5-6-лекция

Кластардың негізгі элементтері: өрістерді, тәсілдерді, конструкторларды, қасиеттерді пайдалану жолдары

Сұрақтар

1. Кластар туралы жалпы мәліметтер
2. Класс спецификаторлары
3. Класс объектілерін жасау
   - 3.1. Объектілерді меншіктеу және салыстыру
   - 3.2. Мәліметтер: өрістер мен константалар
4. Класс өрісінің спецификаторлары мен константалары
5. Тәсілдер
   - 5.1.Тәсілдер параметрлері
   - 5.2. Тәсілді шақыру
   - 5.3. Параметрлерді беру тәсілдері және олардың типтері
   - 5.4. Параметрлерді қолдану ережелері
6. this түйінді сөзі
7. Конструкторлар
8. Қасиеттер

1. Кластар туралы жалпы мәліметтер

Бұған дейінгі программаларда **Main** тәсілі мен бір ғана класс қолданылып жүрді. Енді кластардың құрамын, пайдаланылуын, жасалу ережелерін қарастырамыз. Көптеген түсініктер тілдің осы класс ұғымымен тығыз байланысты, оның бірсыпыра элементтері C++ тіліне ұқсас болып келеді.

- ***Класс*** тұтынушы анықтаған мәліметтер типі болып табылады. Ол бір объектінің немесе процестің нақты моделі болуы тиіс. ***Класс элементтері*** болып программадағы ***мәліметтер*** мен соларды өңдеу үшін әртүрлі әрекеттер орындайтын ***функциялар*** саналады.
- Жалпы .NET ортасының барлық кластарының бір ғана тегі (предок) бар — ол бірыңғай сатылы құрылымға негізделіп жасалған object класы.
- Оның ішіндегі кластар атаулар қайшылығы болмас үшін логикалық түрде атаулар кеңістігіне топталған : әртүрлі кеңістікте бірдей аттар бола береді. Атаулар кеңістігі қабаттасып та орналасуы мүмкін.
- Кез келген программа System атаулар кеңістігін пайдалана алады.

Класс сипаттамасы ***class*** түйінді сөзі мен класс *атынан* тұрады да, жүйелі жақша ішінде класс *тұлғасы* – оның элементтері тізімі орналасады. Бұларға қоса, кластың базалық элементтерін (тегі-предки) және әртүрлі сипаттары анықталатын, міндетті емес бірсыпыра *атрибуттары* мен *спецификаторларын* беруге болады.

Кластың жазылу форматы:

[атрибуттары] [спецификаторлары] class класс_аты [: тегі]
{
класс_тұлғасы
}

- *Класс аты* – C# тілінің жалпы ережелеріне сәйкес программалаушы тағайындайтын идентификатор.
- *Класс тұлғасы* — жүйелі жақшадағы оның элементтері сипаттамаларының тізімі.
- *Атрибуттары* класс жайлы қосымша мәлімет береді.
- *Спецификаторлар* класс қасиеттерін анықтап, басқа программа элементтері үшін мұны қолдануға болатынын немесе болмайтынын білдіреді.

Мұнда міндетті түрде көрсетілетіндері – class түйінді сөзі, класс аты және класс тұлғасы.

Класс тұлғасы — жүйелі жақшаға алынған оның элементтерінің сипаттамалары тізімі. Егер класта ешқандай элемент болмаса, тізім бос болуы мүмкін. Сонымен, кластың ең қарапайым сипаттамасы мынадай болады:

class Demo { } // бос класс

Міндетті емес класс *атрибуттары* жалпы қосымша мәліметтер береді. Олар жайлы кейінірек айтылады.

Класты атаулар кеңістігі ішінде немесе басқа класс ішінде де сипаттауға болады.

Басқа класс ішіне кіретін класс *қабатталған класс* болып табылады.

Жалпы класқа мысал

```
class Monster {
    public Monster()      // конструктор
    {
            this.name  = "Noname";
            this.health = 100;
            this.ammo = 100;
    }
  public Monster( string name ) : this()
    {
            this.name = name;
     }
  public Monster( int  health, int  ammo,
     string  name )
     {
            this.name  = name;
            this.health = health;
            this.ammo = ammo;
     }
  public  int  GetName()          // тәсіл
    {  return name;   }
  public  int  GetAmmo()          // тәсіл
     {  return ammo;}
```

```
public  int  Health {              // қасиет
   get { return health;  }
   set { if (value > 0) health = value;
           else          health = 0;
         }
 }

 public  void  Passport()              // тәсіл
     {   Console.WriteLine(
          "Monster {0} \t health = {1} \
          ammo = {2}", name, health, ammo );
      }

public override string ToString() {
       string buf = string.Format(
        "Monster {0} \t health = {1} \
      ammo = {2}", name, health, ammo);
     return buf;  }

  string name;                    // өріс
  int health, ammo;               // өріс
}
```

2. Класс спецификаторлары

Спецификаторлар класс қасиеттерін және де осы класты программаның басқа элементтері ішінде пайдалануға болатынын не болмайтынын анықтайды.

Кластың сипатталу орнына байланысты кейбір спецификаторларды қолдануға тиым салынатын кездері болады.

Енді класс спецификаторлары мен олардың сипаттамаларын қарастырып шығайық.

Класс спецификаторлары

№	Спецификатор	Сипатталуы
1	**new**	Қабатталған ішкі кластар үшін қолданылады. Тегінен мұраланған класс орнына кластың жаңа сипаттамасын береді. Объектілер иерархиясында пайдаланылады.
2	**public**	Бұл класпен қатынасуға шек қойылмайды.
3	**protected**	Қабатталған ішкі кластар үшін пайдаланылады. Бұны осы класта немесе туынды кластарда қолдануға болады.
4	**internal**	Мұны тек осы программада пайдалануға болады.
5	**protected internal**	Бұны осы класта немесе туынды кластарда қолдануға болады және осы программада пайдалануға болады.
6	**private**	Қабатталған ішкі кластар үшін пайдаланылады. Ішінде осы класс сипатталған кластардың элементтерінде қолданылады.
7	**abstract**	Абстракты класс. Объектілер иерархиясында пайдаланылады.
8	**sealed**	Туындысыз класс. Объектілер иерархиясында пайдаланылады.
9	**static**	Статикалық класс. Тілдің 2.0 нұсқасынан бастап енгізілген.

