

# EXCEPTION HANDLING IN C#

## İstisna Yakalama (Exception Handling )

# Hata Nedir?

## Hata Türleri

---

Hata Nedir?

# Hata Nedir?

Hatalar (errors), program geliştirme veya yürütme sırasında meydana gelen hata veya kusurları ifade eder. Bunları bulup düzeltmezseniz programın yanlış sonuçlar vermesine neden olurlar.

Hatalar, kullanıcı kaynaklı, program geliştiren yazılımcı kaynaklı ya da sistemsel bir hata olabilir.

Yazılımcı, tüm hataların analizini yaparak programın sürekli çalışmasını sağlayacak tedbirler almalıdır.




# Hata Türleri

## 1. Sözdizimi Hatası (Syntax)

- Kodda yazım hatası yaptığınızda, geliştirme sırasında sözdizimi hataları oluşur.
- Örnek değer ataması yapılmayan değişkeni kullanmak.

```
0 references
static void Main(string[] args)
{
    int sayi;
    Console.WriteLine(sayi);
}
```

 (local variable) int sayi

CS0165: Use of unassigned local variable 'sayi'

Show potential fixes (Alt+Enter or Ctrl+.)

## 2. Çalışma Zamanı Hatası (Exception)

- Çalışma zamanı hataları, programın yürütülmesi sırasında ortaya çıkar. Bunlara istisnalar da denir. Bunun nedeni yanlış kullanıcı girdileri, yanlış tasarım mantığı veya sistem hataları olabilir.

```
#region runtime error exception
int sayi = 12;
double bolme = sayi / 0;
#endregion
```

Exception Unhandled

System.DivideByZeroException: 'Attempted to divide by zero.'

Show Call Stack | View Details | Copy Details | Start Live Share session

Exception Settings

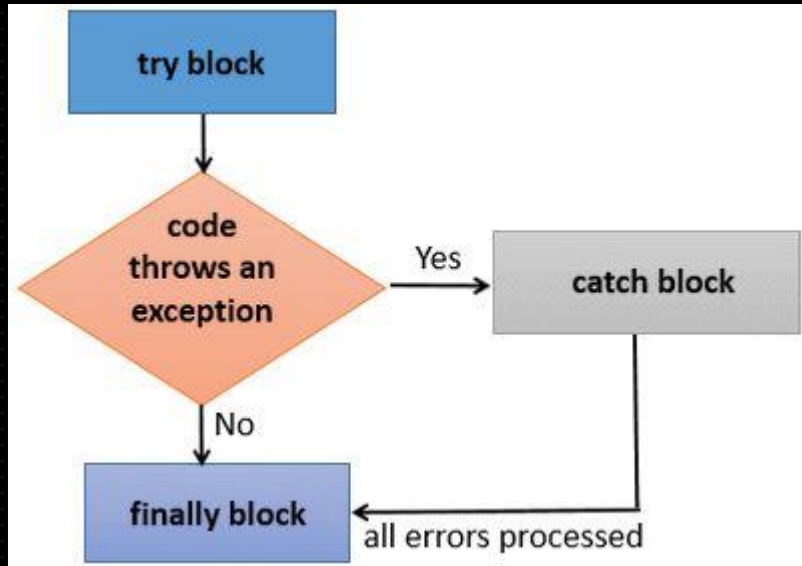
## 3. Mantıksal Hata (Logical)

- Program düzgün yazıldığı halde istenilen sonucu vermediğinde mantık hataları oluşur. Mantık hatalarını bulmak zordur çünkü sonucun yanlış olduğundan emin olmanız gerekir.

```
#region mantıksal hata
int s1, s2, toplam;
s1 = 20;
s2 = 10;
toplam = s1 - s2; //sonuç 10 halbuki toplam=s1+s2
#endregion
```

# İstisna Yakalama (Exception Handling)

İstisna işleme, çalışma zamanı hatalarını algılamak ve işlemek için kullanılan bir mekanizmadır. **try-catch-finally** blokları ve **throw** anahtar sözcüğü kullanılarak elde edilir..



```
try
{
    //Code
}
catch(DivideByZeroException dbe)
{
    //Code
}
catch (Exception e)
{
    //Code
}
finally
{
    //Code
}
```

```
try
{
    //Code
}
finally
{
    //Code
}
```

# İstisna Yakalama (Exception Handling)

```
try
{
    //ististanaya yolaçabilecek ifadeler
}
catch(Exception exc )
{
    //istisna varsa işler
    Console.WriteLine(exc.ToString());
}
finally
{
    /* Nihayet blok, bir istisna atılsa bile
    * kullanılmayan kaynakları serbest bırakmak,
    * herhangi bir temizleme işlemi için kullanılabilir.
    * Örneğin,veritabanı/dosya bağlantısını kapatma.*/
}
```



# İstisnalar İle İlgili Önemli Noktalar

1. İstisnalar, tümü doğrudan veya dolaylı olarak **System.Exception** sınıfından türetilir.

