Тел: +7 (707) 900 92 67
Почта: saken.yan@yandex.com

9 ЛЕКЦИЯ

C# ТІЛІНІҢ НЕГІЗДЕРІ - ҚАЙТА ОРАЛУ!

§9. C# тіліндегі класстар ұғымы.

Кіріспе.

C# - бұл объектті бағытталған бағдарламалау тілі болып табылады. Бұл дегеніміз, C# бағдарламасы өзара байланысқан объектілер ретінде ұсынылуы мүмкін.

Объектілердің сипаттамасы (физикалық және т.б.) класс болып табылады, ал объект өзі сол класстың нақты нұсқасы (экземпляр) болады.

Мысал.

Мынадай мысалды қарастырайық. Әр адамның аты, жасы, басқа да ерекшеліктері бар. Яғни адамды шаблон немесе класс деп қарастыруға болады. Бұл класстың нақты нұсқасы әр түрлі болуы мүмкін, мысалы, адамдардың аттары, жастары және де басқа да ерекшеліктері әр түрлі. Сонымен нақты адам (іс жүзінде осы класстың нақты нұсқасы) осы кластың объектісі болады.

Негізінде, класс программисттің анықтаған жаңа типі болып табылады. Класс келесідей анықталады:

```
class Person
{
  . . . . .
}
```

Кластың барлық функционалдығы оның - жолдарымен, қасиеттерімен, әдістерімен және оқиғаларымен анықталаады.

Мысал. Адам атты кластың жолдары мен әдістерін анықтайық:

```
class Person
{
    public string name; //Бұл жол(қасиет) адамның атын білдіреді.
    public int age;     //Бұл жол(қасиет) адамның жасын білдіреді.

   // Бұл әдісті адам орындай алады деп программист шешті. Бұл әдіс нақты
   // адамның аты мен жасы туралы ақпаратты консольге шығарады.
    public void GetInfo(){

                Console.WriteLine("Имя: {0}  Возраст: {0}", name, age);
    }
}
```

Бұл жерде Person классы адамды сипаттайды. name жолы адамның атын, ал age жолы адамның жасын білдіреді. GetInfo() әдісі осы деректерді консольға шығарады. Person класынан тыс жерден осы кластын деректеріне қол жеткізу үшін (яғни оларды оқу, өзгерту) айнымалылар(жолдар) мен әдістер public модификаторымен анықталады.

Класс конструкторлары.

Қарапайым әдістерден басқа, класстардың конструктор деп аталатын арнайы әдістері болады. Біз осы кластың жаңа объектісін құру үшін конструкторларды қолданамыз. Конструкторлар инициализациялау операциясын орындайды.

Егер класс ішінде конструктор жазылмаса, вертуалды машина автоматты түрде осы класс үшін стандартты параметрсіз, денесіз конструкторды өзі құрып береді. Алайда мұндай конструкторлардан басқа, біз қажетті конструкторларды өзіміз анықтай аламыз:

```
class Person
{
    public string name;
    public int age;

    //Параметрсіз конструктор  .
    public Person() {
        name = "Неизвестно";
        age = 18;
    }

    // Параметрлері бар конструктор.
    public Person(string name, int age) {
        this.name = name;
        this.age = age;
    }

    public void GetInfo()
    {
        Console.WriteLine("Имя: {0}  Возраст: {0}", name, age);
    }
}
```

Кластың объектілерін инициализациялау.

Класстың объектілерін тудыру үшін new деген кілт сөзі қолданылады, содан кейін конструкторлардың бірі шақырылады. Мысалы:

```
static void Main(string[] args)
{
  //Инициализация объекта person1 через конструктор без параметров.
  Person person1 = new Person();

  //Инициализация объекта person2 через конструктор с параметрами.
  Person person2 = new Person("Tom", 22);

  person1.GetInfo();//Вызов метода GetInfo() у объекта person1
  person2.GetInfo();//Вызов метода GetInfo() у объекта person2

  Console.ReadKey();
}
```

Статикалық кластар

Статикалық класс негізінде статикалық емес класспен бірдей келеді, бір айырмашылығы бар: біз статикалық кластың объектілерін құра алмаймыз. Басқаша айтқанда, біз new операторын кластың объектілерін құру үшін пайдалана алмайсыз. себебі бұндай кластың объектілері жоқ, статикалық кластың мүшелеріне класс атауы арқылы қол жеткізіледі.

Мысалы, егер сізде StaticClass деп аталатын статикалық класс болса, және оның MethodA деп аталатын public модификаторымен белгіленген статикалық әдісі болса, онда осы әдісті шақыру үшін келесі төменде көрсетілгендей орындайсыз:

```
//Статикалық класс
public static class StaticClass
{
     //Статикалық кластың әдісі
     public static void MethodA()
     {
      . . . . . .
      }
}

static void Main(string[] args)
{
     //Статикалық класстың әдісін шақыру.
     StaticClass.MethodA();
}
```

Қол жеткізу модификаторлары.

