Объектіге-бағытталған бағдарламалауға кіріспе

2-сабақ

Пәндік аймақ матрицасы

Дағдылар / түсініктер	MTA емтиханының мақсаты
Объектілерді түсіну	Класс негіздерін түсіну (2.1)
Сілтемелер мен құндылықтарды(Values) түсіну	Компьютер қоймасын және мәліметтер типтерін түсіну (1.1)
Инкапсуляцияны түсіну	Инкапсуляцияны түсіну (2.4)
Мұрагерлікті түсіну	Мұрагерлікті түсіну (2.2)
Полиморфизмді түсіну	Полиморфизмді түсіну (2.3)
Интерфейстерді түсіну	Интерфейстерді түсіну (2.4)

ОБъект

- Объектіге бағытталған бағдарламалау - бұл объектілерді қолдануға мүмкіндік беретін бағдарламалау әдісі.
- Объектілер - қасиеттерден, әдістерден және оқиғалардан тұратын дербес деректер құрылымы.

❖ Қасиеттер объект ұсынатын деректерді көрсетеді
❖ Әдістер объектінің әрекетін анықтайды
❖ Оқиғалар объектілер арасындағы байланысты қамтамасыз етеді

Кластар

- Класс объектінің жобасын анықтайды.
- Класс объектілер қалай құрылу керектігін және олар қалай қалай жұмыс жасау керектігін анықтайды.
- Объект класс экземпляры ретінде де белгілі.

C # класын анықтау

```
class Rectangle
{
    private double length;
    private double width;

    public Rectangle(double l, double w)
    {
        length = l;
        width = w;
    }

    public double GetArea()
    {
        return length * width;
    }
}
```

Әдістер

- Әдіс дегеніміз - бірнеше операторлардан тұратын код блогы.
- Әдіс класс қолдайтын әрекеттерді немесе операцияларды анықтайды.
- Әдіс жақшадағы қолжетімділік деңгейін, қайтарылатын мәлімет типін, әдіс атауын және қосымша параметрлер тізімін, содан кейін фигуралық жақшалармен қоршалған код блогын көрсету арқылы анықталады.

Әдіс мысалы

- InitFields әдісі екі параметрді қабылдайды және деректер өрісінің ұзындығы мен енін тиісті түрде белгілеу үшін параметр мәндерін пайдаланады.
- Егер әдістің қайтарылатын мән типі void болса, қайтару операторын мәнсіз пайдалануға болады.
- Егер InitFields әдісіндегідей return операторы қолданылмаса, әдіс код блогының соңына жеткенде орындалуды тоқтатады.

```
public void InitFields(double l, double w)
{
    length = l;
    width = w;
}
```

Конструкторлар

- Конструкторлар - бұл класстың жаңа экземпляры құрылған кезде орындалатын арнайы класс әдістері.
- Конструкторлар мәліметтерді - объект мүшелерін инициализациялау үшін қолданылады.
- Конструкторлар атауы класс атауымен сәйкес аталуы керек, бірақ оларда қайтару типі жоқ.
- Класс үшін әрқайсысы жеке сигнатураға ие бірнеше конструкторларды анықтауға болады.

```
class Rectangle
{
    private double length;
    private double width;

    public Rectangle(double l, double w)
    {
        length = l;
        width = w;
    }
}
```

Объектілерді құру

- Объектілерге олардың қалай құрылу керектігін анықтайтын шаблон қажет.
- Бір шаблоннан жасалған барлық объектілер бірдей көрінеді және әрекет етеді.

```
class Program
{
    static void Main(string[] args)
    {

        Rectangle rect = new Rectangle(10.0, 20.0);
        double area = rect.GetArea();
        Console.WriteLine("Area of Rectangle: {0}",
            area);
    }
}
```

