Exception Handling (İstisnai Durum işleme)

- Bir exception (istisnai durum) bir programın çalışması esnasında ortaya çıkan bir problemdir.
  - Sıfıra bölme (divide by zero)
  - Taşma Hatası (Overflow)
  - Yetersiz Hafıza (Out of Memory)
  - Donanımın bulunamaması
  - Kullanıcının yanlış değer girmesi vb.
- İstisnai durumlar, bir program parçasından diğerine geçiş kontrolü için imkan sağlar.
- C#'de istisnai durum işleme için try, catch, finally ve throw anahtar kelimeleri kullanılır. (Benzer durum Java içinde geçerlidir.)

Exception Handling (İstisnai Durum işleme)

- try: bir try bloğu istisnai durumların oluşabileceği kod bloklarını belirler. Bir veya birden fazla catch bloğu ile devam eder.
- catch: İstisnai durum bulunduğunda yapılacak işlemleri içeren kısımdır. Catch bir istisnai durumun yakalandığını gösterir. Eğer Catch kısmındaki kodlar çalışmıyorsa istisnai durum oluşmamıştır. Bir try birden fazla catch içerebilir.
- finally: Bu blok istisnai durumun oluşup oluşmadığına bakmaksızın verilen bir kod kümesini çalıştırmak için kullanılır. Kullanılması seçimlik bir durumdur, yani finally olmayan bir try-catch tanımlanabilir.
- throw: Bir programda bir problem oluştuğu zaman ilgili hata nesnesi sınıfını atmaktır (throw). Bunun için throw anahtar sözcüğü kullanılır.

Exception Handling (İstisnai Durum işleme)

```
try
{
    //Exception oluşturabilecek satırlar
}
catch (ExceptionName e1)
{
    //Hata yakalandığında yapılacak işlem
}
catch (ExceptionName e2)
{
    //Hata yakalandığında yapılacak işlem
}
catch (ExceptionName eN)
{
    //Hata yakalandığında yapılacak işlem
}
finally
{
    //Her halikarda çalıştırılacak satır
}
```

- Farklı hata durumları için birden fazla catch olabilir.
- Tüm hatalı durumları tek catch içinde de toplamanız mümkündür.
- Throw hata oluştuğunda oluşturulacak kısım olduğu için bu kod bloğu içinde yer almamıştır. (İleriki slaytlarda kullanımını göreceğiz.)