Кластың барлық мүшелері - жолдар, әдістер, қасиеттер модификаторларға ие болады. Модификаторлары класс мүшелері үшін көріну аумағын орнатуға мүмкіндік береді. Яғни, модификаторлар берілген айнымалыны немесе әдісті қолдануға болатын аймақты анықтайды. C# тілінде келесі модификаторлары қолданылады: public, private, protected, internal.

Операторды қайта жүктеу.

Класстарды біз операторларды қайтадан жүктей аламыз. Мысалы, бізде Counter деген класс бар делік:

```
class Counter
{
    public int Value { get; set; }
}
```

Қарапайым мысал ретінде бұл кластың Value деген int типтес бір ғана айнымалысы немесе қасиеті бар болсын делік. Айталық, бізде Counter класының екі объектісі бар болсын делік. Осы екі объектілерге салыстыру және қосу және т.б. стандартты амалдарын қолдану үшін, олардың Value қасиеттеріне негізделген операцияларды орындау қажет. Мысалы:

```
Counter c1 = new Counter { Value = 23 };
Counter c2 = new Counter { Value = 45 };

bool result = c1 > c2;
Counter c3 = c1 + c2;
```

Алайда, Counter объектілері үшін салыстыру да, қосу операцияларын да мүмкін емес. Бұл операцияларды бірқатар қарабайыр типтер үшін қолдануға болады. Мысалы, әдетте біз сандық мәндерді қоса аламыз, бірақ компилятор күрделі типті объектілерді - кластар мен құрылымдарды(структуралар) қалай қосу, салыстыру және т.б. керек екендігін білмейді. Бұл үшін бізге қажет операторларды қайта жүктеу керек. Операторларды қайта жүктеу мына түрде орындалады:

```
public static возвращаемый_тип operator оператор(параметры)
{   }
```

Мысалы, `Counter` класы үшін бірқатар операторларды қайта жүктейік:

```
class Counter
{
    public int Value { get; set; }

    public static Counter operator +(Counter c1, Counter c2)
    {
        return new Counter { Value = c1.Value + c2.Value };
    }
    public static bool operator >(Counter c1, Counter c2)
    {
        return c1.Value > c2.Value;
    }
}
```

Мысал. Мысал ретінде матрица деп аталатын математикалық объектіні сипаттайтын матрица класын жазайық. Яғни, біздің матрица класы матрицаның қасиеттеріне ие болуы керек.

```
class Program
{
      public class matrix
     {
        public int m;
        public int n;

        public double[,] array;

        public matrix(int m, int n)
        {
            //создает нулевую матрицу по указанному размеру:
            if (m <= 0) {
                Console.WriteLine("Err: wrong dimension of matrix");
                throw new Exception("Err: wrong dimension of matrix");
            }
            if (n <= 0) {
                Console.WriteLine("Err: wrong dimension of matrix");
                throw new Exception("Err: wrong dimension of matrix");
            }

            this.m = m; this.n = n;
            this.array = new double[this.m, this.n];

        }

        //Вывод матрицы с клавиатуры
        public void Print()
        {
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    Console.Write(mass[i, j] + "\t");
                }
                Console.WriteLine();
            }
        }

         //Ввод матрицы с клавиатуры
        public void Read()
        {
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    Console.WriteLine("Введите элемент матрицы {0}:{1}", i+1, j+1);
                    mass[i, j] = Convert.ToDouble(Console.ReadLine());
                }
            }
        }

        //Метод для нахождения определителя матрицы.
        public double Determinant(){
        //Алгоритм для нахождения определителя матрицы.
        }


        //Метод для нахождения обратной матрицы.
        public matrix Inverse(){
        //Алгоритм для нахождения обратной матрицы.
```

```
        }

        // Екі матрицаны қосу операторын қайта жүктеу:
        public static matrix operator +(matrix  A, matrix  B)
        {
            matrix Result;

            if
            (   (A.m != B.m) ||
                (A.n != B.n)
            )
            { throw new Exception("wrong dimention to do operation +"); }
            else
            {
                Result = new matrix(m, n);
                for (int i = 0; i < m; i++)
                for (int j = 0; j < n; j++)
                Result.array[i, j] = A.array[i, j] + B.array[i, j];
            }

            return Result;
        }


    }//matrix


    static void Main(string[] args)
    {
      matrix A = new matrix(5,5);
      matrix B = new matrix(5,5);

      A.Read();
      B.Read();

      matrix C = A + B;

      C.Print();
    }

}//Program
```

Лаборатория№9:

Матрица классы үшін екі матрицаны көбейту, екі матрицаны азайту және матрицаны санға көбейту операторларын қайта жүктеңіз.