## FormatException

```
...public class SystemException : Exception
{
    ...public SystemException();
    ...public SystemException(string message);
    ...protected SystemException(SerializationInfo info, StreamingContext context);
    ...public SystemException(string message, Exception innerException);
}
```

```
...public class FormatException : SystemException
{
    ...public FormatException();
    ...public FormatException(string message);
    ...protected FormatException(SerializationInfo info, StreamingContext context);
    ...public FormatException(string message, Exception innerException);
}
```

## OverFlowException

```
...public class SystemException : Exception
{
    ...public SystemException();
    ...public SystemException(string message);
    ...protected SystemException(SerializationInfo info, StreamingContext context);
    ...public SystemException(string message, Exception innerException);
}
```

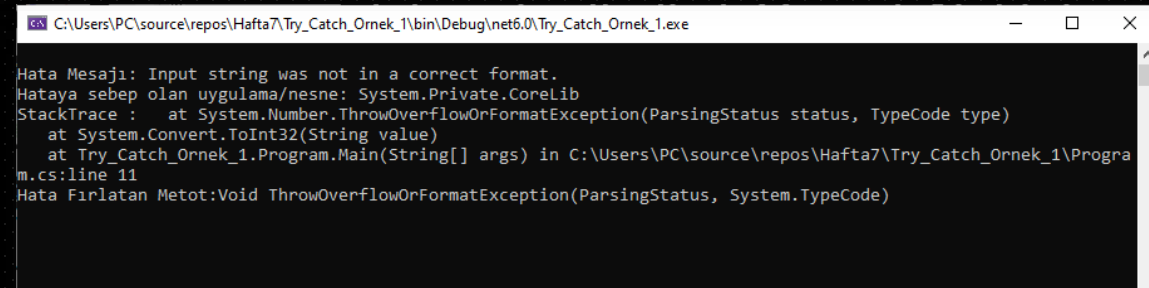
```
...public class OverflowException : ArithmeticException
{
    ...public OverflowException();
    ...public OverflowException(string message);
    ...protected OverflowException(SerializationInfo info, StreamingContext context);
    ...public OverflowException(string message, Exception innerException);
}
```

```
...public class ArithmeticException : SystemException
{
    ...public ArithmeticException();
    ...public ArithmeticException(string message);
    ...protected ArithmeticException(SerializationInfo info, StreamingContext context);
    ...public ArithmeticException(string message, Exception innerException);
}
```

# İstisnalar İle İlgili Önemli Noktalar

2. İstisna nesnesi, **stack** durumu ve hatanın metin açıklaması gibi hata hakkında ayrıntılı bilgiler içerir.

```
int sayi;  
try  
{  
    Console.Write("Bir sayı giriniz:");  
    sayi = Convert.ToInt32(Console.ReadLine());  
}  
catch (Exception exc)  
{  
    Console.WriteLine("\nHata Mesajı: {0}", exc.Message);  
    //Console.WriteLine("Yardım Linki: {0}", exc.HelpLink);  
    Console.WriteLine("Hataya sebep olan uygulama/nesne: {0}", exc.Source);  
    Console.WriteLine("StackTrace :{0}", exc.StackTrace);  
    Console.WriteLine("Hata Fırlatan Metot:{0}", exc.TargetSite);  
}
```



C:\Users\PC\source\repos\Hafta7\Try\_Catch\_Ornek\_1\bin\Debug\net6.0\Try\_Catch\_Ornek\_1.exe

Hata Mesajı: Input string was not in a correct format.  
Hataya sebep olan uygulama/nesne: System.Private.CoreLib  
StackTrace : at System.Number.ThrowOverflowOrFormatException(ParsingStatus status, TypeCode type)  
at System.Convert.ToInt32(String value)  
at Try\_Catch\_Ornek\_1.Program.Main(String[] args) in C:\Users\PC\source\repos\Hafta7\Try\_Catch\_Ornek\_1\Program.cs:line 11  
Hata Fırlatan Metot:Void ThrowOverflowOrFormatException(ParsingStatus, System.TypeCode)

