try catch Yapısı

Hatalar programın çalışması esnasında oluşurlar ve programın kendi çalışmasının kesilmesine neden olurlar. Bu hatalar kullanıcı kaynaklı, yazılımcı kaynaklı ve sistem kaynaklı olabilir.

Burada yazılımcıya düşen oluşabilecek hataların analizini yaparak programın çalışmasının kesilmesini engellemektir.

Programın hiç hata vermemesini sağlamak da tamamen doğru bir yaklaşım değildir.

Doğru olan yaklaşım; oluşabilecek hataların analizinin yapılması, gerekirse; kullanıcıyı bilgilendirerek ondan alınan geri dönüşlerle, gerekmiyorsa; programın kendi içerisinde yanlış sonuçlar üretmesine neden olmayacak şekilde programın akışını yönlendirerek, oluşabilecek hataları kontrol etmektir.

try catch Yapısı

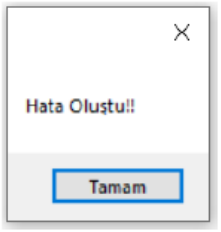
oluşan hataları yakalamak için try catch yapısı kullanılır.

Kullanımı :

```
try
{
        Hatanın oluşabileceği kod satırları
}
catch
{
        Hata oluştuğunda çalıştırılacak kod satırları
}
finally
{
        Hata olsada olmasada çalıştırılacak kod satırları
}
```

Örnek

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        int sıfır=0;
        int sonuc;
        sonuc = 10 / sıfır;
    }
    catch
    {
        MessageBox.Show("Hata Oluştu!!");
    }
}
```



Hatalar (İstisnai Durumlar)

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

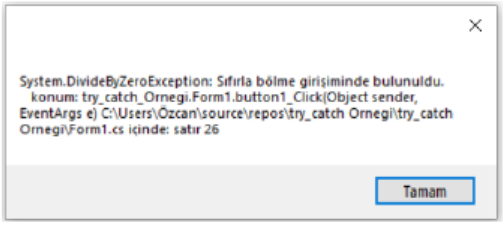
System.Exception sınıfından türeyen bu hata sınıflarının kendine ait üye elemanları vardır. Bu üye elemanlar programcının kendi yazacağı sınıflar da dahil tüm sınıflara kalıtım yoluyla geçmiştir. System.Exception sınıfının önemli üye elemanları :

Message (Mesaj): Ortaya çıkan hatayla ilgili açıklayıcı bir mesaj saklar.
Source (Kaynak): İstisnai durum nesnesinin gönderildiği uygulama ya da dosyanın adıdır.
StackTrace (YığınİzI): 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 gelinmesine yol açan istisnai durumun Exception nesnesidir.
ToString (Dizgiye): Bu metod ilgili hataya ilişkin hata metninin tamamını dizi olarak döndürür.

Exception Nesnesi

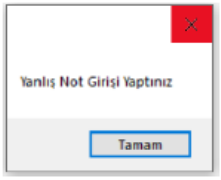
Program içerisinde oluşan hatalar Exception nesnesinden türetilmektedir. Exception nesnesi System sınıfının altında yer almaktadır.oluşan hataların niteliğini ayırt edebilmek için Exception nesnesi içerisinde saklanan hatanın hangi sınıfa ait olduğu değerlendirilerek işlem yapılabilir.

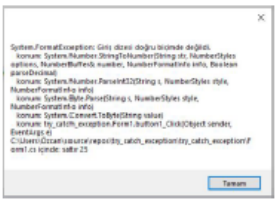
```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        int sıfır=0;
        int sonuc;
        sonuc = 10 / sıfır;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```

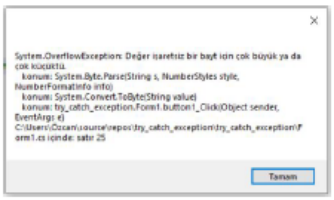


Örnek

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        byte notu;
        notu = Convert.ToByte(Interaction.InputBox("Notu Girin :", "Not Girişi"));
        if (notu < 50)
            MessageBox.Show("KALDINIZ");
        else if (notu >= 50 & notu <= 100)
            MessageBox.Show("GEÇTİNİZ");
    }
    catch
    {
        MessageBox.Show("Yanlış Not Girişi Yaptınız");
    }
    //catch (Exception ex)
    //{
    //    MessageBox.Show(ex.ToString());
    //}
}
```







Örnek

```
static void Main(string[] args)
{
    int deger, sonuc, bolen;
    try
    {
    Console.Write("Sayıyı Girin :");
    deger = Convert.ToInt32(Console.ReadLine());
    Console.Write("Bolen Sayıyı Girin :");
    bolen = Convert.ToInt32(Console.ReadLine());
    sonuc = deger / bolen;
    }
    catch (DivideByZeroException)
    {
        Console.WriteLine("Sıfıra Bölme Hatası");
        sonuc = int.MaxValue;
    }
    catch(ArithmeticException)
    {
        Console.WriteLine("Aritmetik Hata");
        sonuc = int.MaxValue;
    }
    catch
    {
        Console.WriteLine("Tanımlanmamış Hata");
        sonuc = int.MaxValue;
    }
    Console.WriteLine("Sonuç = " + sonuc );
    Console.ReadLine();
}
```

```
Sayıyı Girin :100
Bolen Sayıyı Girin :5
Sonuç = 20
```

```
Sayıyı Girin :100
Bolen Sayıyı Girin :0
Sıfıra Bölme Hatası
Sonuç = 2147483647
```

```
Sayıyı Girin :21212211221212
Aritmetik Hata
Sonuç = 2147483647
```

```
Sayıyı Girin :aaaaa
Tanımlanmamış Hata
Sonuç = 2147483647
```