```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Markup;
using System.Xml.Linq;

// Back in the day while working as a software engineer for Sybase
// all engineers had to enter the amount of time they spent on various
// things each week so that Accounting could capitalize the R&D related
// tasks. The time accounting system was built on top of a Sybase SQL
// Server database, so it didn't take too long before one of the software
// engineers wrote a script to randomly populate the week's time values
// with "reasonable" numbers, and then everyone simply set that script
// up to run as a cron job every Friday. This program is meant to
// emulate that, except it is meant to be run by a human, not Scheduled
// Tasks, if only to show off the WPF bits.
//
// There is an old theory that says that in any C-derived language any
// non-trivial program can be written such that it only uses a single
// for statement with an empty body, with all work happening in the
// for's initializer, test and iterator expressions with an empty body.
// This is an attempt to do just that using multiple things learned in
// class, including:
//
// o  Anonymous classes
// o  Anonymous functions
// o  Generic types
// o  Lamda expressions
// o  Linq to Objects
// o  Linq to SQL
// o  Object initializers
// o  Threading
// o  WPF
// o  XAML, including:
//    - animation
//    - images
//    - element binding
// o  XElement
// o  Reflection (not taught in class, but had to use)
//
// All with only one statement and two semicolons!
//
//     for ( ; ; ) {}
//
// There is absolutely NO error handling in this (because try's would be another
// statement, not an expression). And I would fire anyone working for me that
// wrote something like this. :-)
//
// Written by Jim Lehmer, jim.lehmer@gmail.com.
//
namespace Challenge1
```

```
{
    class Program
    {
        [STAThread] // <=== Required for WPF.
        static void Main(string[] args)
        {
            for // <=== Only one statement.
            (
                // Use anonymous class as "iterator" variable. Need to use it as a
                // wrapper because C#'s for will not allow me to declare variables
                // of multiple types in the for initializer.
                //
                var i = new // <=== Call it i for convention's sake. :-)
                {
                    //
                    // Gratuitous use of XElement to construct an XML element with multiple attributes.
                    //
                    xmlAnim = new XElement
                    (
                        "DoubleAnimation",
                        new XAttribute("Storyboard.TargetName", "myVB"),
                        new XAttribute("Storyboard.TargetProperty", "(Viewbox.Opacity)"),
                        new XAttribute("From", "1"),
                        new XAttribute("To", "0"),
                        new XAttribute("Duration", "0:0:2"),
                        new XAttribute("AutoReverse", "True"),
                        new XAttribute("RepeatBehavior", "Forever")
                    ),
                    totalHours = new List<float>(),
                    user = System.Environment.GetEnvironmentVariable("USERNAME"),
                    ctx = new NorthwindDataContext(),
                    rnd = new Random(),
                    //
                    // Use generic types.
                    //
                    categories = new List<string> { "Meetings", "Coding", "Debugging", "Web surfing" }.GetEnumerator() as IEnumerator<string>, // <=== Thanks,
Mark!
                    app = new Application() { ShutdownMode = ShutdownMode.OnLastWindowClose },
                    mutex = new Mutex(false, "myMutex")
                }
                //   *
                //  *
            ;   // *************** FIRST SEMICOLON ***************
                //  *
                //   *

                //
                // Use Linq to Objects
                //
                ((from hours in i.totalHours select hours).Sum()) <= 40d &&
                  i.categories.MoveNext() ? true :
                  //
                  // Anonymous method.
                  //
                  new Func<bool>
                  (
                    //
```

```
// Lamda expression taking no parameters is the body of the anonymous method
//
() =>
Double.IsNaN // <=== KLUDGE!, but gets us an arbitrary false back (because Application.Run returns a valid number),
             // which we want to return after all the side effects below.
(
    //
    // Use WPF at run-time.
    //
    i.app.Run
    (
        new Window()
        {
            Title = "Only One STATEMENT (but two semicolons) Demo",
            Width = 400,
            Height = 300,
            Name = "myWindow",
            //
            // Build and load XAML at run-time (cast not strictly necessary).