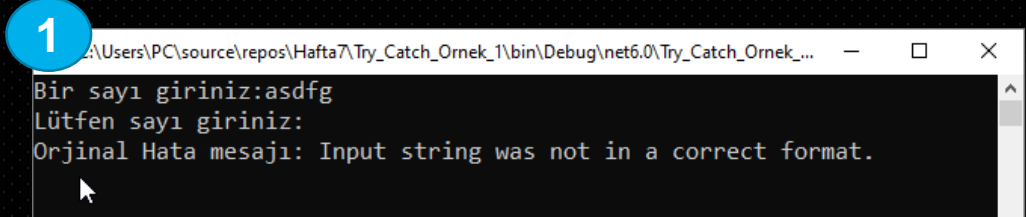


# İstisnalar İle İlgili Önemli Noktalar

3. Bir `try` bloğu, birden fazla `catch` bloğunun yakalayacağı birden çok istisna işlenebilir.

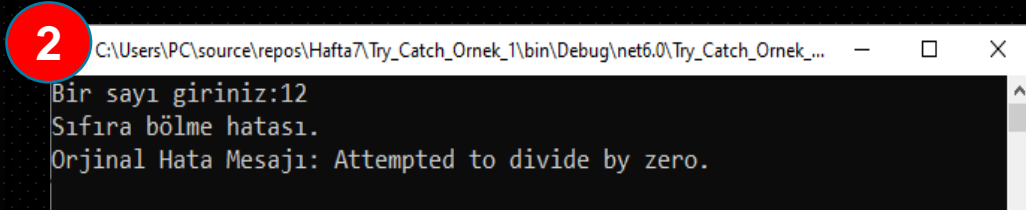
```
int sayi, bolme;
try
{
    Console.Write("Bir sayı giriniz:");
    sayi = Convert.ToInt32(Console.ReadLine());
    bolme = sayi / 0;
}
catch (FormatException exc)
{
    Console.WriteLine("Lütfen sayı giriniz:");
    Console.WriteLine("Orjinal Hata mesajı: {0}", exc.Message);
}
catch (DivideByZeroException exc)
{
    Console.WriteLine("Sıfıra bölme hatası.");
    Console.WriteLine("Orjinal Hata Mesajı: {0}", exc.Message);
}
```

1



```
C:\Users\PC\source\repos\Hafta7\Try_Catch_Ornek_1\bin\Debug\net6.0\Try_Catch_Ornek_...
Bir sayı giriniz:asdfg
Lütfen sayı giriniz:
Orjinal Hata mesajı: Input string was not in a correct format.
```

2

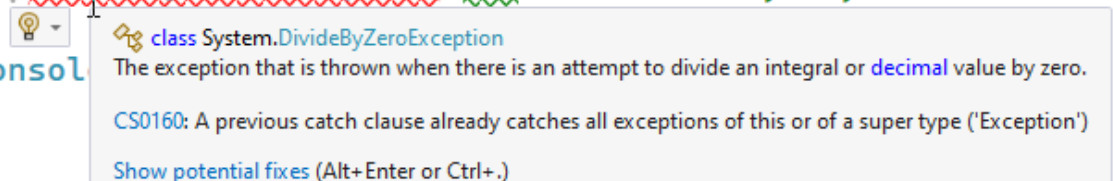


```
C:\Users\PC\source\repos\Hafta7\Try_Catch_Ornek_1\bin\Debug\net6.0\Try_Catch_Ornek_...
Bir sayı giriniz:12
Sıfıra bölme hatası.
Orjinal Hata Mesajı: Attempted to divide by zero.
```

# İstisnalar İle İlgili Önemli Noktalar

4. Daha özel yakalama bloğu, genelleştirilmiş olandan önce gelmelidir. Aksi takdirde özel `catch` bloğu asla yürütülmez.

```
int sayi, bolme;
try
{
    Console.WriteLine("Bir sayı giriniz:");
    sayi = Convert.ToInt32(Console.ReadLine());
    bolme = sayi / 0;
}
catch (Exception exc)
{
    Console.WriteLine("Orjinal Hata mesajı: {0}", exc.Message);
}
catch (DivideByZeroException exc) //bu blok çalışmaz
{
    Console.WriteLine("DivideByZeroException");
}
```