2-6 нөмірлі спецификаторлар *қатынасу спецификаторлары* деп аталады. Олар осы класқа қай жақтардан тікелей қатынас құруға болатынын анықтайды. *Қатынасу спецификаторлары* кестеде көрсетілгендей түрде қолданылуы мүмкін және де бір-біріне жалғасып араласып та тұра береді.

Алдымен тек атаулар кеңістігінде орналасып, қабаттаспай тұратын кластарды қарастырып шығайық. Мұндай кластар үшін тек екі спецификатор қолданылады, олар: public және internal. Егер ешқандай спецификатор көрсетілмесе, онда internal спецификаторы бар деп саналады.

3. Класс объектілерін жасау

Класс бірсыпыра ұқсас нақты объектілерді сипаттап, солардың ортақ қасиеттерін білдіретін жалпы түсінік болып табылады.

Класс объектілері көбінесе оның экземпляр-ры, яғни даналары деп аталады. Объектілерді программалаушылар тікелей командалар арқылы құрады немесе оларды жүйе өздігінен автоматты түрде де жасай алады. Программалаушы класс экземплярын new операциясы арқылы құрады.

Мысалы:

```
Demo   a = new Demo();       // Demo класы
                             //  экземплярын жасау
Demo   b = new Demo();       //  Demo класының
                             //  басқа экземплярын жасау
```

Класс мәліметтердің сілтемелік типтеріне жатады, олар мәліметтің өзін емес, тек адресін есте сақтайды. Сілтемелік типтегі мәліметтер үйіндіде (куча - heep) сақталады. Сонымен ***a*** мен ***b*** объектілердің адрестерін сақтайды.

Егер объектіні сақтауға жеткілікті жады көлемі бөлінбесе, онда new операциясы OutOfMemory-Exception ерекше жағдайын туындатады. Үлкен көлемді объектілерді пайдаланғанда, осындай жағдайды өңдеу әрекетін қарастыру керек.

Әрбір объект жасалғанда, оның мәліметтері сақталатын жеке жады аймағы бөлінеді. Бұған қоса, класта *статикалық элементтер* болуы мүмкін, олар барлық класс объектілері үшін бір ғана экземплярда болады. Көбінесе статикалық элементтерді *класс мәліметтері*, ал басқаларын *экземпляр мәліметтері* деп атайды.

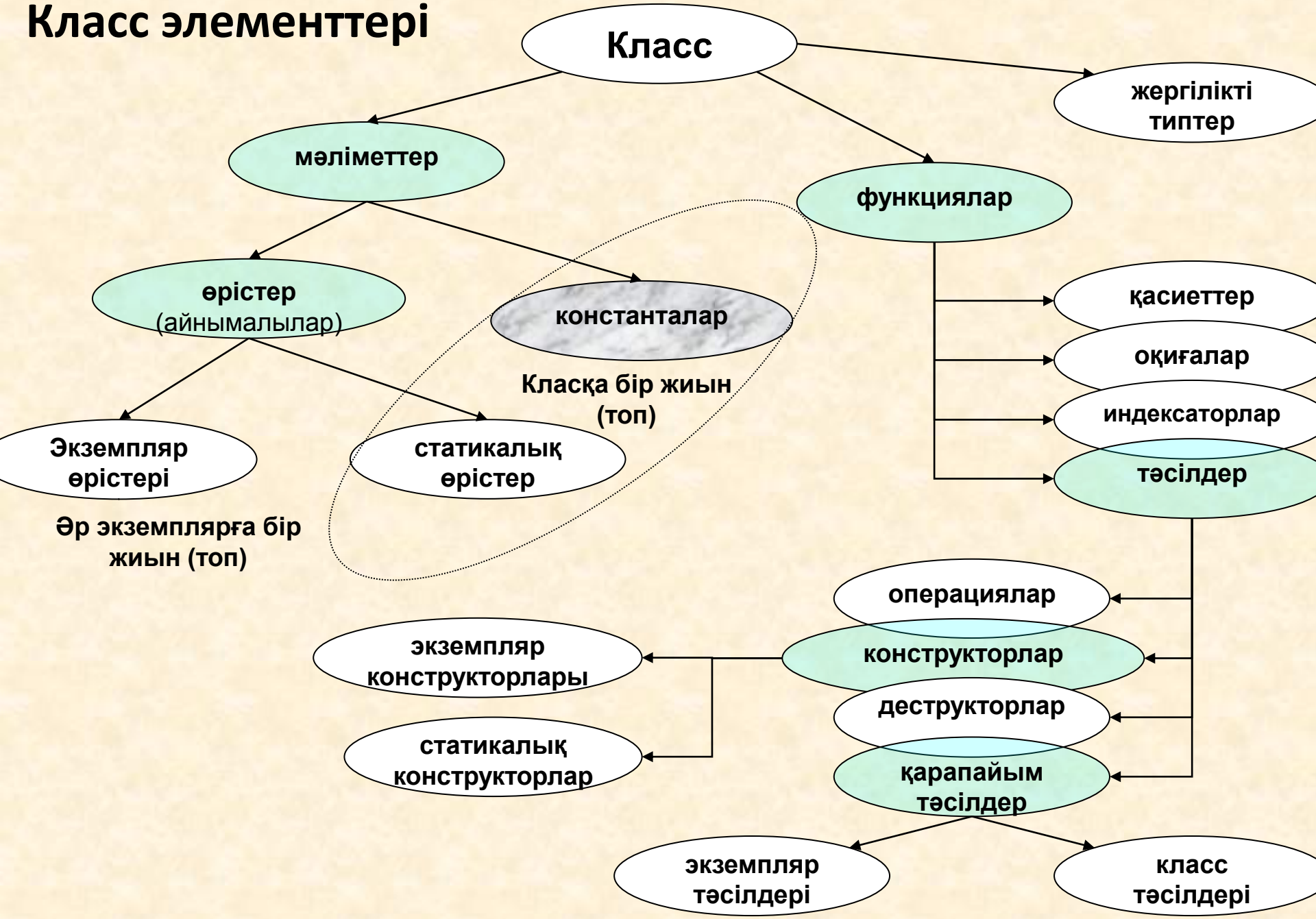
Кластың функционалдық элементтері біреу ғана болады. Класс мәліметтерімен жұмыс істеу үшін класс тәсілдері (статикалық) пайдаланылады да, ал экземпляр мәліметтерімен экземпляр тәсілдері (немесе жай ғана тәсілдер) жұмыс істейді.

