

## NYP 4. ÖDEV

1- Bina, Araba ve Bisiklet sınıfları arasında ilişki olmayan üç sınıf oluşturun. Bir GetCarbonFootprint metodu ile bir ICarbonFootprint arayüzü yazın. Sınıflarınızın her biri bu arayüzü uygulasin, böylece GetCarbonFootprint yöntemi o sınıf için uygun bir karbon ayak izini hesaplasın (Karbon ayak izi = Yakıt Tüketimi x Emisyon Faktörü). Üç sınıfın her birinden nesneler oluşturan, bu nesnelerin referanslarını List<ICarbonFootprint>'e yerleştiren ve ardından her nesnenin GetCarbonFootprint yöntemini çok biçimli olarak çağırarak Listeyi yineleyen bir uygulama yazın.

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

namespace odev
{
    class Program
    {
        interface ICarbonFootprint
        {
            double GetCarbonFootprint();
        }
        class Bina:ICarbonFootprint
        {
            private double YakitTuketimi;
            private double EmisyonFaktoru;
            public Bina(double yakitTuketimi, double emisyonFaktoru)
            {
                YakitTuketimi = yakitTuketimi;
                EmisyonFaktoru = emisyonFaktoru;
            }

            public double GetCarbonFootprint()
            {
                return YakitTuketimi * EmisyonFaktoru;
            }
        }
        class Araba:ICarbonFootprint
        {
            private double YakitTuketimi;
            private double EmisyonFaktoru;

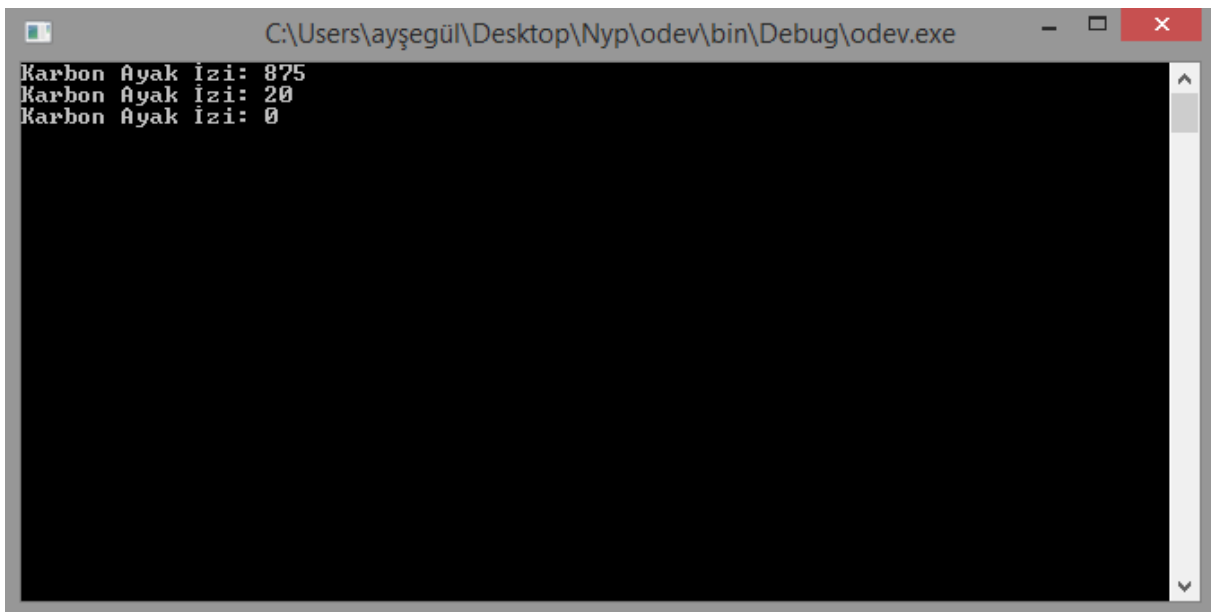
            public Araba(double yakitTuketimi, double emisyonFaktoru)
            {
                YakitTuketimi = yakitTuketimi;
                EmisyonFaktoru = emisyonFaktoru;
            }

            public double GetCarbonFootprint()
            {
```

```

        return YakitTuketimi * EmisyonFaktoru;
    }
}
class Bisiklet:ICarbonFootprint
{
    public double GetCarbonFootprint()
    {
        return 0.0;
    }
}
static void Main(string[] args)
{
    // Bina, Araba ve Bisiklet nesnelerini içeren bir Liste
    oluşturma
    List<ICarbonFootprint> karbonayakizi = new
    List<ICarbonFootprint>
    {
        new Bina(1750,0.5),
        new Araba(100,0.2),
        new Bisiklet()
    };
    // Listeyi gezerek her nesnenin karbon ayak izini yazdırma
    foreach (var ayakizi in karbonayakizi)
    {
        Console.WriteLine($"Karbon Ayak İzi:
{ayakizi.GetCarbonFootprint()}");
    }
    Console.ReadLine();
}
}
}

```



