



YÖNTEMLER VE ALANLAR - III

Şimdi bir projede birçok trigonometri hesabı yapacağımızı düşünelim. Bu projede sinüs, kosinüs ve tanjant hesapları kullanacağız. System isim uzayı içerisindeki Math sınıfı bu hesaplamaları yapmaya yarayan Sin, Cos ve Tan isimli yöntemlere sahiptir. Ama bu yöntemler derece cinsinden değil de radyan cinsinden değerlerle çalışır. Bizim projemizde de derece cinsinden değerlerle çalışılacak olsun. Gerekli dönüşümleri yaparak yeni Sin, Cos ve Tan yöntemlerini yeniden yazalım.

2Pi radyan 360 derecedir. Dönüşüm işlemine bir örnek yazalım. dDerece değişkeni kullanıcıdan alınan derece cinsinden değeri tutuyor olsun. Bu açının sinüsünü hesaplayan ifade aşağıdaki gibi olacaktır.

```
dSonuc = Math.Sin(Math.PI * dDerece / 180);
```

Her hesaplama işleminde bu ifadeleri teker teker yazmak istemeyiz. Şimdi bu dönüşümleri içerisinde izole eden Sin, Cos ve Tan yöntemlerini yazmalıyız.

Şimdi Trig1 isimli yeni bir sınıf yazalım ve içerisinde bu yöntemler bulunsun. Yeri gelmişken yaygın bir teknikten de bahsedelim. Bu sınıfı normalde başka bir projenin parçası olsun diye yazdığımız ve kendi başına kullanmayacağımız için bir Main yöntemine sahip olmayacaktır, ama test amaçlı olarak bir Main yöntemi ilave edelim. Daha sonra gerekli hata ayıklama işlemleri ve testleri yaptıktan sonra bu yöntemi silebiliriz.

Trig1 isimli projenin kodları aşağıdaki gibidir.

```
using System;

class Trig1
{
    static void Main()
    {
        Console.WriteLine("45 derecenin sinüsü: " + Trig1.Sin(45));
        Console.WriteLine("45 derecenin kosinüsü: " + Trig1.Cos(45));
        Console.WriteLine("45 derecenin tanjantı: " + Trig1.Tan(45));
        Console.ReadLine();
    }

    public static double Sin(double dAci)
    {
        return Math.Sin(Math.PI * dAci / 180);
    }

    public static double Cos(double dAci)
    {
        return Math.Cos(Math.PI * dAci / 180);
    }
}
```

```
public static double Tan(double dAci)
{
    return Math.Tan(Math.PI * dAci / 180);
}
```

Bu yöntemler daha henüz başka bir programdan çağrılmadığı halde, başlarına erişim niteleyicisi olarak public ekledik. Bu sınıfı daha büyük bir projede kullanacağımız zaman için hazır hale geldi. Main yöntemi içerisinde bu yöntemleri kullanırken, aynı sınıf içerisinde olmalarına rağmen sınıf ismi ile birlikte yazdık. İleride bu yöntemi de hiçbir şeyi değiştirmeden proje içerisindeki başka bir sınıfın içerisine taşıyabiliriz.

Dikkat ederseniz her üç yöntem de aynı dönüşüm ifadelerini içeriyor. Her birinde tekrar tekrar yazdık. İyi bir programcı tekrarlayan kodları bir yere toplar ve tekrar yazmadan kullanır.

Çözümlerden bir tanesi ayrı bir yöntem yazmaktır.

```
static double DerecedenRadyana(double dDerece)
{
    return Math.PI * dDerece / 180;
}
```

Bu durumda programın tamamı şu şekilde olacaktır.

```
using System;

class Trig2
{
    static void Main(string[] args)
    {
        Console.WriteLine("45 derecenin sinüsü: " + Trig2.Sin(45));
        Console.WriteLine("45 derecenin kosinüsü: " + Trig2.Cos(45));
        Console.WriteLine("45 derecenin tanjantı: " + Trig2.Tan(45));
        Console.ReadLine();
    }

    static double DerecedenRadyana(double dDerece)
    {
        return Math.PI * dDerece / 180;
    }

    public static double Sin(double dAci)
    {
        return Math.Sin(DerecedenRadyana(dAci));
    }

    public static double Cos(double dAci)
    {
        return Math.Cos(DerecedenRadyana(dAci));
    }

    public static double Tan(double dAci)
    {
```

```
        return Math.Tan(DerecedenRadyana(dAci));
    }
}
```

DerecedenRadyana yöntemine dikkat etti iseniz, başında public niteleyicisi yok. Yazmadığımız için private kullanılmış gibi oldu, çünkü private varsaylan değerdir. Başka sınıflar içerisinde ulaşılmasını istediğimiz yöntemleri public, sadece sınıf içerisinde kullanılacak olan yöntemleri private yaptık.