Осыған дейінгі программаларда кластың бір ғана функционалдық элементі — тәсілдер ғана қолданылды. *Өрістер* мен *тәсілдер* кластың негізгі элементтері болып саналады. Класта бұлардан басқа бірсыпыра элементтерді – *қасиеттерді, оқиғаларды, индексаторларды, операцияларды, конструкторларды, деструкторларды* және де *типтерді* (келесі суретті қ.) беруге болады.

Константалар класпен байланысқан оның өзгермейтін мәндерін сақтайды.

Өрістерде класс мәліметтері болады.

Класс элементтері



Индексаторлар класс элементтерін олардың реттік нөмірлеріне қарай пайдалануды қамтамасыз етеді.

Тәсілдер класпен немесе экземплярмен атқарылатын есептеулерді орындайды.

Қасиеттер кластың сипаттамаларын – жазылу және оқылу тәсілдерін, яғни олардың берілу және алыну жолдарын анықтайды.

Конструкторлар жалпы класты немесе оның экземплярларын инициалдау әрекеттерін жүзеге асырады.

Деструкторлар объектіні өшіруге (жоюға) дейін орындалатын әрекеттерді анықтайды.

Операциялар операциялық таңбалары арқылы объектілермен атқарылатын әрекеттерді береді.

Оқиғалар класс жасайтын ескертпелерді (уведомления) анықтайды.

Типтер — класқа қатысты ішкі мәліметтер типтері.

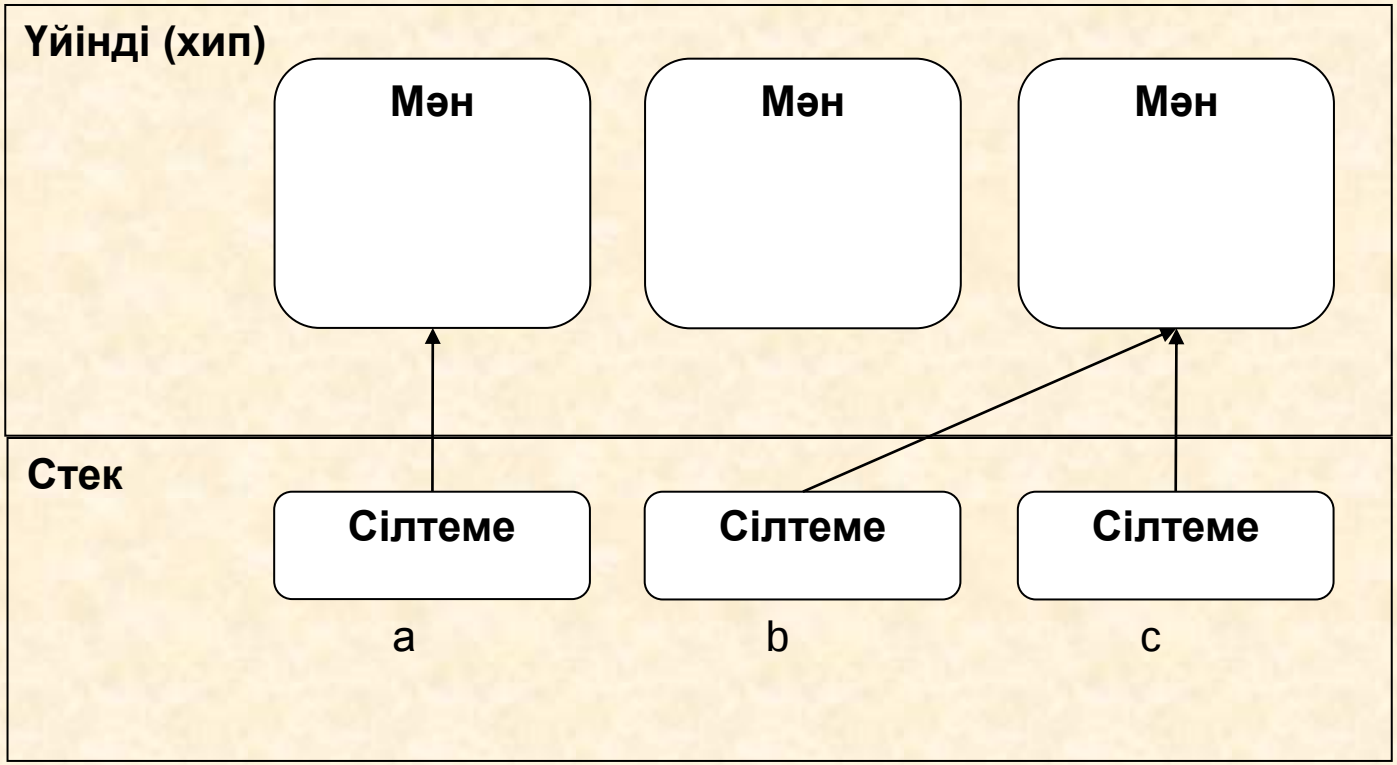
Осы кластардың алғашқы бес элемент түрлері жиі қолданылады.