```

C:\Users\aysegül\Desktop\Nyp\odev\bin\Debug\odev.exe
Karbon Ayak İzi: 875
Karbon Ayak İzi: 20
Karbon Ayak İzi: 0

```

2- Bir hayvanat bahçesi uygulaması için farklı hayvan türlerini temsil eden sınıflar oluşturmanız isteniyor. Bu senaryoda, Kalıtım (Inheritance), Virtual, Abstract, Interface Polymorphism, Sealed kavramlarını kullanarak bir sınıf hiyerarşisi oluşturmanız bekleniyor.

Hayvan (Animal) Sınıfı:

Ad, yaş ve ses çıkarabilme gibi temel özelliklere sahip bir Hayvan sınıfı oluşturun. MakeSound adında bir metod tanımlayın.

Kuş (Bird) Sınıfı:

Hayvan sınıfından türeyen bir Kuş sınıfı oluşturun. Kuşların uçabilme özelliğini temsil eden bir property ekleyin. MakeSound metodunu override edin.

Kaplumbağa (Turtle) Sınıfı:

Hayvan sınıfından türeyen bir Kaplumbağa sınıfı oluşturun. Kaplumbağaların yüzebilme özelliğini temsil eden bir property ekleyin. MakeSound metodunu override edin.

Uçabilen (IFlyable) Arayüzü:

Uçabilen hayvanları temsil eden bir arayüz oluşturun. İçinde Fly adında bir metod bulunmalıdır.

Yüzme Yeteneği (ISwimmable) Arayüzü:

Yüzebilen hayvanları temsil eden bir arayüz oluşturun. İçinde Swim adında bir metod bulunmalıdır.

Sealed Sınıf:

Kanatlı Şahin adında bir sınıf oluşturun. Bu sınıf, Kuş sınıfından türemeli ve uçuş yeteneği mühürlenmelidir. Bu yapıyı kullanarak bir kuş, bir kaplumbağa ve bir kanatlı şahin oluşturun, özelliklerini kullanarak bilgilerini ekrana yazdırın.

```
using System;
namespace odev
{
    class Program
    {
        public abstract class Animal
        {
```

```

public string Ad { get; set; }
public int Yas { get; set; }
public Animal(string ad, int yas)
{
    Ad = ad;
    Yas = yas;
}
public virtual void MakeSound()
{
    Console.WriteLine("Hayvan sesi çıkarıyor.");
}
}
public class Bird : Animal, IFlyable
{
    public bool UcabilmeOzelligi { get; set; }
    public Bird(string ad, int yas, bool UcmaOzelligi) : base(ad,
yas)
    {
        UcabilmeOzelligi = UcmaOzelligi;
    }
    public override void MakeSound()
    {
        Console.WriteLine("Cik Cik");
    }

    public void Fly()
    {
        if (UcabilmeOzelligi == true)
            Console.WriteLine(Ad + " " + "uçuyor.");
        else
            Console.WriteLine(Ad + " " + "uçamaz.");
    }
}

public class Turtle : Animal, ISwimmable
{
    public bool YuzebilmeOzelligi { get; set; }
    public Turtle(string ad, int yas, bool YuzmeOzelligi) :
base(ad, yas)
    {
        YuzebilmeOzelligi = YuzmeOzelligi;
    }
    public override void MakeSound()
    {
        Console.WriteLine("Kvak Kvak");
    }

    public void Swim()
    {
        if (YuzebilmeOzelligi == true)
            Console.WriteLine(Ad + " " + "yüzüyor.");
        else
            Console.WriteLine(Ad + " " + "yüzemez.");
    }
}

```

```

    }
    public interface IFlyable
    {
        void Fly();
    }
    public interface ISwimmable
    {
        void Swim();
    }
    public sealed class KanatliSahin : Bird
    {
        public KanatliSahin(string ad, int yas) : base(ad, yas, true)
        {
        }
    }
    static void Main(string[] args)
    {
        // Kuş nesnesi
        Bird bird = new Bird("Penguen", 3, false);
        bird.MakeSound();
        bird.Fly();
        Console.WriteLine();
        // Kaplumbağa nesnesi
        Turtle turtle = new Turtle("Caretta caretta", 4, true);
        turtle.MakeSound();
        turtle.Swim();
        Console.WriteLine();
        // Kanatlı Şahin nesnesi
        KanatliSahin sahin = new KanatliSahin("Şahin", 2);
        sahin.MakeSound();
        sahin.Fly();
        Console.ReadLine();
    }
}

```

```

C:\Users\ayşegül\Desktop\Nyp\odev\bin\Debug\odev.exe
Cik Cik
Penguen uçamaz.

