // әдіс операторлары
   }
}
```

Әрбір айнымалыны немесе әдісті жариялау алдында *қатынас_түрі* көрсетілетініне назар аударыңдар. Бұл қатынас спецификаторы, мысалы, **public** сөзі, ол осы класс мәлімет-мүшесіне қатынасу, яғни қол жеткізу жолын анықтайды. Класс мүшелері өз аймағында жабық (**private**) немесе жалпы ашық (**public**) болуы мүмкін. Спецификаторды көрсету міндетті емес, егер ол жоқ болса, онда жарияланатын мүше класс аймағында жабық болып саналады. Қатынасу спецификаторы келесі тарауларда толығырақ қарастырылады.

Кластарды нақты мысалдар арқылы көрсету үшін кітап туралы ақпараттарды біртіндеп жасыратын (инкапсуляциялайтын) класс құрайық. Бұл класта кітап туралы ақпараттың бес

элементі: *кітаптың аты, кітаптың авторы, жарыққа шыққан жылы, баспаның аты* және *беттер саны* сақталатын болады.

Бір кітаптың баспа табақтарының саны класс құрамында орналасқан беттер саны өрісінің мәндеріне байланысты болады, сол себепті мұндағы есептеуді кластың өзінде орындаған дұрыс. Оған қоса, бұл есептеуді класқа енгізе отырып, біз бұл класты пайдаланатын басқа программаларды осы есептеуді өздерінің жүргізуінен босатамыз. Осыған орай кодты қайталау да орын алмайды. Және де класқа баспа табақтарының санын есептеуді қосу объектіге бағытталған құрылымды жақсартуға ықпал етеді, өйткені кітапқа тікелей байланысты шамалар класс құрамында орналасқан (басқаларынан жасырылған – инкапсуляцияланған).

```
class Kitap {
   // экземпляр айнымалылары
   public string aty;      // кітаптың аты
   public string avtor;    // автордың аты-жөні
   public int zhyly;       // кітаптың шыққан жылы
   public string baspa;    // баспаның аты
   public int bet_sany;    // кітаптың беттер саны

   // класс әдістері
   public int baspa_tabagy() { return (int)(bet_sany/16); }
   //кітаптың  толық баспа табақтарының санын анықтау функциясы

   public void Shygaru()
   { Console.WriteLine(aty + " кітабының авторы – " + avtor + ",
ол " + baspa + "баспасынан " + zhyly + " жылы жарық көрді, беттер
саны: " + bet_sany); }
}
```

Экземпляр айнымалысы қатынасу спецификаторынан бөлек, жай жергілікті айнымалы тәрізді жарияланады. **Class** сөзі мәліметтердің жаңа типі құрылғанын білдіреді.

Класс әдістерін қарастырайық.

```
public void Shygaru() { }
```

Бұл жолда **Shygaru** атты параметрлері жоқ әдіс жарияланған. Ол үшін **public** типі көрсетілген, яғни оны программаның кез келген бөлігінен шақырып пайдалануға болады. **Shygaru()** әдісі **void** типіндегі бос мән қайтарады, яғни ол оны шақырған программа бөлігіне ешқандай да мән қайтармайды. Қарастырылып отырған жол жүйелі жақшамен аяқталады, ол әдістің ішкі операторларын (тұлғасын) орналастыруға жол ашады.

Shygaru() әдісінің ішкі тұлғасы тек бір оператордан ғана тұрады.

```
Console.WriteLine(aty + " кітабының авторы – " + avtor + ", ол "
+ baspa + "баспасынан " + zhyly + " жылы жарық көрді, беттер саны:
" + bet_sany);
```

Бұл оператор кітап туралы ақпаратты экранға шығарады. Ал **Kitap** типіндегі әрбір объектіде **aty**, **avtor**, **baspa**, **zhyly**, **bet_sany** айнымалыларының өз көшірмелері болғандықтан, **Shygaru()** әдісін шақырғанда, есептеулерді орындау кезінде осы айнымалылардың шақырған объектідегі көшірмелері пайдаланылады.

Shygaru() әдісі жабылатын жүйелі жақшамен аяқталады. Бұл жақша кездескенде, басқару кері қарай программаны шақырған бөлікке беріледі.

Kitap типіндегі белгілі бір объект жасау үшін, келесі операторды пайдаланамыз.