Объектілерді (экземплярларды) құру мысалы

```
class Monster { ... }
 class Class1
{
        static void Main()
        {
            Monster X = new Monster();
            X.Passport();
            Monster Vasia = new Monster( "Vasia" );
            Vasia.Passport();
            Monster Masha = new Monster( 200, 200, "Masha" );
            Console.Writeline(Masha);
        }
    }
```

Программа жұмысы нәтижесі:
Monster Noname health = 100 ammo = 100
Monster Vasia health = 100 ammo = 100
Monster Masha health = 200 ammo = 200

3.1. Объектілерді меншіктеу және салыстыру



Объектілер

- b = c
- Сілтемелік типтегі шамалар тек бір жерде орналасқан мәліметтер-ге сілтеме жасап тұрса,тең болып саналады (b == c, бірақ a != b олардың мәндері тең болса да немесе екі мәнде де null-ге тең болған жағдайда да).

3.2. Мәліметтер: өрістер мен константалар

- Кластағы мәліметтер айнымалылар немесе константалар түрінде бола алады.
- Класта сипатталған айнымалылар класс *өрістері* деп аталады.
- Өрістерді сипаттау кезінде элементтердің әртүрлі сипаттамаларын беретін атрибуттар мен спецификаторларды көрсетуге болады:

[атрибуттар] [спецификаторлар] [const] типі аты

[= бастапқы_мәні]

- Алдымен барлық өрістер автоматты түрде өзіне сәйкес типтегі нөлмен инициалданады (мысалы, int типтегі өрістерге 0 меншіктеледі, ал объектілерге сілтемелерге — null мәні меншіктеледі). Мұнан кейін барып оларға тікелей инициалдау кезінде берілген мәндер меншіктеледі.

4. Класс өрісі спецификаторлары мен константалары

Спецификатор	Сипатталуы
new	Кластың мұраланған элементін жасыратын өрістің жаңа сипаттамасы
public	Элементпен қатынасуға шек қойылмайды
protected	Осы немесе туынды кластардан қатынасуға болады
internal	Осы жинақтаудан (сборки) ғана қатынасуға болады
protected internal	Осы немесе туынды кластардан және осы жинақтаудан (сборки) ғана қатынасуға болады
private	Тек осы кластан ғана қатынасуға болады
static	Кластың барлық даналары үшін бір ғана өріс
readonly	Өрісті тек оқуға ғана болады
volatile	Өріс басқа процесс немесе жүйе арқылы өзгертіле алады

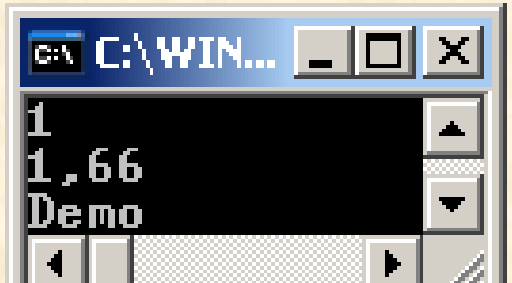
- Келісім бойынша класс элементтері жабық (private) болып саналады.
- static спецификаторымен сипатталған өрістер мен константалар барлық класс объектілері үшін бір ғана дана түрінде болады, сондықтан оларды класс аты бойынша пайдаланады.
- *Класс өрісін пайдалану үшін* қатынасу операциясы (нүкте) қолданылады. Нүктенің оң жағында өріс аты, ал сол жағында – жай өрістер үшін экземпляр аты немесе статикалық өрістер үшін – класс аты жазылады.

Келесі мысалда қарапайым Demo класы мен оның өрістерін пайдаланудың екі түрлі жолы көрсетілген. Ондағы **y** өрісі жабық болып саналады, өйткені оның типі алдында ешқандай спецификатор жоқ, сол себепті ол алдын ала келісім бойынша private болып саналады.

Класс мысалы

```
using System;
namespace Listing5_1
{    class Demo
   {  public int a = 1;                        // мәліметтер өрісі
      public const double  c = 1.66;           // константа
      public static string s = "Demo";         // кластың статикалық өрісі
      double y;                                // мәліметтердің жабық өрісі
   }
   class Class1
   {   static void Main()
      {
         Demo x = new Demo();      // Demo класының бір данасын жасау
         Console.WriteLine( x.a );     // x.a - класс өрісін пайдалану
         Console.WriteLine( Demo.c ); // Demo.c – константаны пайдалану
         Console.WriteLine( Demo.s );  // статикалық өрісті пайдалану
  }}}
```

Нәтижесі:



5. Тәсілдер

- Тәсіл — класпен немесе экземплярмен атқарылатын есептеулерді орындайтын функционалдық элемент. Тәсілдер кластың тәртібін анықтайды да, оның **интерфейсін** құрайды.
- Тәсіл — аты арқылы пайдалануға болатын, аяқталған код фрагменті. Ол бір рет жазылады да, қанша рет қажет болса, сонша рет шақырылып орындала береді.
- Бір тәсіл оған аргумент ретінде берілген әртүрлі мәліметтерді өңдей алады.

```
double a = 0.1;
double b = Math.Sin(a);
Console.WriteLine(a);
```