Kvak Kvak
Caretta caretta yüzüyor.

Cik Cik
Şahin uçuyor.

```

### 3- Enumeration ve Class Kullanımı

Bir okul yönetim sistemi yazıyorsunuz. Öğrencilerin sınıf seviyelerini temsil eden bir numaralandırma ve bu öğrencilerin bilgilerini saklayan bir sınıf oluşturmanızı istiyoruz.

Bir Enumeration olan SinifSeviyesi oluşturun. Bu, İlkokul, Ortaokul ve Lise seviyelerini içermelidir.

Bir Ogrenci sınıfı oluşturun. Bu sınıf, öğrencinin adını, soyadını, sınıf seviyesini ve öğrenci numarasını içermelidir.

Ogrenci sınıfında bir metot ekleyin: BilgileriGoster(). Bu metot, öğrenci bilgilerini ekrana yazdırmalıdır.

Main metodu içinde, birkaç farklı sınıf seviyesinde öğrenciyi temsil eden örnek Ogrenci nesneleri oluşturun ve bilgilerini ekrana yazdırın.

```
using System;

namespace odev
{
    class Program
    {
        public enum SinifSeviyesi
        {
            Ilkokul,
            Ortaokul,
            Lise
        }
        public class Ogrenci
        {
            public string OgrenciAdi { get; set; }
            public string OgrenciSoyadi { get; set; }
            public SinifSeviyesi SinifSeviyesi { get; set; }
            public int OgrenciNo { get; set; }
            public Ogrenci(string ad, string soyad, SinifSeviyesi
sinifSeviye, int ogrenciNo)
            {
                OgrenciAdi = ad;
                OgrenciSoyadi = soyad;
                SinifSeviyesi = sınıfSeviye;
                OgrenciNo = ogrenciNo;
            }
            public void BilgileriGoster()
            {
                Console.WriteLine($"Öğrencinin Adı Soyadı: {OgrenciAdi}
{OgrenciSoyadi}");
                Console.WriteLine($"Sınıf Seviyesi: {SinifSeviyesi}");
                Console.WriteLine($"Öğrenci No: {OgrenciNo}");
            }
        }
    }
}
```

```

        Console.WriteLine();
    }
}
static void Main(string[] args)
{
    Ogrenci ogrenci1 = new Ogrenci("Ahmet", "Yılmaz",
SinifSeviyesi.Ilkokul, 9065);
    Ogrenci ogrenci2 = new Ogrenci("Ilgaz", "Kaya",
SinifSeviyesi.Ortaokul, 9056);
    Ogrenci ogrenci3 = new Ogrenci("Ayşegül", "Saygı",
SinifSeviyesi.Lise, 9059);
    ogrenci1.BilgileriGoster();
    ogrenci2.BilgileriGoster();
    ogrenci3.BilgileriGoster();
    Console.ReadLine();
}
}
}

```

```

C:\Users\ayşegül\Desktop\Nyp\odev\bin\Debug\odev.exe
Öğrencinin Adı Soyadı: Ahmet Yılmaz
Sınıf Seviyesi: Ilkokul
Öğrenci No: 9065

