



## EKRANDA YAZI YAZDIRMAK

Program yazmanın iki boyutundan bahsedebiliriz, bunlar ise programcı açısından program ve kullanıcı açısından programdır. Program geliştiriciler olarak başlangıçta her iki kişi de biziz. Rolümüz programcından kullanıcıya değişince ya da kullanıcıdan programcıya değişince, bakış açımız da değişecektir. Mesela, programımızın verdiği çıktı, kullanıcı açısından girdidir. Kullanıcının programa girdiği veriler program açısından girdi, kullanıcı açısından çıktıdır.

Burada bahsedilen girdi ve çıktı kavramları üzerinde durmak istiyorum. Programın dış dünyaya verdiği bilgiler, gösterdiği tepkiler çıktı olarak tanımlanırken, dış dünyadan aldığı bilgiler ise girdi olarak tanımlanır. Bir bilgisayar programı da aldığı girdileri yorumlar, saklar, bu bilgilere dayanarak dış dünyaya bir takım sonuçlar gönderir. Bu modele I/O modeli denir.

## I/O Modelleri

Windows ortamında çalışan bir program kullanıcıdan girdileri klavye, mouse, buton ya da kaydırma çubukları gibi çeşitli kontroller yardımı ile alır. Bir Windows programı çıktıyı kullanıcıya yazı veya grafik olarak gösterir. Bir fırını çalıştıran program ise farklı bir kullanıcı I/O modeline sahiptir. Burada girdiler fırını ön panelindeki butonlarla yapılır. Çıktılar ise fırının açılması, fırının kapanması, tepsinin döndürülmesi, ışığın yakılması şeklinde olacaktır. C ve C# gibi dillerde kullanıcı girdi ve çıktısı için özel bir model yoktur. Bu bir eksiklik gibi görünüyor fakat değil. Bunu şöyle söylemek daha doğru olacaktır: C#, kullanıcıyı herhangi bir girdi ve çıktı modelini kullanmaya zorlamaz. Siz yazdığınız programın nerede kullanılacağına göre girdi ve çıktı modelinizi kendiniz seçersiniz.

Bu bölümde konsol ekranı yardımıyla kullanıcıya çıktı verebilen bir program yazmaya çalışacağız. Bunu yapmadan önce bazı kavramları açıklamakta yarar var. Ekranda yazı göstermeye yarayan `System.Console.WriteLine()` ve `System.Console.Write` komutlarıdır. Bu komutlar neden üç parçadan oluşuyor? Bu parçaların adları nelerdir?

`System` bir isim uzayıdır. İsim uzayları C#'a C++ dilinden miras kalmıştır. İsim uzayları birbiri ile alakalı yöntemleri bir paket içinde toplamaya yarayan bir paketleme sistemidir. Bir isim uzayının içindeki bir yöntemin adı diğer bir isim uzayı içinde de bulunabilir. Bu da isim uzayının bir diğer işlevidir.

`System` isim uzayı; kullanıcı ile iletişim kurmaya yarayan I/O modeli olan `Console`, matematiksel işlemlerin bir araya toplandığı `Math` gibi temel sınıfları içinde barındırır. Burada sınıf terimi geçti. Bir önceki bölümde de anlattığımız gibi, C# programının temel birimi sınıftır. Bunu daha doğrusu şöyle söylemek lazım: Nesne yönelimli dillerin temel birimi sınıftır. Sınıfların özellikleri ve yöntemleri vardır. Bir arabayı örnek alalım. Rengi, modeli fiyatı gibi özellikleri vardır. Hareket etmek, korna çalmak, durmak gibi yöntemleri vardır. Daha sonra bu temel araba sınıfından bir takım farklı özellik ve yöntemlere sahip olan otobüs, kamyon, minibüs gibi sınıflar türetiriz. Bu sınıfların örnekleri olan nesnelere ise araba sınıfının niteliklerini taşırlar. Türetilmiş sınıftan oluşturulan nesne de türetilmiş sınıfın özelliklerini ve temel sınıfın özelliklerini taşır.

Yöntemler ise sınıfın dış dünya ile etkileşimini sağlayan eylemleridir. Mesela `Console` sınıfının `WriteLine` yöntemi ekranda bir satır yazı gösterir. `Write` yöntemi ise ekranda yazı gösterir ama alt satıra geçmez işini bitirince.

`WriteLine` yöntemini kullanan bir program bu yönteme bir argüman yollar:

```
1 | System.Console.WriteLine(ToplamMeyve);
```

Bazı programcılar parantez ile argümanı ayırmak için argümanın sağına ve soluna boşluk koymayı tercih ederler.

```
1 | System.Console.WriteLine( Toplam );
```

WriteLine yöntemine bir değişkeni argüman olarak geçebildiğimiz gibi, bir sayısal ifadeyi de geçebiliriz.

```
1 | System.Console.WriteLine( Elma + Armut );
```

Yukarıdaki iki satırı incelersek ilkinde bir işlemin sonucunu tutan değişkenin değerini yazdırıyoruz. İkincisinde ise işlemin kendisini direkt parantezlerin içine yazıyoruz. Sonucu hesaplayıp ekrana yazıyor.

Eğer amacımız sadece 2 ve 3 sayılarının toplamını ekrana yazdırmak ise

```
1 | System.Console.WriteLine( 2+3 );
```

yazabiliriz.

### using Direktifi

Eğer programda birden fazla System.Console.WriteLine yöntem çağrısı yapıyorsak System isimuzayını using direktifini sınıf tanımı yapmadan önce programımızın başına ekleriz. using direktifi ile tanımladığımız herhangi bir isimuzayına dahil sınıfların yöntemlerini çağırırken artık isimuzayı ismini yazmadan sadece sınıf.yöntem() şeklinde yöntemi yazarız. Bu bize zaman kazandırır. using direktifine direktif denmesinin sebebi programın çalışma zamanında hiçbir işlevinin olmamasındandır.

```
1 | Console.WriteLine(Toplam);
```

Şimdi artık çalışan ve ekrana yazı yazan bir program yazalım.

```
1 | using System;
2 | class BirProgram
3 | {
4 |     static void Main()
5 |     {
6 |         int Elma, Armut, Toplam;
7 |
8 |         Elma = 2;
9 |         Armut = 3;
10 |        Toplam = Elma + Armut;
11 |        Console.WriteLine( Toplam );
12 |    }
13 | }
```