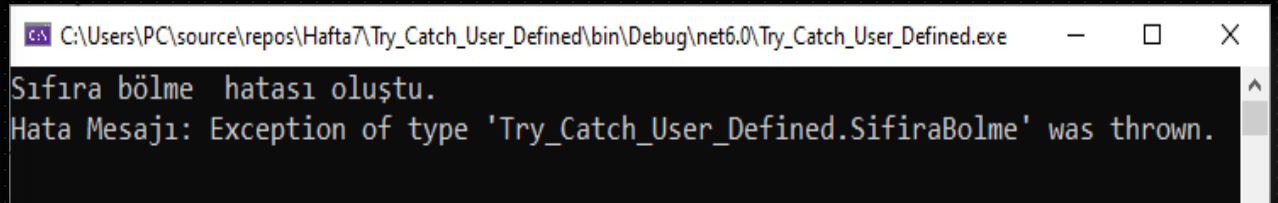
The tooltip shows the definition of `class System.DivideByZeroException`: "The exception that is thrown when there is an attempt to divide an integral or decimal value by zero." It also displays the error code `CS0160: A previous catch clause already catches all exceptions of this or of a super type ('Exception')` and a link to "Show potential fixes (Alt+Enter or Ctrl+.)".

# İstisnalar İle İlgili Önemli Noktalar

5. İstisnalar, **throw** anahtar sözcüğü kullanılarak bir program tarafından açıkça oluşturulabilir. (Kullanıcı Tanımlı İstisna- User Defined Exception )

```
internal class Program
{
    1 reference
    public static double BolmeIslemi(double pay, double payda)
    {
        if (payda == 0)
        {
            throw new SifiraBolme();
        }
        return pay / payda;
    }
    0 references
    static void Main(string[] args)
    {
        double sonuc, sayi = 12, sayi2 = 0;
        try
        {
            sonuc = BolmeIslemi(sayi, sayi2);
        }
        catch (Exception exc)
        {
            Console.WriteLine("Hata Mesajı: {0}", exc.Message);
        }
    }
}
```

```
2 references
class SifiraBolme : Exception //SifiraBolme sınıfı Exception kalıtım
{
    1 reference
    public SifiraBolme()
    {
        Console.WriteLine("Sıfıra bölme hatası oluştu.");
    }
}
```



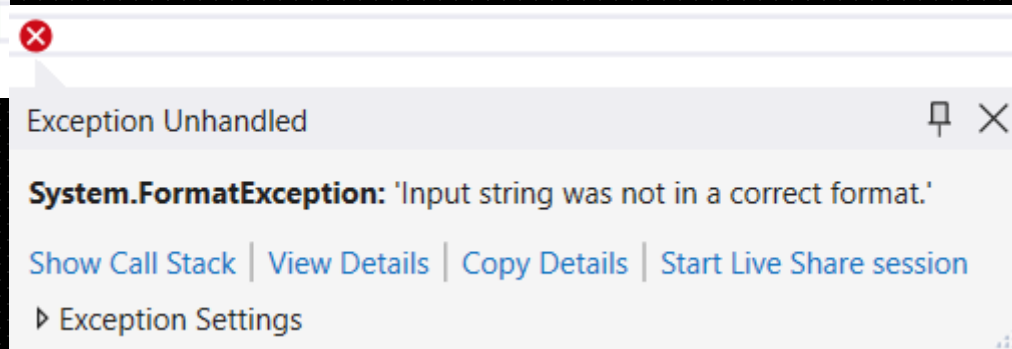
C:\Users\PC\source\repos\Hafta7\Try\_Catch\_User\_Defined\bin\Debug\net6.0\Try\_Catch\_User\_Defined.exe

Sıfıra bölme hatası oluştu.  
Hata Mesajı: Exception of type 'Try\_Catch\_User\_Defined.SifiraBolme' was thrown.

# İstisnalar İle İlgili Önemli Noktalar

6. Herhangi bir istisna işleme mekanizması kullanılmazsa, program bir hata mesajı vererek çalışmayı durdurur.

```
int sayi, bolme;  
Console.Write("Bir sayı giriniz:");  
sayi = Convert.ToInt32(Console.ReadLine());  
bolme = sayi / 0;
```



# İstisnalar İle İlgili Önemli Noktalar

7. Parantez veya bağımsız değişken içermeyen bir `catch` bloğu , `try` bloğu içinde meydana gelen tüm istisnaları yakalayabiliriz.

```
try
{
    //ististanaya yolaçabilecek ifadeler
}

catch(Exception)
{
    // tüm hataları yakalar.
}
```

```
try
{
    //ististanaya yolaçabilecek ifadeler
}

catch
{
    // tüm hataları yakalar.
}
```

