

Interface (Arayuz)

- Bir Interface, başka sınıflar için bir rehberdir.
- Interface'ler içinde kod yazılmaz dediğimiz gibi sadece yön verir class'a ne yapacağını söyler.
- Bir class birden çok interface'i uygulayabilir ama class bir interface'yi kendisine uyguluyor ise kesinlikle tüm metotlarını ezmek zorundadır. Fakat bu ezme işleminde override kullanılmaz.
- Interface'leri tanımlarken default (geçerli) olarak public atanır protected veya private olarak belirtemeyiz.
- Interface'lerin yapıcı metodu olmaz aynı şekilde değişkenleri tanımlayamayız.
- Arayüz elemanlarını static olarak tanımlayamayız.

Interface

```
public interface IArayuz
{
    void EkranaYaz();

    int Yas
    {
        get;
        set;
    }

    string Isim
    {
        get;
        set;
    }
}
```

property var

- I harfi Interface'e başlarken genelde kullanılır ve tavsiye edilir. .Net'in içindeki Interface'lerde bu standart olarak kabul edilmiştir.

```
public class Kisiler : IArayuz
{
    private int y;
    private string i;
    public Kisiler()
    {
        y = 18;
        i = "Yok";
    }

    public Kisiler(string ad, int yas)
    {
        Yas = yas;
        Isim = ad;
    }

    public int Yas
    {
        get{ return y; }
        set{
            if (value < 0) y = 0;
            else y = value;
        }
    }

    public string Isim
    {
        get{ return i; }

        set{ i = value; }
    }

    public void EkranaYaz()
    {
        Console.WriteLine("Adım:" + i);
        Console.WriteLine("Yaşım:" + y);
    }
}
```

Interface'in tüm üyeleri kullanılır. Ama sıralaması önemli değildir. Ayrıca türetilen sınıf içine interface'den farklı üyelere yazılabilir.

set veya get'ten birini yazmazsan çalışmaz

insert olduğu için çağırırken override dememize gerek kalmaz direk çağırabiliyoruz

Birden fazla interface

- Bir sınıf birden fazla interface içerebilir ve interface'ler içindeki tüm üyeleri kullanmak zorundadır.

```
//İngiliz ölçü birimleri için
interface IEnglishDimensions
{
    float Length();
    float Width();
}

//Diğer ölçü birimleri için
interface IMetricDimensions
{
    float Length();
    float Width();
}
```

```
class Box : IEnglishDimensions, IMetricDimensions
{
    float lengthInches;
    float widthInches;
    public Box(float length, float width)
    {
        lengthInches = length;
        widthInches = width;
    }
    float IEnglishDimensions.Length()
    {
        return lengthInches;
    }
    float IEnglishDimensions.Width()
    {
        return widthInches;
    }
    float IMetricDimensions.Length()
    {
        return lengthInches * 2.54f;
    }
    float IMetricDimensions.Width()
    {
        return widthInches * 2.54f;
    }
}
```

iki tane interface var

bunun mantığı şu ingilizler ölçüğü inç şeklinde aldıklarından uzunluğu 2,54 ile çarpıyormuş. Burda da hem sade uzunluk alma hemde inç şeklinde alma imkanı var

Birden fazla interface

```
class Box : IEnglishDimensions, IMetricDimensions
{
    float lengthInches;
    float widthInches;
    public Box(float length, float width)
    {
        lengthInches = length;
        widthInches = width;
    }
    float IEnglishDimensions.Length()
    {
        return lengthInches;
    }
    float IEnglishDimensions.Width()
    {
        return widthInches;
    }
    float IMetricDimensions.Length()
    {
        return lengthInches * 2.54f;
    }
    float IMetricDimensions.Width()
    {
        return widthInches * 2.54f;
    }
}
```

```
//İngiliz ölçü birimleri için
interface IEnglishDimensions
{
    float Length();
    float Width();
}
//Diğer ölçü birimleri için
interface IMetricDimensions
{
    float Length();
    float Width();
}
```

```
static void Main()
{
    Box box1 = new Box(30.0f, 20.0f);
    IEnglishDimensions eDimensions = (IEnglishDimensions)box1;
    IMetricDimensions mDimensions = (IMetricDimensions)box1;

    System.Console.WriteLine("Length(in): {0}", eDimensions.Length());
    System.Console.WriteLine("Width (in): {0}", eDimensions.Width());
    System.Console.WriteLine("Length(cm): {0}", mDimensions.Length());
    System.Console.WriteLine("Width (cm): {0}", mDimensions.Width());
}
```

bilinçli dönüşüm yapmak zorunda

Interface türüne göre sınıfa bilinçli dönüşüm yapılır.

