



KONSOLDAN VERİ GİRMEK

Programlar çoğu zaman bizim girdiğimiz verilere göre işlemlerini yürütürler. Daha önce, görsel programlarda fare hareketleri ile, butonlara tıklamalarla veya yazı kutularına klavyeden bir şeyler yazarak veri girişi yaptık. Veri girişlerinin klavye veya fare ile yapıldığını bu nedenle biliyoruz zaten.

C# konsol uygulamalarında veri girişi daima klavyeden yazı yazma ile gerçekleşir. Genellikle konsol uygulamalarında program kullanıcıdan bir şeyler girmesini ister ve kullanıcının verdiği yanıtı göre kullanıcının ne demek istediğini anlamaya çalışır.

İşte bizim konsolda klavyeden giriş yapmamızı sağlayan Console sınıfının Read ve ReadLine yöntemleridir. Console.WriteLine ve Console.Write yöntemlerini kullanırken giriş argümanlarını yazmamız gerekiyordu. Ve bu yöntemlerden dönen bir argüman yoktu. Console.Read ve Console.ReadLine yöntemlerinde ise giriş argümanı yok ama bu yöntemlerden dönen dönüş argümanları mevcut. Biz bu iki yöntemi içeren bir program yazdığımızda ve bu yöntemleri çağırdığımızda kontrol biz Enter tuşuna basana kadar programa tekrar geçmeyecektir. Buradan da bu iki yöntemin ve dolayısı ile programın çalışmaya devam edebilmesi için bu yöntemlerin çağrıları yapıldıktan sonra Enter tuşuna basmamız gerektiğini anlıyoruz. Bu arada Console.ReadLine yönteminin bu iki yöntem içinde kullanımı kolay olan yöntem olduğunu söyleyebiliriz. Şimdi konsoldan string girişi yapan ilk programı yazalım.

```
string yazi;  
yazi = Console.ReadLine();
```

Yukarıda görüldüğü gibi 1. satırda string tipinde bir değişken tanımlandı ve ikinci satırda Console.ReadLine() yöntemi çağrılarak yazı adlı string değişkene bu yöntemden dönen (kullanıcının konsol ekranında yazdıkları) değerler atandı. Burada dikkati çeken bir noktada Console.ReadLine yönteminden dönen değer string bir değişken herhangi bir işleme tabi tutulmadan direkt olarak atanabildiğidir.

Burada iki satırda yaptığımız işlemi tek satırda da gerçekleştirebilmemizin mümkün olduğunu biliyoruz.

```
string yazi = Console.ReadLine();
```

Şimdi kullanıcıdan bir şeyler alan bir program yazalım.

```
using System;  
  
class Program  
{  
    static void Main()  
    {  
        Console.WriteLine("Bir şeyler yaz ve Enter tuşuna bas:");  
        string yazi = Console.ReadLine();  
        Console.WriteLine("Klavyeden yazdığınız yazı:" + yazi );  
    }  
}
```

Yukarıdaki programda kullanıcıdan bir şeyler yazmasını istedik ve bu yazdıklarını bir string değişkene `Console.ReadLine` yöntemi ile atayarak daha sonra bu değişkeni `Console.WriteLine` yöntemi ile konsolda görüntüledik. Burada dikkat edilirse ilk satırda `Write` yöntemi kullanıldı çünkü kullanıcının yazdıklarının bizim `Console.Write` yöntemi ile konsola yazdırdığımız yazıların yanına yazılmasını istedik. Kullanıcı bir şeyler yazıp Enter tuşuna bastığında `Console.ReadLine` yöntemi imlecin aşağı satıra düşmesini sağlıyor. Tam bu sırada `Console.ReadLine` yöntemi değer döndürme işlemini gerçekleştiriyor.

`Console.WriteLine` ve `Console.Write` yöntemleri bir integer değişkeni kapalı bir şekilde string tipine dönüştürerek konsola yazdırıyordu. Ama `Console.ReadLine` yönteminin böyle bir özelliği mevcut değil. Yani biz klavyeden bir integer okumak istediğimizde bunu `Int32.Parse` yöntemini kullanarak integer tipine çevirmemiz gerekiyor.

```
using System;

class Program
{
    static void Main()
    {
        string S;
        int I;
        Console.Write("Bir sayı yazınız ve Enter tuşuna basınız:");
        S = Console.ReadLine();
        I = Int32.Parse( S );
        Console.WriteLine("Girdiğiniz sayı:" + I );
    }
}
```

Yukarıdaki programda iki durumdan söz edebiliriz. Birincisi; `Console.ReadLine` yönteminden dönen değerlerin bir string olması ve string değişkene doğrudan atanabilmesi. İkincisi ise; bu string değişkenin `Int32.Parse` yöntemi ile integer bir değişkene aktarılması.

Ayrıca buradaki `S` string değişkenini `Console.ReadLine` ve `Int32.Parse` yöntemlerini birleştirerek ortadan kaldırabiliriz.

```
I = Int32.Parse( Console.ReadLine() );
```

Bu satırda ilk olarak `Console.ReadLine` yöntemi çağrılıyor ve konsoldan girilen yazıları okuyor. Daha sonra okuduğu değerleri `Int32.Parse` yöntemine aktarıyor.

Şimdi kullanıcıdan sayı almayı bildiğimize göre gerçek hayata yakın bir program yazabiliriz.

```
using System;

class Bilet
{
    static void Main()
    {
        int YetiskinUcreti = 10 , CocukUcreti =7 ;

        Console.Write("Yetiřkin bilet adedini giriniz:");
        int YetiskinBiletSayisi = Int32.Parse(Console.ReadLine());

        Console.Write("Cocuk bilet adedini giriniz:");
        int CocukBiletSayisi = Int32.Parse(Console.ReadLine());

        int ToplamMaliyet = YetiskinUcreti * YetiskinBiletSayisi +
CocukUcreti*CocukBiletSayisi;

        Console.WriteLine("Toplam bilet ücreti=" + ToplamMaliyet);
    }
}
```

Program kullanıcıdan yetiřkin ve çocuk bilet adedini alıyor. Daha sonra, önceden bildiđi bilet ücretleri ile adetleri çarpıp toplam ücreti bize söylüyor.

Konsoldan veri giriři yapmak konsol çıktıřı almaktan daha zordur. Veri giriřlerinde kontrol her zaman zordur. Kullanıcı konsoldan istediđimiz şartlara uymayan veriler de girebilir.

Mesela program ařađıdaki konsol çıktıřını verir.

1-100 arasında bir sayı giriniz:

Ve kullanıcı ařađıdaki ifadeyi girer.

10A

Burada yanlış yazılma ihtimali var. Ama başka programlarda olduđu gibi yanlış yazıları düzelten bir durum söz konusu deđil. Bazı programlar bu tür yanlışlar için dođrulama ve kontrol etme işlemleri gerçekleştirirler. Mesela yanlış yazılan "A" harfini ekrana yazmamak veya bir beep sesi ile uyarmak gibi. Ama maalesef. Net konsolu bu tür tekniklere izin vermez. Konsol kullanıcının satır boyunca yazmasını ve Enter tuřuna basmasını bekler ve kullanıcının yazdıđını işlemeye başlar. Eđer uygun giriř yapılmamıřsa sonuç elde edilemez. Ama kullanıcıdan tekrar giriř yapması istenebilir.

Kullanıcı giriřlerini kontrol eden ve oluřan istisnaları anlaşılır bir şekilde kullanıcıya gösteren kontrol yapıları daha sonraki yazılarda bahsedilecek.