

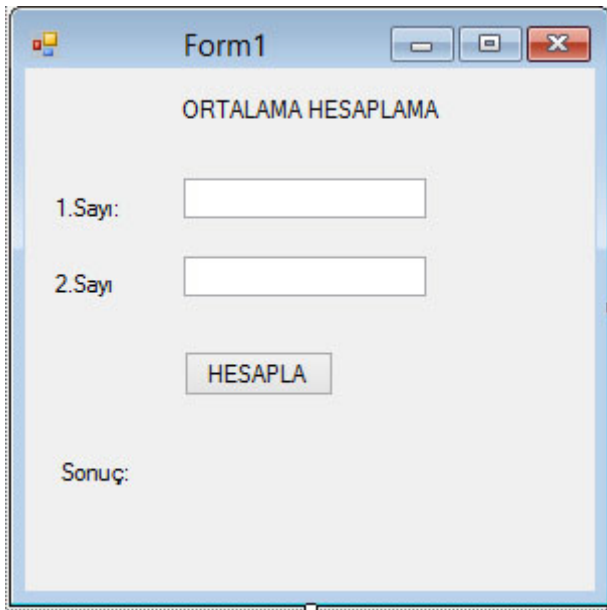
# C#

## TRY

## CATCH

C# dilinde programın çalışma anında meydana gelecek hataları engellemeye yarayan yöntemler bulunmaktadır. Try-catch-finally blokları C#'da **hataları yakalamak** için en fazla kullanılan kod bloğudur.

```
Try
{
    //Hata oluşabilecek kodlar
}
Catch
{
    //Hata oluştuğunda çalışacak kodlar
}
Finally
{
}
}
```



The image shows a screenshot of a Windows application window titled "Form1". Inside the window, the text "ORTALAMA HESAPLAMA" is displayed at the top. Below this, there are two input fields. The first is labeled "1.Sayı:" and the second is labeled "2.Sayı:". Below these input fields is a button labeled "HESAPLA". At the bottom of the form, there is a label "Sonuç:" followed by a space, indicating where the result of the calculation will be displayed.

```
private void btnHesapla_Click(object sender, EventArgs e)
{
    try
    {
        double sayi1 = Convert.ToDouble(txtSayi1.Text);
        double sayi2 = Convert.ToDouble(txtSayi2.Text);
        double ort = (sayi1 + sayi2) / 2;
        lblSonuc.Text = ort.ToString();
    }
    catch (Exception hataTuru)
    {
        lblSonuc.Text = "Hata meydana geldi." + hataTuru;
    }
    finally
    {
        txtSayi1.Text = "";
        txtSayi2.Text = "";
    }
}
```

**throw new DivideByZeroException();**

**System format EXCEPTION** // STRING İFADE GİRİLDİĞİNDE

0 bölündüğünde **INFINITY**

DOUBLE SINIRI AŞILDIĞINDA // **OVER FLOW**

## Birden Fazla Catch Bloğu Oluşturmak

Uygulamalarda bir çok sebepten dolayı hata oluşma riski bulunmaktadır. Şayet farklı hata türleri oluştuğunda farklı kod blokları işletilmek istenirse birden fazla catch bloğu kullanılabilir.

```
Try
{
    //hataya müsait kod bloğu
}

catch(OverflowException hata1)
{
    //Taşma hatası meydana gelirse çalıştırılacak kod bloğu
}

catch(ArithmeticException hata2)
{
    //Aritmetiksel hata meydana gelirse çalıştırılacak kod bloğu
}
```

Not: Finally kod bloğunun kullanılması tamamen programcının isteğine ve ihtiyacına bağlıdır. Kullanılması zorunlu değildir.

[http://bidb.itu.edu.tr/seyrifeteri/blog/2013/09/06/c-ta-i-stisnai-durum-y%C3%B6netimi-\(exception-handling\)](http://bidb.itu.edu.tr/seyrifeteri/blog/2013/09/06/c-ta-i-stisnai-durum-y%C3%B6netimi-(exception-handling))

## C#'ta İstisnai Durum Yönetimi (Exception Handling)

Eyl 06, 2013

Bazı programlar yazılırken hata vermediği halde çalışma sırasında hata verebilir. Bu hataları kontrol etme işlemine İstisnai Durum Yönetimi (Exception Handling)denir. .NET mimarisinde sık oluşan hataları yakalamak için gerekli sınıflar System.Exception sınıfı altında bulunmaktadır. Bu sınıfların yetersiz görüldüğü yerde programcı kendi hata sınıfını kendisi de yazabilir.

#### Sık Kullanılan İstisnai Durum Sınıfları

Çalışma zamanında beklenmedik bir hatanın oluşumu sonrasında oluşturulan nesnelere istisnai durum sınıf nesneleri denir. Bir hatanın oluşması, çalışma zamanında ilgili hatayı temsil eden sınıf türünden bir nesnenin oluşturulması anlamına gelir. Sık kullanılan istisnai durum sınıfları :