Тәсіл синтаксисі

[атрибуттар] [спецификаторлар] типі тәсіл_аты ([параметрлер]) тәсіл_тұлғасы

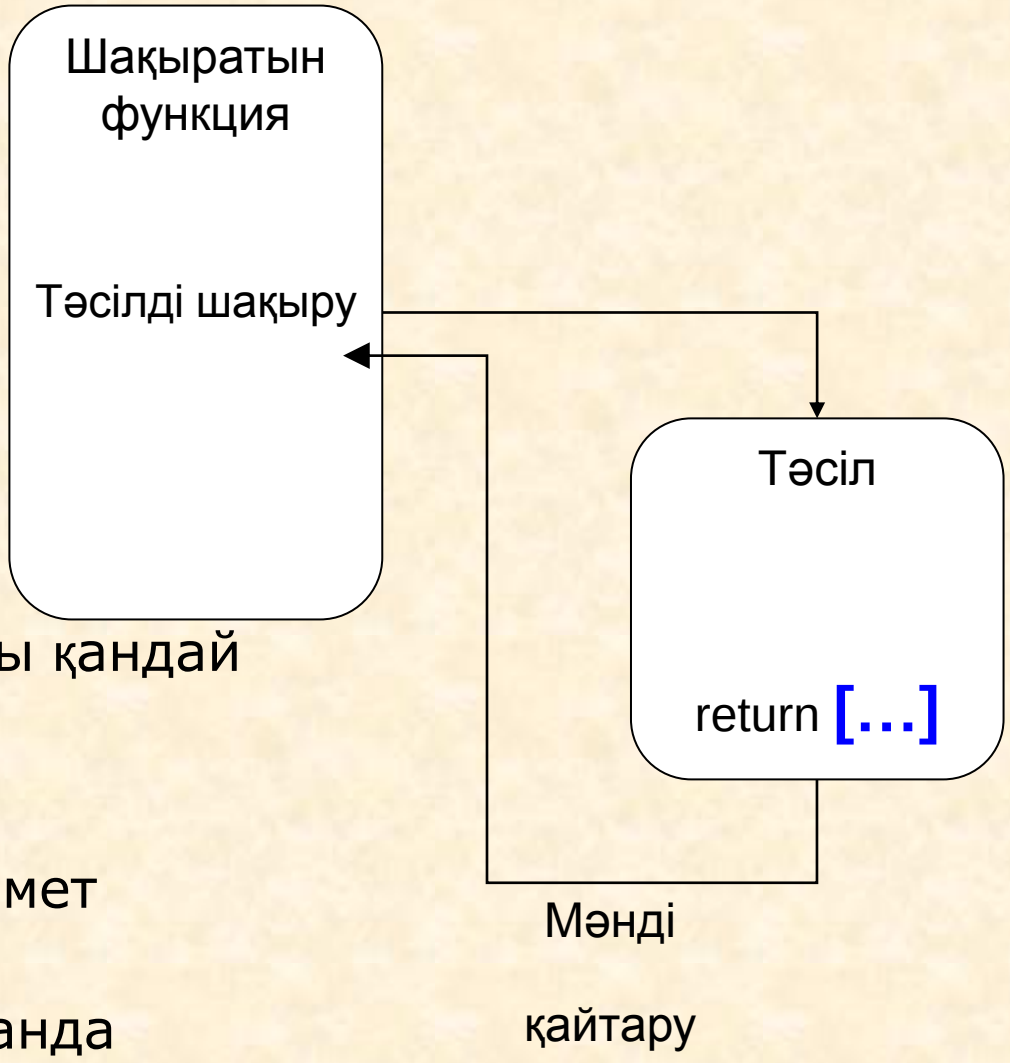
- Спецификаторлар: new, **public**, protected, internal, protected internal, private, static, virtual, sealed, override, abstract, extern.
- Класс тәсілі оның өрістерімен тікелей қатынаса алады.
- Мысал:

```
class Demo {
    double y;                              //  кластың жабық өрісі

    public void Sety( double z ) {     // кластың жабық тәсілі
        y = z;
    }
}
… Demo x = new Demo();          // басқа кластың тәсілінде...
   x.Sety(3.12); …              // тәсілді шақыру
```

Тәсілдер мысалдары

```
public void Sety(double z) {
    y = z;
}
public double Gety() {
   return y;
}
```



- Тәсіл типі осы тәсіл арқылы қандай типтегі мән есептелетінін анықтайды.
- Параметрлер тәсілмен мәлімет алмасу үшін қолданылады. Параметр – тәсілді шақырғанда соған сәйкес аргумент мәнін қабылдайтын жергілікті (локальді) айнымалы.

```
x.Sety(3.12);
double t = x.Gety();
```

5.1.Тәсілдер параметрлері

- Параметрлер тәсілге беруге болатын аргументтер мәндерінің жиынын анықтайды.
- Шақыру кезінде аргументтер тізімі параметрлер тізімі ретінде беріледі, сондықтан олар біріне-бірі сәйкес келуі тиіс.
- Әрбір параметр үшін оның должны задаваться его типі, аты және кей кездерде параметр түрі берілуі керек.
- Тәсіл аты оның параметрлері санымен, типімен және спецификатор-ларымен бірге тәсіл сигнатурасын құрайды, осылар бір тәсілдің екінші бір тәсілден айырмасы болып табылады.
- Класта сигнатуралары бірдей тәсілдер болмауы тиіс.
- Static спецификаторымен сипатталған тәсіл кластың тек статикалық өрістерін пайдалануы керек.
- Статикалық тәсіл класс аты арқылы шақырылады, ал жай тәсіл — экземпляр аты бойынша шақырылады.

Қарапайым тәсіл мысалы

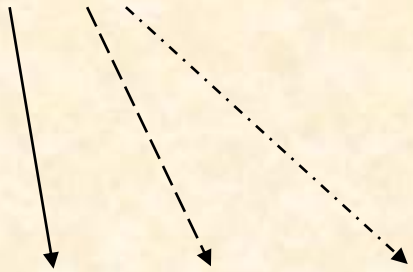
```
using System;
namespace Lisning5_2 {
class Demo    {
        public int a = 1;
        public const double c = 1.66;
        static string s = "Demo";
        double y;
        public double Gety() { return y;  }            // y-ті алу тәсілі
        public void Sety( double y_ ){  y = y_; }    // y -ті орнату тәсілі
        public static string Gets() { return s; }      // s-ті алу тәсілі
}
class Class1 {
      static void Main()
       {  Demo x = new Demo();
          x.Sety(0.12);                              // y-ті орнату тәсілін шақыру
          Console.WriteLine(x.Gety());               // y-ті алу тәсілін шақыру
          Console.WriteLine(Demo.Gets());  // s-ті алу тәсілін шақыру
//         Console.WriteLine(Gets());          // өзге кластағы тәсілді шақыру
}}}                                                  // нұсқасы
```

Нәтижесі:

5.2. Тәсілді шақыру

1. Аргументтер орнында тұрған өрнектер есептеледі.
2. Тәсіл параметрлеріне жады бөлініп беріледі.
3. Әрбір параметрге соған сәйкес аргумент беріледі. Мұнда аргументтер мен параметрлердің типтері сәйкестігі тексеріледі, қажет болса, олар түрлендіріледі. Типтер сәйкес болмаса, қате жайында диагностикалық хабарлама беріледі.
4. Тәсіл тұлғасы орындалады.
5. Егер тәсіл мән қайтаратын болса, ол оны шақыру нүктесіне беріледі; егер тәсіл типі void болса, программаны басқару шақыру нүктесінен кейінге операторға беріледі.