```
Kitap kitap1 = new Kitap();   //  Kitap типіндегі объект жасау
```

Осы операторды орындағаннан кейін жаңа **kitap1** объектісі **Kitap** класының бір данасы (экземпляры) болып табылады, яғни ол нақты "физикалық" тұрғыда пайда болады. **new** операторы объект үшін динамикалық түрде (яғни орындалу кезінде) компьютер жадынан орын бөледі де, оған сілтеме қайтарады, сонан соң сол сілтеме айнымалыда сақталады.

kitap1 айнымалысын жариялауды ол сілтеме жасап тұратын объектіні құрудан бөлек жасауға болады, ол былайша орындалады.

```
Kitap kitap1;       // объектіге сілтеме жасауды жариялау
kitap1 = new Kitap();    // Kitap типіндегі объект үшін жадыдан
                         // орын бөлу
```

Бірінші жолда **Kitap** типіндегі объектіге сілтеме ретінде **kitap1** айнымалысы жарияланады. Мұндағы **kitap1** – объектіге сілтеме жасай алатын айнымалы, бірақ оның өзі объект болып табылмайды. Ал екінші жолда **Kitap** типіндегі жаңа объект құрылады да, сол объектіге сілтеме **kitap1** айнымалысына меншіктеледі.

Кластардың объектілеріне сілтеме арқылы қол жеткізуге болатын мүмкіндікке байланысты кластар *сілтемелік тип* болып аталып кетті. **Kitap1** айнымалысы объектінің өзін емес, тек оған сілтемені ғана сақтайды.

Book типіндегі әрбір объект **aty**, **avtor**, **zhyly**, **baspa**, **bet_sany** экземплярлар айнымалыларына сәйкес, өз көшірмелерін құрып сақтайтын болады. Осы айнымалыларды пайдалану үшін класс мүшесімен қатынас құратын, яғни оны қолданатын оператор керек болады, ол өз құрылымына сәйкес нүкте-оператор деп аталады. Нүкте-оператор объект атын класс мүшесінің атымен байланыстыру үшін қажет. Төменде нүкте-оператордың жалпы жазылу формасы көрсетілген.

```
объект.мүше
```

Бұл формада объект сол жақта, ал мүше – оң жақта тұрады. Мысалы, **kitap1** объектісінің **avtor** айнымалысына **"Muhtar Auezov"** мәнін меншіктеу келесідей оператор арқылы орындалады.

```
kitap1.avtor = "Muhtar Auezov";
```

Жалпы түрде, нүкте-оператор экземплярлар айнымалылары мен әдістерді (функцияларды) пайдалану үшін керек болады.

Мысал 1. **Kitap** класының объектілерін пайдаланатын программаның толық мысалы.

```
// Kitap класы пайдаланылатын программа
using System;
class Kitap {
   public string aty;      // кітаптың аты
   public string avtor;    // автордың аты-жөні
   public int zhyly;       // кітаптың шыққан жылы
   public string baspa;    // баспаның аты
   public int bet_sany;    // кітаптың беттер саны

   // класс әдістері
   public int baspa_tabagy() { return (int)(bet_sany/16); }
   // кітаптың толық баспа табақтарының санын анықтау функциясы
   public void Shygaru() { Console.WriteLine("\"" + aty +"\"" + "
kitabinin avtory - " +avtor + ", ol \"" +baspa + "\" baspasynan "
+zhyly + " zhyly zharyk kordi, better sany: " +bet_sany + ".");
}
// Бұл класта Kitap типіндегі объект жарияланады
class Program {
```

```
    static void Main() {
        Kitap kitap1 = new Kitap(); //Kitap типіндегі объект құру
        kitap1.aty = "C# 4.0 The Complete Reference";
        kitap1.avtor = "Herbert Shildt";
        kitap1.zhyly = 2010;
        kitap1.baspa = "McGraw-Hill Education";
        kitap1.bet_sany = 976;
        kitap1.Shygaru();
        int bt = kitap1.baspa_tabagy();
        Console.WriteLine("\""+kitap1.aty + "\" kitabi " +bt + "
baspa tabagynan turady");
        Console.ReadKey();
    }
}
```

Осы программа келесідей нәтиже береді.