Қаситеттер

- Қасиеттер - бұл мәліметтер өрісі ретінде қолжеткізуге болатын, бірақ әдіс ретінде кодты қамтитын класс мүшелері.
- Қасиеттің екі қол жетімділік әдісі бар: get және set. Get қолжетімділік әдісі қасиет мәнін қайтару үшін қолданылады, ал set қолжетімділік әдісі қасиетrt жаңа мән тағайындау үшін қолданылады.

```
class Rectangle
{
    private double length;

    public double Length
    {
        get
        {
            return length;
        }
        set
        {
            if ( value > 0.0)
                length = value;
        }
    }
}
```

this кілттік сөзі

- This кілттік сөзі - класстың ағымдағы экземпплярына сілтеме.
- Бұл түйінді сөзді ағымдағыобъекттінің кез келген мүшесіне сілтеме жасау үшін пайдалануға болады.

```
class Rectangle
{
    private double length;
    private double width;

    public Rectangle(double l, double w)
    {
        this.length = l;
        this.width = w;
    }
}
```

Делегаттар

- Делегаттар дегеніміз - арнайы қолтаңбасы бар әдіске сілтемені қамтуы мүмкін арнайы объектілер.

```
public delegate void RectangleHandler(Rectangle rect);
```

- Мұнда сіз RectangleHandler делегатын анықтайсыз және ол void қайтаратын және Rectangle типінің бір ғана параметрін қабылдайтын әдіске сілтемені қамтуы мүмкін.

```
public void DisplayArea(Rectangle rect)
{
    Console.WriteLine(rect.GetArea());
}
```

- DisplayArea әдісінің қолтаңбасы RectangleHandler делегатына сәйкес келеді, сондықтан оны оның бір экземплярына тағайындауға болады.

Оқиғалар

- Оқиғалар - бұл класс үшін қызықты нәрсе болған кезде басқа кластарға немесе объектілерге хабарлау тәсілі.
- Хабарлама жіберетін класс оқиға жариялаушы деп аталады.
- Хабарламаны алған класс оқиға жазылушысы деп аталады.

```
class Rectangle
{
    public event EventHandler Changed;
    private double length;
    public double Length
    {
        get
        {
            return length;
        }
        set
        {
            length = value;
            Changed(this, EventArgs.Empty);
        }
    }
}
```

Оқиғаларға жазылу

- Оқиғаларды өңдеуші әдіс қолтаңбасы оқиға делегатының талаптарына сәйкес келеді.

```
class Program
{
    static void Main(string[] args)
    {
        Rectangle r = new Rectangle();
        r.Changed += new EventHandler(r_Changed);
        r.Length = 10;
    }

    static void r_Changed(object sender, EventArgs e)
    {
        Rectangle r = (Rectangle)sender;
        Console.WriteLine(
            "Value Changed: Length = {0}",
            r.Length);
    }
}
```

Атаулар кеңістігі

- Атаулар кеңістігі - бұл кодты ұйымдастыруға және глобальды ерекше класс атауларын жасауға мүмкіндік беретін тіл элементі.
- .NET Framework өзінің барлық кластарын ұйымдастыру үшін аттар кеңістігін пайдаланады.

- System аттар кеңістігі барлық іргелі кластарды топтайды.
- System.Data аттар кеңістігі мәліметтерге қол жеткізу үшін кластарды ұйымдастырады.
- System.Web аттар кеңістігі веб-кластарда қолданылады.

```
namespace CompanyA
{
    public class Widget { }
}

namespace CompanyB
{
    public class Widget { }
}
```

Static мүшелері

- static кілттік сөзі жекелеген объектілерге емес, кластың өзіне тиесілі мүшелерді жариялау үшін қолданылады.
- Класс экземпляры құрылған кезде, экземплярдың әр өрісі үшін жеке көшірме жасалады, бірақ барлық экземплярға статикалық өрістің тек бір көшірмесі беріледі.
- Статикалық мүшеге экземпляр объектісі арқылы сілтеме жасау мүмкін емес. Оның орнына статикалық мүшеге класс атауы арқылы сілтеме жасалады.

