# Chapter 25: Disposable objects

## Section 25.1: Basic concept of IDisposable

Any time you instantiate a class that Implements `IDisposable`, you should call `.Dispose`1 on that class when you have finished using it. This allows the class to clean up any managed or unmanaged dependencies that it may be using. Not doing this could cause a memory leak.

The `Using` keyword ensures that `.Dispose` is called, without you having to *explicitly* call it.

For example without `Using`:

```
Dim sr As New StreamReader("C:\foo.txt")
Dim line = sr.ReadLine
sr.Dispose()
```

Now with `Using`:

```
Using sr As New StreamReader("C:\foo.txt")
    Dim line = sr.ReadLine
End Using '.Dispose is called here for you
```

One major advantage `Using` has is when an exception is thrown, because it *ensures* `.Dispose` is called.

Consider the following. If an exception is thrown, you need to need to remember to call .Dispose but you might also have to check the state of the object to ensure you don't get a null reference error, etc.

```
    Dim sr As StreamReader = Nothing
    Try
        sr = New StreamReader("C:\foo.txt")
        Dim line = sr.ReadLine
    Catch ex As Exception
        'Handle the Exception
    Finally
        If sr IsNot Nothing Then sr.Dispose()
    End Try
```

A using block means you don't have to remember to do this and you can declare your object inside the `try`:

```
    Try
        Using sr As New StreamReader("C:\foo.txt")
            Dim line = sr.ReadLine
        End Using
    Catch ex As Exception
        'sr is disposed at this point
    End Try
```

1 Do I always have to call Dispose() on my DbContext objects? Nope

## Section 25.2: Declaring more objects in one Using

Sometimes, you have to create two `Disposable` objects in a row. There is an easy way to avoid nesting `Using` blocks.

This code

```
Using File As New FileStream("MyFile", FileMode.Append)
    Using Writer As New BinaryWriter(File)
        'You code here
        Writer.Writer("Hello")
    End Using
End Using
```

can be shortened into this one. The main advantage is that you gain one indentation level:

```
Using File As New FileStream("MyFile", FileMode.Append), Writer As New BinaryWriter(File)
    'You code here
    Writer.Writer("Hello")
End Using
```