- **System.OutOfMemoryException:** Programın çalışması için yeterli bellek kalmadıysa oluşur.
- **System.StackOverflowException:** Stack (Yığın) bellek bölgesinin birden fazla metod için kullanılması durumunda oluşur. Genellikle kendini çağıran metodların hatalı kullanılmasıyla meydana gelir.
- **System.NullReferenceException:** Bellekte yer ayrılmamış bir nesne üzerinden sınıfın üye elemanlarına erişmeye çalışırken oluşur.
- **System.OverflowException:** Bir veri türüne kapasitesinden fazla veri yüklemeye çalışılırken oluşur.
- **System.InvalidCastException:** Tür dönüştürme operatörüyle geçersiz tür dönüşümü yapılmaya çalışıldığında oluşur.
- **System.IndexOutOfRangeException:** Bir dizinin olmayan elemanına erişilmeye çalışılırken fırlatılır.
- **System.ArrayTypeMismatchException:** Bir dizinin elemanına yanlış türde veri atanmaya çalışılırken oluşur.
- **System.DividedByZero:** Sıfıra bölme yapıldığı zaman oluşur.
- **System.ArithmeticException:** DividedByZero ve OverflowException bu sınıftan türemiştir. Hemen hemen matematikle ilgili tüm istisnaları yakalayabilir.
- **System.FormatException:** Metodlara yanlış biçimde parametre verildiğinde oluşur.

#### Tüm Hata Sınıflarında Bulunan Önemli Üye Elemanlar

Yukarıda bahsedilen System.Exception sınıfından türeyen bu hata sınıflarının kendine ait üye elemanları vardır. Bu üye elemanlar programcının kendi yazacağı sınıflar da dahil tüm sınıflara kalıtım yoluyla geçmiştir. System.Exception sınıfının önemli üye elemanları :

- **Message (Mesaj):** Ortaya çıkan hatayla ilgili açıklayıcı bir mesaj saklar.
- **Source (Kaynak):** İstisnai durum nesnesinin gönderildiği uygulama ya da dosyanın adıdır.
- **StackTrace (Yığınizi):** Hatanın oluştuğu metod ve program hakkında bilgi içerir.
- **HelpLink (YardımBağlantısı):** Hatayla ilgili olan yardım dosyasının yol bilgisini saklar.
- **TargetSite (HedefAlanı):** İstisnai durumu yaratan metod ile ilgili bilgi verir.
- **InnerException (Dahiliİstisna):** "catch" bloğu içerisinden bir hata yaratılırsa "catch" bloğuna gelmesine yol açan istisnai durumun Exception nesnesidir.
- **ToString (Dizgiye):** Bu metod ilgili hataya ilişkin hata metninin tamamını dizi olarak döndürür.

#### İstisnai Durum Yakalama

Bahsedilen sınıfların kullanılabilmesi için gerekli bazı anahtar sözcükler vardır. Bu sözcükler try, catch, finally ve throw'dur.

#### try-catch (dene-yakala) Blokları

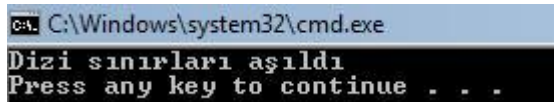
"try" bloğu içine hata oluşabilecek kısım, "catch" içine ise programın verebileceği muhtemel hata yazılır. Art arda birkaç "catch" bloğu kullanılabilir. Aşağıda try-catchbloğuna bir örnek verilmiştir.

```

namespace exception
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                int[] a = new int[4];
                Console.WriteLine(a[5]);
            }
            catch (IndexOutOfRangeException)
            {
                Console.WriteLine("Dizi sınırları aşıldı");
            }
        }
    }
}

```

Ekran çıktısı:



try-catch-finally (dene-yakala-sonuç) Blokları finally bloğu da try-catch ardına yazılarak programın hata verip vermediğine bakmadan çalışır. Finally bloğunun kullanılması zorunlu değildir ve kullanıldığı takdirde kendi üstündeki try-catch bloklarının tüm bağlantılarını kapatır, yani üst bloklarda veritabanıyla bir bağlantı yapılmışsa bu kapatılır. Aşağıda try-catch-finally bloğunun bir örneği bulunmaktadır.

```

namespace exception{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Sayı giriniz");
            try
            {
                int i = int.Parse(Console.ReadLine());
            }
            catch (FormatException exp)
            {
                Console.WriteLine(exp.Message);
            }
            catch (OverflowException exp)
            {
                Console.WriteLine(exp.Message);
            }
            finally
            {
                Console.WriteLine("finally bloğu çalışıyor...");
            }
        }
    }
}

```

Bu program C#'ta tanımlı tamsayı(integer) sınırlarını aşan yanlış bir değere eşitlendiğinde hata verir, int sınırları içinde bir değere atanırsa hata vermez. Her iki durumda da "finally" bloğu çalışır.

Doğru değer girildiğinde ekran çıktısı:

```
C:\Windows\system32\cmd.exe
Sayı giriniz
5
finally bloğu çalışıyor...
Press any key to continue . . .
```

Yanlış değer girildiğinde ekran çıktısı:

```
C:\Windows\system32\cmd.exe
Sayı giriniz
123994672365436536
Value was either too large or too small for an Int32.
finally bloğu çalışıyor...
Press any key to continue . . .
```

throw (fırlat) Anahtar Sözcüğü

throw ifadesiyle istenilen istisna istenilen anda ortaya çıkarılır. Bu istisnanın kullanıldığı yerde program durur ve istenen istisnayı üretir. Aşağıda "throw" anahtar sözcüğünün kullanıldığı bir örnek yer almaktadır.

namespace exception

```
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                throw new DivideByZeroException();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Bu örnekte sıfıra bölünebilme hatası programcı tarafından oluşturulmuş ve bu hata "try" bloğunu "catch" bloğuna düşürmüştür.

Ekran çıktısı:

```
C:\Windows\system32\cmd.exe
Attempted to divide by zero.
Press any key to continue . . .
```