Öğrencinin Adı Soyadı: Ilgaz Kaya
Sınıf Seviyesi: Ortaokul
Öğrenci No: 9056

Öğrencinin Adı Soyadı: Ayşegül Saygı
Sınıf Seviyesi: Lise
Öğrenci No: 9059

```

#### 4- Struct ve Enumeration Kullanımı

Bir oyun karakteri tasarlamak üzeresiniz. Bu karakterin pozisyonunu (x, y), sağlık durumunu ve karakterin türünü (örneğin, savaşçı, büyücü) içeren bir OyunKarakter struct'ı oluşturun.

Bir Enumeration olan KarakterTuru oluşturun. Bu, Savaşçı, Büyücü ve Avcı türlerini içermelidir.

Bir OyunKarakter struct'ı oluşturun. Bu, karakterin pozisyonunu temsil eden iki integer (X ve Y), sağlık durumunu temsil eden bir integer (Saglik) ve karakterin türünü temsil eden bir KarakterTuru özelliği içermelidir.

OyunKarakteri struct'ında bir metot ekleyin: DurumuGoster(). Bu metot, karakterin türünü ve sağlık durumunu ekrana yazdırmalıdır.

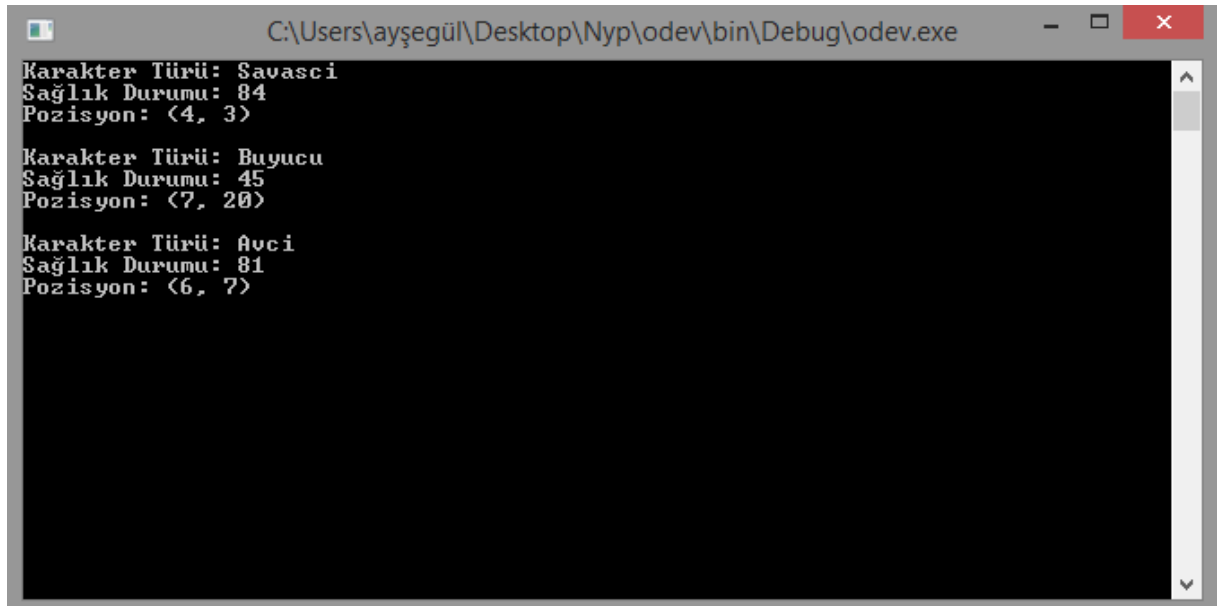
Main metodu içinde, farklı türlerdeki oyun karakterlerini temsil eden örnek OyunKarakteri struct'larını oluşturun ve durumlarını ekrana yazdırın.

```
using System;

namespace odev
{
    class Program
    {
        public enum KarakterTuru
        {
            Savasci,
            Buyucu,
            Avci
        }
        public struct OyunKarakteri
        {
            public int X { get; set; }
            public int Y { get; set; }
            public int SaglikDurumu { get; set; }
            public KarakterTuru KarakterTuru { get; set; }
            public OyunKarakteri(int x, int y, int saglik, KarakterTuru
karakterturu)
            {
                X = x;
                Y = y;
                SaglikDurumu = saglik;
                KarakterTuru = karakterturu;
            }
            public void DurumuGoster()
            {
                Console.WriteLine($"Karakter Türü: {KarakterTuru}");
                Console.WriteLine($"Sağlık Durumu: {SaglikDurumu}");
                Console.WriteLine($"Pozisyon: ({X}, {Y})");
                Console.WriteLine();
            }
        }
        static void Main(string[] args)
        {
            OyunKarakteri karakter1 = new OyunKarakteri(4, 3, 84,
KarakterTuru.Savasci);
            OyunKarakteri karakter2 = new OyunKarakteri(7, 20, 45,
KarakterTuru.Buyucu);
            OyunKarakteri karakter3 = new OyunKarakteri(6, 7, 81,
KarakterTuru.Avci);
            karakter1.DurumuGoster();
            karakter2.DurumuGoster();
            karakter3.DurumuGoster();
            Console.ReadLine();
        }
    }
}
```



```
}  
}  
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\ayşegül\Desktop\Nyp\odev\bin\Debug\odev.exe". The window contains three lines of text, each representing a character's data. The first line is "Karakter Türü: Savasci", "Sağlık Durumu: 84", and "Pozisyon: (4, 3)". The second line is "Karakter Türü: Büyücü", "Sağlık Durumu: 45", and "Pozisyon: (7, 20)". The third line is "Karakter Türü: Avcı", "Sağlık Durumu: 81", and "Pozisyon: (6, 7)". The text is displayed in a monospaced font on a black background.

```
C:\Users\ayşegül\Desktop\Nyp\odev\bin\Debug\odev.exe  
Karakter Türü: Savasci  
Sağlık Durumu: 84  
Pozisyon: (4, 3)  
  
Karakter Türü: Büyücü  
Sağlık Durumu: 45  
Pozisyon: (7, 20)  
  
Karakter Türü: Avcı  
Sağlık Durumu: 81  
Pozisyon: (6, 7)
```