            //
            Content = (Viewbox)XamlReader.Load // <=== Second attempt at async here using AsyncLoad. Can't attach LoadCompleted event.
            (
                new MemoryStream
                (
                    Encoding.UTF8.GetBytes // <=== Thanks, Mark!
                    (
                        @"<Viewbox xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation' " +
                            @"xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml' " + // <=== Thanks, Mike!
                            @"x:SynchronousMode='Async' " +
                            @"Name='myVB' Width='400' Height='300' >" +
                            @"<Label Name='myLabel' " +
                                @"VerticalAlignment='Center' HorizontalAlignment='Center' " +
                                @"VerticalContentAlignment='Center' HorizontalContentAlignment='Center' " +
                                //
                                // Binding to other element
                                //
                                @"Width='{Binding ElementName=myVB, Path=Width}' " +
                                @"Height='{Binding ElementName=myVB, Path=Height}' " +
                                @"FontWeight='Bold' FontSize='16pt' " +
                            @">" +
                                @"Total hours: " +
                                //
                                // Linq to Objects.
                                //
                                (from hours in i.totalHours select hours).Sum().ToString() +
                                @"<Label.Background>" +
                                    @"<ImageBrush ImageSource='Bliss.jpg' />" +
                                @"</Label.Background>" +
                            @"</Label>" +
                            //
                            // Stupid animation.
                            //
                            @"<Viewbox.Triggers>" +
                                @"<EventTrigger RoutedEvent='Viewbox.Loaded' SourceName='myVB'>" +
                                    @"<BeginStoryboard>" +
                                        @"<Storyboard>" +
                                            //
```

```
                                                    // REALLY gratuitous use of that gratuitous XElement!
                                                    //
                                                    i.xmlAnim.ToString() +
                                                @"</Storyboard>" +
                                            @"</BeginStoryboard>" +
                                        @"</EventTrigger>" +
                                    @"</Viewbox.Triggers>" +
                                @"</Viewbox>"
                            )
                        ),
                        //
                        // Black magic needed to make relative path in ImageBrush.ImageSource to work.
                        //
                        new ParserContext() { BaseUri = new Uri("pack://siteoforigin:,,,") } // <=== Thanks, Google!
                    )
                }
            )
        )
    ).Invoke() // <=== Call our anonymous method.
    //   *
    //  *
;   // *************** LAST SEMICOLON ***************
    //  *
    //   *

    i.totalHours.Add(i.rnd.Next(10, 20)),
    i.mutex.WaitOne(),
    //
    // Use Linq to SQL and object initializers.
    //
    i.ctx.TimeCards.InsertOnSubmit
    (
        new TimeCard()
        {
            Category = i.categories.Current,
            Hours = i.totalHours.Last(),
            UserId = i.user,
            Year = System.DateTime.Now.Year,
            Week = ((System.DateTime.Now.DayOfYear / 7) + 1)
        }
    ),
    i.mutex.ReleaseMutex(),
    //
    // How 'bout some gratuitous threading using a lamda expression as an anonymous function
    // and using reflection to call the methods in the lamda expression (because two of the methods
    // return void, it is hard to use them in an expression otherwise).
    //
    new Thread
    (
        new ThreadStart
        (
            () =>
            new StringBuilder
            (
                i.mutex.GetType().GetMethod("WaitOne", new Type[] {}).Invoke(i.mutex, new object[] {}).ToString().Length +
                (int)(i.ctx.GetType().GetMethod("SubmitChanges", new Type[] {}).Invoke(i.ctx, null) == null ? 0 : 1) +
                (int)(i.mutex.GetType().GetMethod("ReleaseMutex", new Type[] {}).Invoke(i.mutex, new object[] {}) == null ? 0 : 1)
```

```
                        ).ToString()
                    )
                ) { IsBackground = true }.Start()
            )
            { /* Intentionally empty - coulda just closed the for statement with a semicolon, but that woulda added another semicolon! :-) */ }
        }
    }
}
```