Exception Handling

```
class Program
{
    static void Main(string[] args)
    {
        int x = 5;
        int y = 0;
        //sıfıra bölme hatası
        try
        {
            int z = x / y;
            Console.WriteLine(z);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Sıfıra bölme hatası"); //Kişisel çıktı
            Console.WriteLine(ex.Message); //ex değeri içinden dönen çıktı
            Console.WriteLine(ex.Source); //ex değerinin kaynağı
        }

        //karakter içeren bir değeri sayıyı çevirme hatası
        string sayi = "rfdhjfd";
        try
        {
            Convert.ToInt32(sayi);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

Hatanın oluştuğu kısım, bu aşamadan sonra bağlı catch bloğu devreye girer. Hatanın oluştuğu kısımdaki bir alt satırdaki kod çalışmaz.

Bütün hataların ortak sınıfı

Hatanın oluştuğu kısım

Finally bloğu kullanma seçimliktir.

Exception Properties

- Exception özellikleri kullanarak hata hakkında bilgiler edinebiliriz.
  - HelpLink: Genelde boştur, string değer döndürür. Özellikle kendi exception'larınızda hata durumda bir web sayfasına kullanıcığı göndermek için Link tutmada kullanılabilir.
  - Message: Hatanın nedenini kısa bir mesajla verir. Mesaj sadece okunabilir durumdadır. (sadece get var demek ki!)
  - Source: Uygulama ismidir. String değerdir.
  - StackTrace: Derlenen programının dizinidir.
  - TargetSite: Hatanın oluştuğu yerdeki metotun ismidir. Hata kayıtlarını tutmadan yardımcı olur.

C# - Exception Handling

- C# istisnai durumları sınıflarda bulunur.
- C#'ta istisnai durum sınıfları, System.Exception sınıfından direct / indirect olarak oluşur.
- Örneğin System.ApplicationException ve System.SystemException sınıfları System.Exception sınıfından kalıtılmıştır. (derived)
- System.ApplicationException uygulama programları tarafından oluşturulan istisnai durumları destekler. System.SystemException sınıfı ise sistem istinai durumları için tanımlanmış bir base sınıftır.
- The System.ApplicationException class supports exceptions generated by application programs. Hence the exceptions defined by the programmers should derive from this class.

C# - Exception Handling

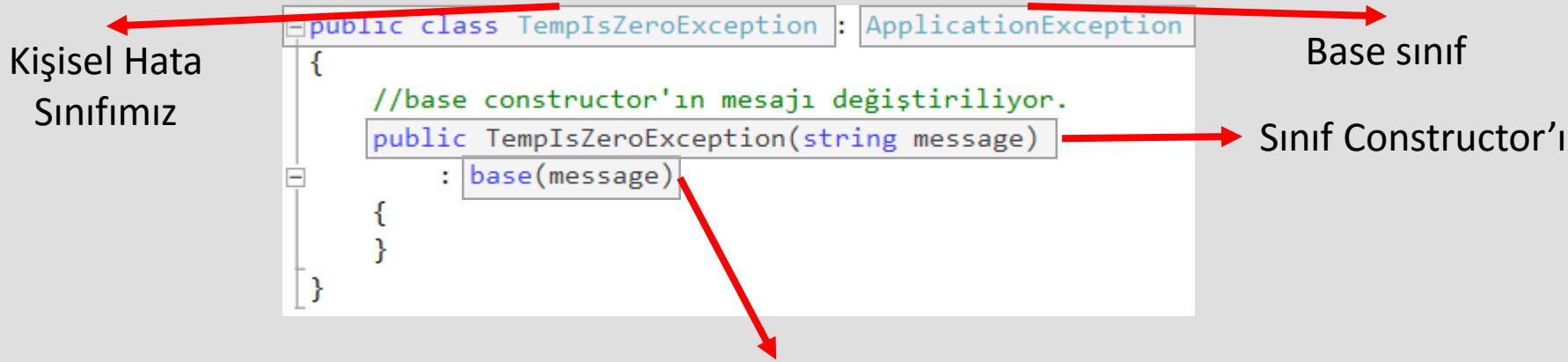
- System.SystemException'dan türemiş bazı istisnai durum sınıfları:

Exception Sınıfı	Tanımı
System.IO.IOException	I / O hatalarını işler. (Dosya yazma / okuma)
System.IndexOutOfRangeException	Bir dizi dışına çıkıldığında oluşan hataları işler.
System.ArrayTypeMismatchException	Bir veri tipi, dizi tipi ile uyuşmadığında oluşan hataları işler.
System.NullReferenceException	Bir null nesneden kaynaklandığında oluşan hataları işler.
System.DivideByZeroException	Sıfıra bölme hatalarını işler.
System.InvalidCastException	Veri tipinden yanlış işlenmesinden kaynaklı hataları işler.
System.OutOfMemoryException	Yetersiz hafıza durumunda üretilen hataları işler.
System.StackOverflowException	Stack taştığında oluşabilecek hataları işler.

Daha fazla hata sınıfı vardır. Uygulamaya göre de farklı hata sınıfları karşınıza çıkabilir.

User-Defined Exceptions (Kullanıcı-Tanımlı)

- Kendi istisnai durumunuzu ayrıca tanımlayabilirsiniz.
- Kullanıcı-tanımlı istinai durum sınıfları ApplicationException sınıftan türetilir.



- Base sınıfın constructor'ı üzerinden string değer gönderme ile base'in verdiği hatayı değiştirme.
- Normal şartlarda ApplicationException sınıfı «Application Error» veya «Uygulama Hatası» gibi hata mesajı verir. Bu örnekte bu hata mesajını değiştiriyoruz.

User-Defined Exceptions (Kullanıcı-Tanımlı)

```
public class TempIsZeroException : ApplicationException
{
    //base constructor'ın mesajı değiştiriliyor.
    public TempIsZeroException(string message)
        : base(message)
    {
    }
}

public class Temperature
{
    public int temperature = 0;

    public void showTemp()
    {
        if (temperature == 0)
        {
            throw (new TempIsZeroException("Zero Temperature found"));
        }
        else
        {
            Console.WriteLine("Temperature: {0}", temperature);
        }
    }
}
```

Throw hata oluşturmak için kullanılır. Throw ile bir nesne tetiklenebilir.

User-Defined Exceptions (Kullanıcı-Tanımlı)

- Eğer direkt showtemp metotu çağrılırsa program hataya düşer. temperature değeri o'dan farklı yapılıp çağrılırsa catch bölümüne düşmeden program çalışır.
- Bu örnekte try-catch kullanımı ile kendi yarattığımız bir exception'ı kullandık.

```
class Program
{
    static void Main(string[] args)
    {
        Temperature temp = new Temperature();
        try
        {
            //temp.temperature = 10;
            temp.showTemp();
        }
        catch (TempIsZeroException e)
        {
            Console.WriteLine("TempIsZeroException: {0}", e.Message);
        }
        Console.ReadKey();
    }
}
```

Kişisel Exception Sınıfımız

Bu string bilgiyi güncellemiştik.