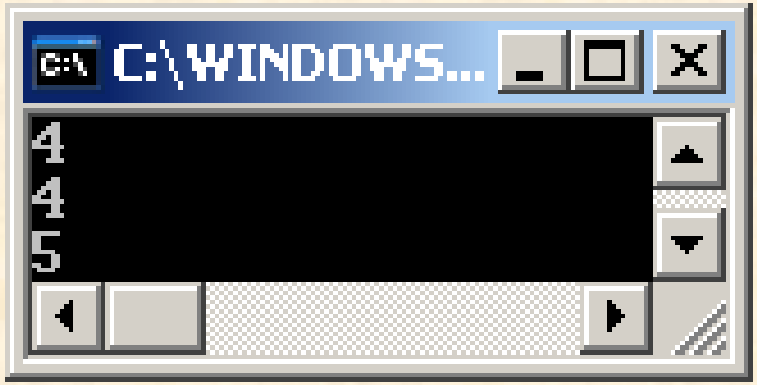
Объектіні сипаттау: SomeObj obj = new SomeObj();
Аргументті сипаттау: int b; double a, c;
Тәсілді шақыру: obj.P(a, b, c);



P тәсілінің тақырыбы: public void P(double x, int y, double z);

Тәсілге параметрлерді беру мысалы

```
using System;
namespace Lisning5_3 {
class Class1
   {   static int Max(int a, int b)       // макс. мәнді таңдау
       {
          if ( a > b ) return a;
          else         return b;
       }
       static void Main()
       {
          int a = 2, b = 4;
          int x = Max( a, b );          // Max тәсілін шақыру
          Console.WriteLine( x );                  // нәтижесі: 4
          short t1 = 3, t2 = 4;
          int y = Max( t1, t2 );       // үйлесімді тип пар-рлері
          Console.WriteLine( y );                  // нәтижесі : 4
          int z = Max( a + t1, t1 / 2 * b );   // өрнек
          Console.WriteLine( z );                  // нәтижесі : 5
}}}
```



5.3. Параметрлерді беру тәсілдері және олардың типтері

Параметрлерді беру тәсілдері: мән бойынша және сілтеме бойынша.

- *Параметрлерді мән бойынша беру кезінде* тәсіл аргументтер мәнінің көшірмелерін алады да, тәсіл операторлары осы көшірмелермен жұмыс істейді.
- *Параметрлерді сілтеме (адрес) бойынша беру кезінде* тәсіл аргументтер адресінің көшірмесін алады да, сол адрестердегі аргументтерді пайдаланады.

C# тілінде параметрлердің төрт типі бар:

- мән-параметрлер;
- сілтеме-параметрлер (**ref**);
- нәтижелік (шығыс) параметрлер (**out**);
- жиым-параметрлер (**params**).

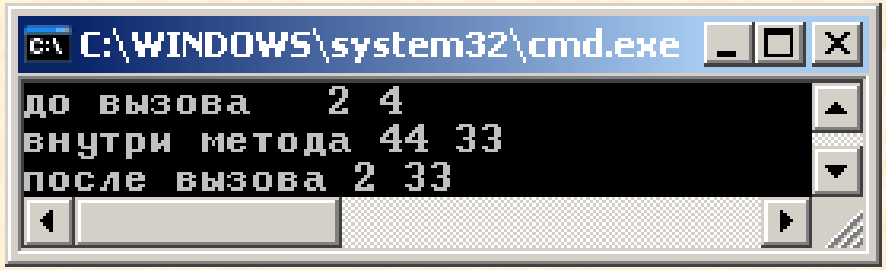
Түйінді сөздер параметрлер типтерін сипаттаудың алдында тұрады. Егер ол болмаса, параметр мән-параметр болып есептеледі. Мысалы:

```
public int Calculate( int a, ref int b, out int c, params int[] d ) …
```

Мысал: мән-параметрлер мен ref сілтемелері

```
using System;
namespace Listing5_4
{   class Class1
    {   static void P( int a, ref int b )
        {
            a = 44; b = 33;
            Console.WriteLine( "тәсіл ішінде {0} {1}", a, b );
        }
        static void Main()
        {
            int a = 2, b = 4;
            Console.WriteLine( "шақыруға дейін   {0} {1}", a, b );
            P( a, ref b );
            Console.WriteLine("шақырудан кейін  {0} {1}", a, b );
}}}
```

Программа жұмысы нәтижесі:
шақыруға дейін 2 4
тәсіл ішінде 44 33
шақырудан кейін 2 33

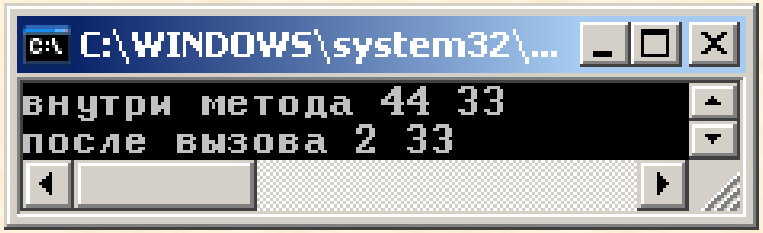


Мысал: out шығыс параметрлері

```
using System;
namespace Listing5_5
{   class Class1
    {   static void P( int x, out int y )
        {
            x = 44; y = 33;
            Console.WriteLine( " тәсіл ішінде {0} {1}", x, y );
        }
        static void Main()
        {
            int a = 2, b;          // b-ны инициалдау қажет емес

            P( a, out b );
            Console.WriteLine( "шақырудан кейін  {0} {1}", a, b );
}}}
```

Программа жұмысы нәтижесі:
тәсіл ішінде 44 33
шақырудан кейін 2 33



5.4. Параметрлерді қолдану ережелері

1. Мән-параметрлер үшін мән бойынша параметр беру қолданылады. Бұл мүмкіндік тәсілдің бастапқы мәліметтері үшін қолданылады.