Ama ayrı bir yöntem kullanmak ve onu çağırmak programımıza fazladan yük getireceği için çok anlamlı olmadı. Bunun yerine bir dönüşüm faktörü yazıp kullanmak daha anlamlı olur.

```
double dDonusturucu = Math.PI / 180;
```

Sin yönteminin kullanımı da aşağıdaki hale gelecektir.

```
return Math.Sin(dDonusturucu * dAci);
```

Dönüşüm faktörünün nerede tanımlanacağına karar verelim. Sin yönteminin içinde tanımlarsak sadece Sin yöntemi içinde yerel olarak kullanılır diğer yöntemlerden ulaşamaz. Diğerleri içerisinde de ayrı ayrı tanımlamak gerekiyor. Aynı ifadeleri tekrar tekrar yazmak ise bizim amacımıza terstir.

Çözüm, dDonusturucu faktörünü **alan** olarak tanımlamaktır. Farkında olmasak da alan kavramı ile daha önce karşılaştık. Math sınıfı, PI ve E olmak üzere iki tane alana sahiptir. PI ve E, önemli sabit değerler olan pi ve e sayılarının değerlerini tutan özel değişkenlerdir. Bütün yöntemlerin dışında ama sınıfın içerisinde bir yerde alanlar tanımlanır. Daha sonra, tanımladığımız alan bütün yöntemler tarafından erişilebilen genel bir değişken olur.

Alan içeren bir sınıf aşağıdaki gibi başlayabilir.

```
class Trig3
{
    static double dDonusturucu = Math.PI / 180;
```

Genellikle alan tanımlamaları sınıfın en üst kısmında olur ama tabi ki bu da bir zorunluluk değildir.

Diğer yöntemler gibi alan da static anahtar kelimesi ile tanımlandı, yöntemin kopyası ile değil de sadece kendisi ile çalışabilir. Bu meseleyi de daha sonra bir yazıda anlatacağız.

public niteleyicisini kullanmadık, çünkü sadece bu sınıf içerisinde ulaşılabilir olmasını istiyoruz. Şu hali ile diğer sınıflardan ulaşamaz olsa bile alanımız sınıf içerisindeki yöntemler tarafından ulaşılabilir ve değiştirilebilir. static kelimesini const ile değiştirerek alanımızı sadece okunabilir hale getirebiliriz. Herhangi bir alan const olarak tanımlanmışsa aynı zamanda da kapalı olarak static olarak tanımlanmış demektir.

```
class Trig3
{
    const double dDonusturucu = Math.PI / 180;
```

Şimdi çık sık karşılaşılan bir soruna da çözüm bulalım. Math sınıfındaki public tanımlanmış PI alanını Pi şeklinde mi PI şeklinde mi yazacağımızı unutabiliriz. Trig3 sınıfımızın içerisine hem PI hem de Pi şeklinde iki tane public alan tanımlayalım ve istediğimiz zaman projenin her hangi bir yerinde kullanılabilir hale getirelim.

```
public const double PI = Math.PI;  
public const double Pi = PI;
```

Bu arada programda yapacağımız değişiklikler bittiğine göre son halini tekrar yazalım.

```
using System;  
  
class Trig3  
{  
    public const double PI = Math.PI;  
    public const double Pi = PI;  
    const double dDonusturucu = Math.PI / 180;  
  
    static void Main()  
    {  
        Console.WriteLine("45 derecenin sinüsü: " + Trig3.Sin(45));  
        Console.WriteLine("45 derecenin kosinüsü: " + Trig3.Cos(45));  
        Console.WriteLine("45 derecenin tanjantı: " + Trig3.Tan(45));  
        Console.ReadLine();  
    }  
  
    public static double Sin(double dAci)  
    {  
        return Math.Sin(dDonusturucu * dAci);  
    }  
  
    public static double Cos(double dAci)  
    {  
        return Math.Cos(dDonusturucu * dAci);  
    }  
  
    public static double Tan(double dAci)  
    {  
        return Math.Tan(dDonusturucu * dAci);  
    }  
}
```

Trig3 projesinin sınıf diyagramı aşağıdaki gibidir.