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(Rectangle.ShapeName);
    }
}

class Rectangle
{
    public static string ShapeName
    {
        get { return "Rectangle"; }
    }
}
```

Құндылықтар мен сілтемелер

- Мән типі деректерді тікелей өзінің жадында сақтайды.
- Сілтеме түрлері тек жадтағы орынға сілтемені сақтайды. Нақты деректер сілтеме жасалған жадта сақталады.
- Сілтеме типті айнымалыныны дәл сол типті басқа айнымалыға көшіргенде, тек сілтемелер көшірмеленеді.Нәтижесінде көшірмеленген соң екі айнымалы да бір ғана объектіні көрсететін болады.

```
class Program
{
    public class Rectangle
    {
        public double Length { get; set; }
        public double Width { get; set; }
    }

    static void Main(string[] args)
    {
        Rectangle r1, r2;
        r1 = new Rectangle { Length = 10.0, Width = 20.0 };
        r2 = r1;
        r2.Length = 30;
        Console.WriteLine(r1.Length);
    }
}
```

```
class Program
{
    public struct Rectangle
    {
        public double Length { get; set; }
        public double Width { get; set; }
    }

    static void Main(string[] args)
    {
        Rectangle r1, r2;
        r1 = new Rectangle { Length = 10.0, Width = 20.0 };
        r2 = r1;
        r2.Length = 30;
        Console.WriteLine(r1.Length);
    }
}
```

Инкапсуляция

- Инкапсуляция - бұл өзгеруі мүмкін конструктивті шешімдерді жасыру мақсатында класқа немесе оның мүшелеріне кіруді шектейтін механизм.
- Қол жетімділік модификаторлары қай типті немесе тип элементін қолдануға болатындығын анықтайды.

Қол жетімділік модификаторлары	Сипаттамасы
public	Қолжетімділік шектеулі емес.
private	Қол жетімділік тек класс ішінде ған мүмкін болады.
protected	Қолжетімділік класс және одан тура немесе жанама түрде туындайтын кластарда (мұрагерлеу) ғана мүмкін болады.
internal	Қолжетімділік берілген жинақта(сборка) кодпен шектелген.
protected internal	protected және internal комбинациясы - яғни бір жинақтағы кез-келген кодпен және тек басқа жинақта алынған туынды кластармен қатынау шектелген.

Мұрагерлік

- Мұрагерлік - бұл , ол классты бір рет жасап, содан кейін бұл кодты жаңа кластардың негізі ретінде қайта-қайта қолданадуға мүмкіндік беретін ООП функциясы.
- Функционалдығы мұраланған класс базалық класс деп аталады.
- Функционалдылықты мұра ететін класс туынды класс деп аталады.
- Туынды класс негізгі кластан ерекшеленетін қосымша функцияларды да анықтай алады.Класстан айырмашылығы, құрылымдар мұрагерлікті қолдамайды.

Мұрагерлік-мысал

```
class Polygon
{
    public double Length { get; protected set; }
    public double Width { get; protected set; }
}

class Rectangle : Polygon
{
    public Rectangle(double length, double width)
    {
        Length = length;
        Width = width;
    }

    public double GetArea()
    {
        return Width * Length;
    }
}
```

Абстрактылы кластар

- Абстрактылы кластар негізгі кластың жалпы анықтамасын береді, оны бірнеше туынды кластар ортақ қолдана алады.
- Абстрактылы кластар көбінесе толық емес орындалуды қамтамасыз етеді.
- Абстрактылы класс экземплярын құру үшін сіз оны мұрагерлеп, оның орындалуын аяқтауыңыз керек.

```
abstract class Polygon
{
    public double Length { get; protected set; }
    public double Width { get; protected set; }

    abstract public double GetArea();
}

class Rectangle : Polygon
{
    public Rectangle(double length, double width)
    {
        Length = length;
        Width = width;
    }

    public override double GetArea()
    {
        return Width * Length;
    }
}
```

Жабық кластар

- Жабық кластар толық функционалдылықты қамтамасыз етеді, бірақ оны негізгі класс ретінде пайдалануға болмайды.
- Сіздің орындалуыңыз (реализацияңыз) аяқталған кезде және сіз класс немесе оның мүшелері мұрагерленгенін қаламасаңыз, sealed кілттік сөзін қолданыңыз.