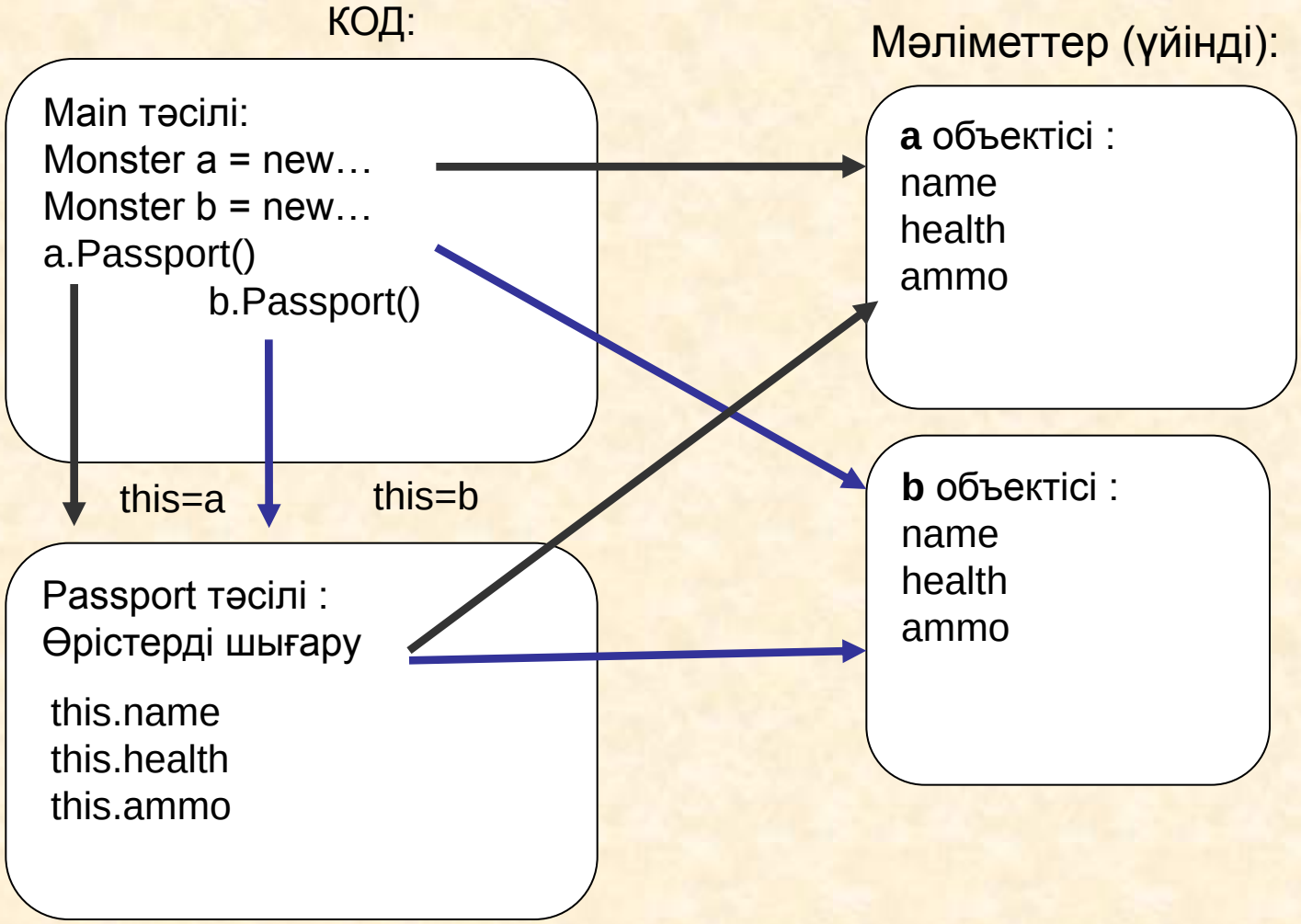
- Тәсілді шақыру кезінде мән бойынша берілетін параметр орнында өрнек болуы мүмкін (және де оның жеке жағдайы ретінде — айнымалы немесе константа). Мұнда өрнек типі параметр типіне келтірілуі керек.

2. Сілтеме-параметрлер және нәтижелік (шығыс) параметрлер адрес бойынша беріледі. Мұндай мүмкіндік тәсілдің қосалқы нәтижелерін (побочные результаты) беру кезінде қолданылады.

- Тәсілді шақыру кезінде **ref** сілтемесі бойынша берілетін параметр орнында тек сол типтегі инициалданған айнымалы аты ғана болуы мүмкін. Параметр аты алдында ref түйінді сөзі көрсетіледі.
- Тәсілді шақыру кезінде **out** нәтижелік (шығыс) параметр орнында тек сол типтегі айнымалы аты ғана болуы мүмкін. Оны инициалдау қажет емес. Параметр аты алдында out түйінді сөзі көрсетіледі.

6. this түйінді сөзі

Тәсілдің ол өңдеуге тиіс объект өрісімен жұмыс істеуін қамтамасыз ету үшін, сол тәсілге автоматты түрде **this** жасырын параметрі беріледі, ол функцияны шақырған объектіге сілтеме жасап тұрады.



this сөзін нақты түрде пайдалану

this параметрі тікелей түрде мынадай жағдайларда қолданылады:

```
// тәсілден оны шақырған объектіге сілтеме қайтару үшін:
class Demo
{       double y;
        public Demo T()    { return this; }
//  аты тәсіл параметрі атымен бірдей өрісті идентификациялау үшін:
        public void Sety( double y ) { this.y = y; }
}
```

7. Конструкторлар

Конструктор объектіні инициалдау үшін қажет. Ол класс объектісін жасау кезінде new операциясы арқылы шақырылады. Конструктор аты класс атымен бірдей болады.