```
"C# 4.0 The Complete Reference" kitabinin avtory –
Herbert Shildt ol "McGraw-Hill Education" baspasynan
2010 zhyly zharyk kordi, better sany: 976.
"C# 4.0 The Complete Reference"  kitabi 61  baspa
tabagynan turady
```

Бұл программа екі кластан: **Kitap** және **Program** тұрады. **Program** класында **Main()** әдісінің көмегімен алдымен **Kitap** класының **kitap1** экземпляры құрылады, содан кейін **kitap1** экземпляры айнымалыларын ары қарай пайдалану үшін, оларға мәндер меншіктеледі. Мұндағы атап айтатын бір жайт – **Kitap** және **Program** – екеуі екі бөлек кластар болып табылады. Олардың арасындағы жалғыз байланыс – біреуінде екіншісінің бір экземпляры құрылады. Олар жеке кластар болғанмен, **Program** класының коды **Kitap** класының мүшелерін пайдалана алады, өйткені олар ашық (**public**) кластар түрінде жарияланған. Егер оларды жариялағанда, **public** қатынасу спецификаторы көрсетілмесе, онда олардың қатынасуы **Kitap** класымен шектеліп, соның әсерінен оларды **Program** класында пайдалана алмайтын едік.

Ары қарай кодтың **Main()** әдісіндегі келесі жолын талдайық.

```
kitap1.Shygaru();
```

Бұл жолда **Shygaru()** әдісі **kitap1** объектісі үшін шақырылады. **Shygaru()** әдісі шақырылған соң, программаны басқару осы әдіске беріледі. Ал ол жұмысын аяқтаған соң, жұмысты басқару кері қарай программаның әдісті шақырған бөлігіне беріледі де, жұмыс сол әдісті шақырған кодтан ары қарай жалғасады.

Shygaru() әдісінде келесі жайтқа назар аудару керек: **aty**, **avtor**, **zhyly**, **baspa**, **bet_sany** экземплярлар айнымалыларын пайдалану нүкте-операторынсыз тікелей орындалады. Егер әдісте оның бір класында анықталған экземпляр айнымалылысы қолданылатын болса, онда ол тікелей объектіге сілтеме жасалмай-ақ, нүкте-операторысыз орындалады. Әдіс әрқашанда өз класындағы белгілі бір объектіге қатысты шақырылады. Шақыру орындалысымен, объект белгілі болады.

Жалпы, әдістен кері қайту екі шарт бойынша жүзеге асырылуы мүмкін. Біріншіден, әдістің ішкі операторларын (тұлғасын) жабатын жүйелі жақша кездескенде, мысалы, ол жоғарыдағы программада **Shygaru()** әдісінің мысалында орын алып тұр. Екіншіден, ол **return** операторы орындалғанда жүзеге асады. **return** операторының да екі түрі болады: бірі – **void** типіндегі әдістер үшін, яғни олар мән қайтармайтын әдістерде орын алады, ал екіншісі – нақты мәндер қайтаратын әдістерде кездеседі.

Void типіндегі әдісті бірден аяқтау үшін **return** операторының келесі формасын пайдалану жеткілікті.

```
return;
```

Бұл оператор орындалғанда, жұмысты басқару программаның әдісті шақыратын бөлігіне қайтарылады да, әдіс ішіндегі қалған операторлар орындалмай қалады.

Қайтарылатын мәндер программалауда әртүрлі мақсаттар үшін қолданылады. Кейбір жағдайларда, мысалы, **baspa_tabagy()** әдісіндегі сияқты, қайтарылатын мән белгілі бір есептеу нәтижесі болады, ал кейде – ол жай ғана әдістің табысты немесе қатемен аяқталғанын білдіреді, ал тағы бір кездерде – жалпы жұмыс жағдайының (қалпының) кодын береді. Қойылған мақсаттарға қарамастан, мән қайтаратын әдістерды пайдалану C# тілінде тілінде программалаудың ажырамас бір бөлігі болып табылады.

Әдістен алынған мәнді программаның шақыру бөлігіне қайтару үшін **return** операторының келесі формасы қолданылады:

```
return мән;
```

мұндағы *мән* – бұл нақты қайтарылатын мән.

Мұнда мынаған назар аударыңыз: **baspa_tabagy()** әдісі шақырылған кезде, ол меншіктеу операторының оң жақ бөлігінде көрсетіледі. Ал оның сол жағында **baspa_tabagy()** әдісі қайтаратын мәнді қабылдап алатын айнымалы орналасады. Сол себепті, келесі оператор орындалған соң,

```
int bt = kitap1.baspa_tabagy();
```

bt айнымалысында кітаптың баспа табақтарының саны (**kitap1** объектісі) сақталып қалады.

Және де енді **baspa_tabagy()** әдісінің қайтарылатын мәні **int** болатынына назар аударыңыз. Бұл программаның әдісті шақыру бөлігіне бүтін мән қайтарылатынын білдіреді. Әдістің қайтаратын мәні типінің маңызы зор, өйткені әдістің қайтаратын мәні әдісте көрсетілген мәліметтер типіне сәйкес болуы тиіс.

Мұндағы **Kitap** және **Program** кластарына бір файлда орналасу міндетті емес. Бұлардың әрқайсысын жеке файлға, мысалы, Kitap.cs және Program.cs жазып, екі файлды да өз жобамызға енгізіп, сонан кейін барып компиляциялау керек.

Параметрлерді пайдалану

Әдісті шақырған кезде оған бір немесе бірнеше мәндерді беруге болады. Әдіске берілетін мән *аргумент* деп аталады. Ал аргументті қабылдайтын айнымалы *формальді параметр* немесе жай параметр деп аталады. Параметрлер әдістің атынан кейін жай жақша ішінде жарияланады. Параметрлерді жариялау синтаксисі қарапайым айнымалыларды жариялау сияқты орындалады. Ал параметрлердің әсер ету аймағы әдістің ішкі тұлғасы (денесі) болып табылады. Әдіске аргументтерді берудің ерекше жағдайларынан басқа кездерде параметрлер кез келген басқа айнымалылар тәрізді жұмыс істей береді.

Мысал 2. Төмендегі программада параметрлі әдісті пайдалану мысалы келтірілген.

```
using System;
class Shenber {
   public int xcoord;       // шеңбер центрінің x осі
                            // бойынша координатасы
   public int ycoord;       // шеңбер центрінің y осі
                            // бойынша координатасы
```

```
   public double radius;   // шеңбер радиусы
   public double uzyndyq() { return 2 * 3.14 * radius; }
   public bool ishki_nukte(int x, int y)
   {
      double d = Math.Sqrt(Math.Pow(xcoord - x, 2) +
                  Math.Pow(ycoord - y, 2));
      if (d < radius) return true;
      return false;
   }
}
class Program {
   static void Main() {
      Shenber shenber1 = new Shenber();
      shenber1.xcoord = 2;
      shenber1.ycoord = 2;
      shenber1.radius = 3;
      int x = 3, y = 4;
      bool n = shenber1.ishki_nukte(x, y);
      Console.Write("Koordinatalary (" + x + ", " + y + ")
bolatyn nukte centri (" + shenber1.xcoord + ", " +
shenber1.ycoord + ") nuktesinde ornalasqan, radiusy " +
shenber1.radius + " bolatyn nuktenin ishinde ");
      if(n) Console.Write("zhatady");
      else Console.Write("zhatpaiady");
      Console.ReadKey();
   }
}
```

Shenber класында **ishki_nukte()** әдісі қолданылған, оған шеңбер ішінде жататын нүктенің координаталары берілсе, **true** мәнін қайтарады. Ал басқа (қарама-қарсы) жағдайда, ол **false** мәнін қайтарады.

Бұл программаның нәтижесі төмендегідей болады:

```
Koordinatalary (3,4) bolatyn nukte centri (2,2)
nuktesinde ornalasqan, radiusy  3 bolatyn nuktenin
ishinde zhatady
```

Әдісте бір немесе бірнеше параметрлер болуы мүмкін. Оның әрбір параметрі бір-бірінен үтір арқылы бөліне отырып жарияланады, олардың әрқайсысы үшін, басқаларынан өзгеше, өз типі көрсетіледі. Мысалы, **ishki_nukte(int x, int y)** коды дұрыс жазылған болып табылады.