```
sealed class Rectangle : Polygon
{
   // a sealed class implementation goes here
}

public class Sample : Polygon
{
    // example of a sealed method
    sealed public override string GetName()
    {
        return "MyPolygon";
    }
}
```

Объектттен мұрагерлеу

- Object класы - .NET Framework-тағы барлық кластардың негізгі базасы.
- .NET Framework барлық кластары тікелей немесе жанама түрде Object класынан мұрагерленеді.

```
class Polygon
{
   public double Length { get; protected set; }
   public double Width { get; protected set; }
}

// the above class is equivalent to the following

class Polygon : Object
{
    public double Length { get; protected set; }
    public double Width { get; protected set; }
}
```

Casting

- C # тілінде объектті оның кез-келген негізгі типіне келтіре аласыз.
- .NET Framework барлық кластары тікелей немесе жанама түрде Object класынан мұрагерленеді.
- Туынды кластардың объектісін негізгі класс объектісіне тағайындау арнайы синтаксисті қажет етпейді:

```
Object o = new Rectangle(10, 20);
```

- Негізгі класс объектісін туынды класс объектісіне тағайындау нақты берілуі керек:

```
Rectangle r = (Rectangle) o;
```

- Орындалу уақытында, егер o мәні Rectangle класына сәйкес келмесе, орындалу ортасы System.InvalidCastException қателігін береді.

is операторы

- Программа орындалу барысында InvalidCastException секілді қателерді болдырмау үшін is операторын қолдануға болады

```
if (o is Rectangle)
{
    Rectangle r = (Rectangle)o;
}
```

- Мұнда жұмыс ортасы o объектісінің мәнін тексереді. cast операторы егер o Rectangle объектісін қамтитын жағдайда ғана орындалады.

as операторы

- as операторы cast операциясына ұқсайды, бірақ егер типтерді түрлендіру мүмкін болмаса, ерекшелік алынып тасталудың (exception) орнына null қайтарылады.

```
Rectangle r = o as Rectangle;
if (r != null)
{
    // do something
}
```

- Жұмыс уақытында, егер O айнымалы мәнін тіктөртбұрышқа айналдыру мүмкін болмаса, n айнымалысына null мәні тіркеледі. Ерекшеліктер болмайды.

Полиморфизм

- Полиморфизм дегеніміз - туынды кластардың базалық класстармен ортақ функционалдылықты бөлісу қабілеті, бірақ бәрібір өзінің ерекше мінез-құлқын анықтайды.

- Полиморфизм сізге туынды класс объектілерін жұмыс уақытында негізгі класстың объектілері ретінде өңдеуге мүмкіндік береді. Егер әдіс орындалу уақытында шақырылса, оның нақты түрі анықталады, ал сәйкес әдіс туынды кластан шақырылады.

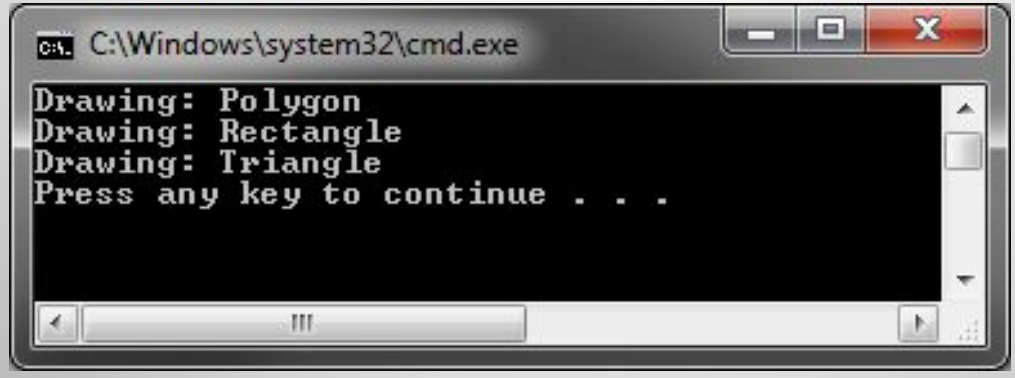
Полиморфизм - Мысал

- Келесі кластар жиынтығын қарастырыңыз:

```
class Polygon
{
    public virtual void Draw()
    {
        Console.WriteLine("Drawing: Polygon");
    }
}

class Rectangle : Polygon
{
    public override void Draw()
    {
        Console.WriteLine("Drawing: Rectangle");
    }
}

class Triangle : Polygon
{
    public override void Draw()
    {
        Console.WriteLine("Drawing: Triangle");
    }
}
```

Полиморфизм - Мысал

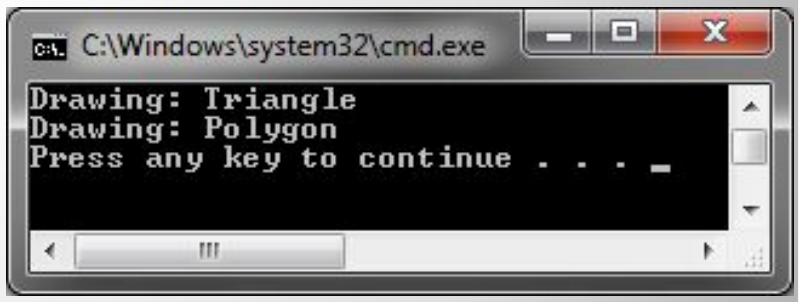
```
static void Main(string[] args)
{
    List<Polygon> polygons = new List<Polygon>();
    polygons.Add(new Polygon());
    polygons.Add(new Rectangle());
    polygons.Add(new Triangle());

    foreach (Polygon p in polygons)
    {
        p.Draw();
    }
}
```



Қайта анықтау және жаңа кілттік сөздер

- override кілттік сөзі туынды кластағы негізгі класс мүшелерін алмастырады.
- new кілттік сөзі туынды класта бірдей атпен жаңа мүше жасайды және негізгі кластың орындалуын жасырады.

```
class Triangle : Polygon
{
    public new void Draw()
    {
        Console.WriteLine("Drawing: Triangle");
    }
}


class Program
{
    static void Main(string[] args)
    {
        Triangle t = new Triangle();
        t.Draw();

        Polygon p = t;
        p.Draw();
    }
}
```



Интерфейстер

- Интерфейстер объектілерді бір-бірімен жүзеге асырудың егжей-тегжейін білместен әрекеттесе алатын келісім-шарттар құру үшін қолданылады.
- Интерфейс анықтамасы кез-келген деректер өрісінен немесе әдіс денелері сияқтыжүзеге асыру туралы мәліметтерден тұра алмайды..
- System аттар кеңістігінде анықталған жалпы интерфейс - бұл ICompara интерфейс келесідей анықталға

```
interface IComparable
{
    int CompareTo(object obj);
}
```

- IComparable қолданатын әр класс CompareTo әдісінің ішінде өзінің салыстыру логикасын ұсына алады.

Қайталау

- Объектілер
- ✔ Кластар, әдістер, қасиеттер, делегаттар, оқиғалар
- ✔ Атаулар кеңістігі
- ✔ Статикалық мүшелер
- Құндылықтар мен ұсыныстар
- Инкапсуляция
- ✔ Қол жетімділік модификаторлары
- Мұрагерлік
- ✔ Абстракт және жабық кластар
- ✔ Casting, is және as операторлары
- Полиморфизм
- ✔ Қайта анықтау және жаңа кілттік сөздер
- Интерфейстер