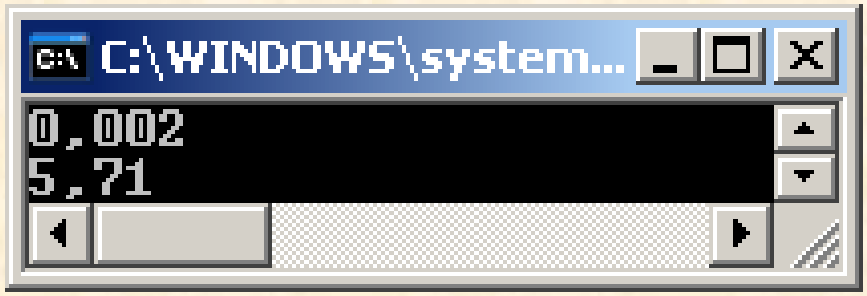
Конструкторлар қасиеттері:

- Конструктор ешқандай мән (тіпті void типін де) қайтармайды.
- Кластың, инициалдаудың әр түрлері үшін әртүрлі параметрлері бар, бірнеше конструкторлары болуы мүмкін.
- Егер программалаушы бірде-бір конструкторды көрсетпесе немесе кейбір өрістер инициалданбаса, мәндік типтегі өрістерге — нөл, ал сілтемелік типтегі өрістерге — null мәні меншіктеледі.
- Параметрсіз шақырылатын конструктор (үнсіз) келісім бойынша алынған конструктор деп аталады.

Конструкторы бар класс мысалы

```
using System;
namespace Listing5_6  {
class Demo
   {
      public Demo( int a, double y )        // конструктор
      {
         this.a = a;
         this.y = y;
      }
      int a;
      double y;
   }

   class Class1
   {   static void Main()
      {
         Demo a = new Demo( 300, 0.002 );  // конструкторды шақыру
         Console.WriteLine( a.Gety() );       // нәтиже: 0,002
         Demo b = new Demo( 1, 5.71 );       // конструкторды шақыру
         Console.WriteLine( b.Gety() );       // нәтиже : 5,71
  } } }
```



Екі конструкторы бар класс мысалы

```
class Demo
   {
      public Demo( int a )                  //  1 конструктор
      {
         this.a = a;
         this.y = 0.002;
      }
      public Demo( double y )                // 2 конструктор
      {
         this.a = 1;
         this.y = y;
      }
      ...
   }
      ...
         Demo x = new Demo( 300 );    // 1 конструкторды шақыру
         Demo y = new Demo( 5.71 );   // 2 конструкторды шақыру
```

Жалпы класс мысалы

```
class Monster {
   public Monster()      // конструктор
   {
         this.name  = "Noname";
         this.health = 100;
         this.ammo = 100;
   }
 public Monster( string name ) : this()
   {
         this.name = name;
    }
 public Monster( int health, int ammo,
    string name )
    {
         this.name  = name;
         this.health = health;
         this.ammo = ammo;
    }
 public int GetName()          // тәсіл
   {  return name;   }
 public int GetAmmo()        // тәсіл
    {  return ammo;}
```

```
public int Health {                  // қасиет
   get { return health;  }
   set { if (value > 0) health = value;
           else                 health = 0;
         }
 }

 public void Passport()                 // тәсіл
     {   Console.WriteLine(
  "Monster {0} \t health = {1} \
    ammo = {2}", name, health, ammo );
      }

public override string ToString(){
     string buf = string.Format(
       "Monster {0} \t health = {1} \
     ammo = {2}", name, health, ammo);
    return buf;  }

  string name;
  int health, ammo;
}
```

8. Қасиеттер

- Қасиеттер класс өрістерімен қатынас құру үшін пайдаланылады. Көбінесе, қасиет жабық өрістермен қатынас құру тәсілдерін анықтайды.
- Қасиет синтаксисі:

[спецификаторлар] типі қасиет_аты

{

[get қатынасу_коды]

[set қатынасу_коды]

}

Қасиеттерді пайдалану кезінде, автоматты түрде онда көрсетілген оқу (**get)** және орнату (**set)** блоктары шақырылады.

- Мұнда не get, не set бөлігі болмауы мүмкін, бірақ екеуінің де бірден жоқ болуы мүмкін емес. Егер set бөлігі болмаса, қасиет тек оқылады (read-only), ал егер get бөлігі болмаса – тек жазылады (write-only).

Қасиеттерді сипаттау мысалы

```
public class Button: Control
{  private string caption;      // қасиетпен байланысты  өріс
   public string Caption {      // қасиет
      get { return caption; }  // қасиетті алу жолы

      set                              // қасиетті орнату жолы
         { if (caption != value) { caption = value; }
}}    ...
```

Программада қасиет класс өрісі тәрізді болып көрінеді:

```
Button ok = new Button();
ok.Caption = "OK";              // қасиетті орнату тәсілі шақырылған
   свойства
string s = ok.Caption;          // қасиетті алу  тәсілі шақырылған
```

Класс мысалы

```
class Monster {
   public Monster()      // конструктор
   {
         this.name  = "Noname";
         this.health = 100;
         this.ammo = 100;
   }
  public Monster( string name ) : this()
   {
         this.name = name;
    }
  public Monster( int health, int ammo,
    string name )
    {
         this.name  = name;
         this.health = health;
         this.ammo = ammo;
    }
  public int GetName()          // тәсіл
   {  return name;   }
  public int GetAmmo()         // тәсіл
    {  return ammo;}
```

```
public int Health {              // қасиет
   get { return health;  }
   set { if (value > 0) health = value;
         else           health = 0;
        }
 }

 public void Passport()              // тәсіл
    {   Console.WriteLine(
  "Monster {0} \t health = {1} \
    ammo = {2}", name, health, ammo );
     }

public override string ToString(){
    string buf = string.Format(
      "Monster {0} \t health = {1} \
    ammo = {2}", name, health, ammo);
    return buf;  }

  string name;
  int health, ammo;
}
```

Тындағандарына

рахмет!