Bilinçli dönüşüm sonuçları dönüşüm türünden değişkenlere aktarılabilir.

Multiple Inheritance (Çoklu Kalıtım)

- Bir sınıf direkt iki sınıfı kalıtım olarak kullanmaz. Dolaylı yoldan kullanabilir.
- Diğer taraftan interface örneklerinde olduğu gibi bu durum interface için geçerli değildir.

```
class Baba
{
}
class Anne: Baba
{
}
class Cocuk: Anne
{
}
```

```
class Baba
{
}
class Anne
{
}
class Cocuk: Anne, Baba
{
}
```

Error List

1 Error | 0 Warnings | 0 Messages

Description

1 Class 'Cocuk' cannot have multiple base classes: 'Anne' and 'Baba'

```
interface Baba
{
}
interface Anne
{
}
class Cocuk: Anne, Baba
{
}
```

.Net Interface'leri

- .Net içinde bir çok interface türü vardır. Örneğin
 - IEnumerable
 - IList
 - IDictionary
 - IComparable

.Net Interface - IEnumerable

- IEnumerable
 - Tüm array'leri ve List'leri IEnumerable arayüzünden türetilmişlerdir. Bu sebepten dolayı IEnumerable üzerinden bu sınıf türlerine ulaşmak mümkündür.

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        int[] values = { 1, 2, 3 };
        List<int> values2 = new List<int>() { 1, 2, 3 };

        Display(values);
        Display(values2);
    }

    static void Display(IEnumerable<int> values)
    {
        foreach (int value in values)
        {
            Console.WriteLine(value);
        }
    }
}
```

Yandaki örnek IEnumerable arayüzü sayesinde sadece tek fonksiyon kullanarak hem int[] hem de List<int> veri türlerini ekranda gösterir.

Bu örnekte int[] ve List veri türlerinin IEnumerable özelliğini içermesini kullandık. IComparable örneğinde kendi sınıfımıza türetme yapacağız.

.Net Interface - IComparable

- Karşılaştırma işlemi ile dizi üzerinde sıralama yapmak için kullanılır. Bu arayüz tanımlandığında CompareTo metodunu kullanmak şarttır.

```
using System;
using System.Collections.Generic;

class Calisan : IComparable<Calisan>
{
    public int Maas { get; set; }
    public string Ad { get; set; }

    public int CompareTo(Calisan diger_Calisan)
    {
        //Maas eşit ise alfabetik sıralama [A'dan Z'ye]
        if (this.Maas == diger_Calisan.Maas)
        {
            return this.Ad.CompareTo(diger_Calisan.Ad);
        }
        // Geçerli durum if gerçekleşmezse çalışır. [Yüksekten düşüğe]
        return diger_Calisan.Maas.CompareTo(this.Maas);
    }

    //toString metodu direkt object içinde vardır ve virtual türündendir.
    //Override edilmesi şarttır
    public override string ToString()
    {
        return this.Maas.ToString() + "," + this.Ad;
    }
}
```

Arayüz

Bu metod şart kullanılmalı. Arayüzden geliyor.

Object'ten geliyor, override edilip toString durumunda ne değer döndüreceği söyleniyor.

.Net Interface - IComparable

```
using System;
using System.Collections.Generic;

class Calisan : IComparable<Calisan>
{
    public int Maas { get; set; }
    public string Ad { get; set; }

    public int CompareTo(Calisan diger_Calisan)
    {
        //Maas eşit ise alfabetik sıralama [A'dan Z'ye]
        if (this.Maas == diger_Calisan.Maas)
        {
            return this.Ad.CompareTo(diger_Calisan.Ad);
        }
        // Geçerli durum if gerçekleşmezse çalışır. [Yüksekten düşüğe]
        return diger_Calisan.Maas.CompareTo(this.Maas);
    }

    //toString metodu direkt object içinde vardır ve virtual türündendir.
    //Override edilmesi şarttır
    public override string ToString()
    {
        return this.Maas.ToString() + "," + this.Ad;
    }
}
```

```
class Program
{
    static void Main()
    {
        List<Calisan> list = new List<Calisan>();
        list.Add(new Calisan() { Ad = "Ali", Maas = 10000 });
        list.Add(new Calisan() { Ad = "Veli", Maas = 10000 });
        list.Add(new Calisan() { Ad = "Ayşe", Maas = 10000 });
        list.Add(new Calisan() { Ad = "Fatma", Maas = 50000 });
        list.Add(new Calisan() { Ad = "Metin", Maas = 8000 });

        //IComparable.CompareTo() Fonksiyonuna göre sıralar
        list.Sort();

        //Calisan.ToString kullanır
        foreach (var element in list)
        {
            Console.WriteLine(element);
        }
    }
}
```