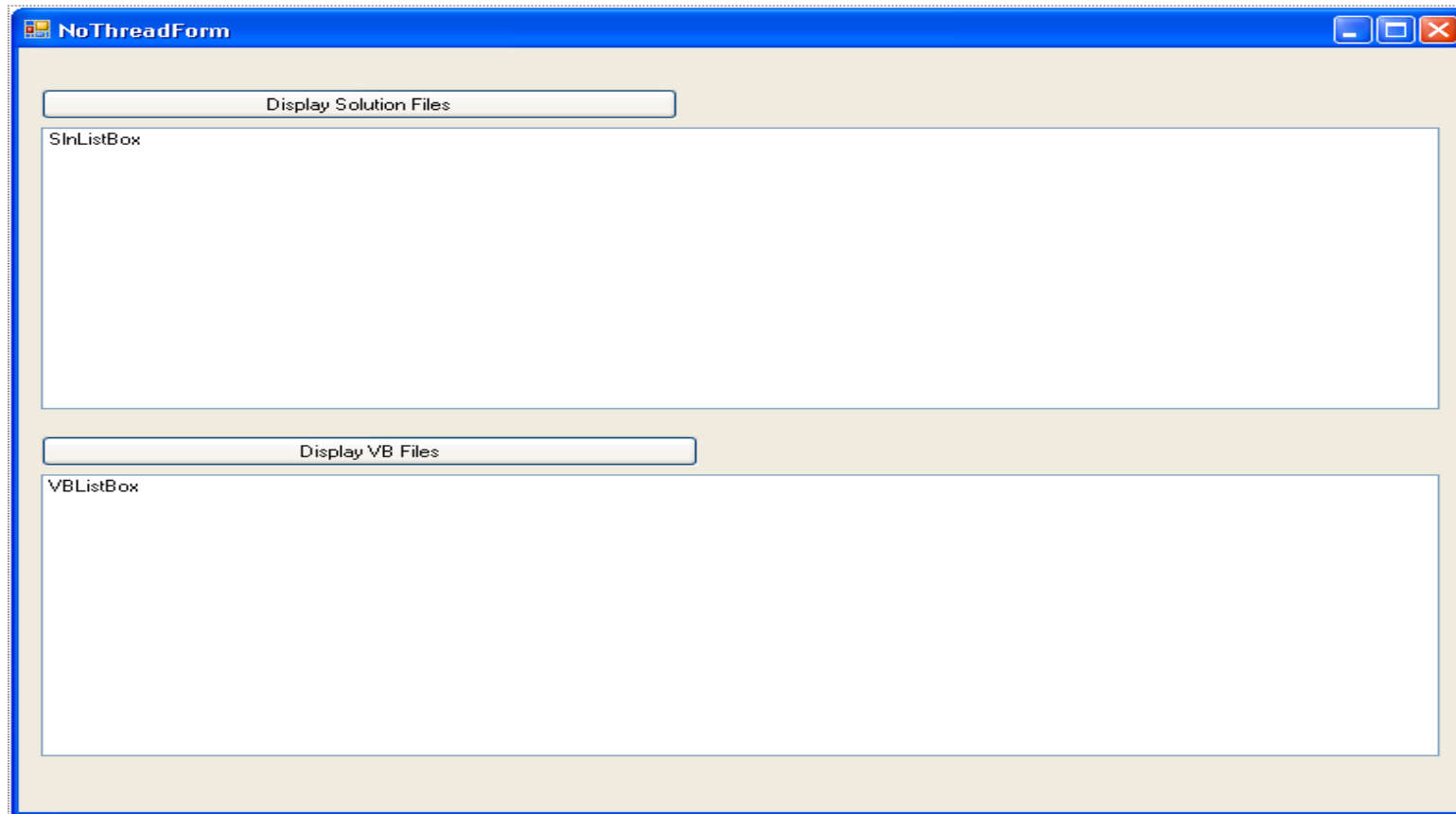


Two Background Workers

Create a new Windows Form Application and name it TwoBackgroundWorkers.

Add the appropriate control to produce a form layout as below:



The screenshot shows a Windows Form titled "NoThreadForm" with a blue title bar and standard Windows window controls (minimize, maximize, close). The form has a light beige background and contains two identical sections. Each section consists of a button at the top and a list box below it. The first section has a button labeled "Display Solution Files" and a list box labeled "SlnListBox". The second section has a button labeled "Display VB Files" and a list box labeled "VBListBox". The list boxes are currently empty.

Note: This form employs multithreading.

Add the following code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Management;
using System.IO;

namespace TwoBackgroundWorkers
{
    public partial class Form1 : Form
    {
        private BackgroundWorker SlnWorker;
        private BackgroundWorker CsWorker;
        public delegate void AddToListBox(string msg);

        public Form1()
        {
            InitializeComponent();
        }

        private void WriteToSlnListBox(string msg)
        {
            if (SolutionListBox.InvokeRequired)
            {
                // It's on a different thread, so use Invoke.
                AddToListBox d = new AddToListBox(WriteToSolutionListBox);
                this.Invoke(d, new object[] { msg });
            }
            else
            {
                //It's on the same thread, no need to use Invoke.
                SolutionListBox.Items.Add(msg);
            }
        }
    }
}
```

```
}

private void FindDrivesSolution(string drv)
{
    FindSolutionFiles(drv);
}

private void FindSolutionFiles(string drv)
{
    try
    {
        foreach (string FileName in Directory.GetFiles(drv, "*.sln"))
        {
            WriteToSlnListBox(FileName);
        }

        foreach (string DirName in Directory.GetDirectories(drv))
        {
            FindSolutionFiles(DirName);
        }
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message.ToString)
    }
}

private void WriteToSolutionListBox(string msg)
{
    SolutionListBox.Items.Add(msg);
}

private void SolutionButton_Click(object sender, EventArgs e)
{
    SlnWorker.RunWorkerAsync();
}

private void WriteToCsListBox(string msg)
```

```

{
    if (CSharpListBox.InvokeRequired)
    {
        // It's on a different thread, so use Invoke.
        AddToListBox d = new AddToListBox(WriteToCSharpListBox);
        this.Invoke(d, new object[] { msg });
    }
    else
    {
        //It's on the same thread, no need to use Invoke.
        CSharpListBox.Items.Add(msg);
    }
}

private void SlnOnWork_DoWork(object sender, DoWorkEventArgs e)
{
    FindDrivesSolution("C:\\");
}

private void CsOnWork_DoWork(object sender, DoWorkEventArgs e)
{
    FindDrivesCSharp("C:\\");
}

private void FindDrivesCSharp(string drv)
{
    FindCSharpFiles(drv);
}

private void FindCSharpFiles(string drv)
{
    try
    {
        foreach (string FileName in Directory.GetFiles(drv, "*.cs"))
        {
            WriteToCsListBox(FileName);
        }
    }
}

```

```

        foreach (string DirName in Directory.GetDirectories(drv))
        {
            FindCSharpFiles(DirName);
        }
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message.ToString)
    }
}

private void WriteToCSharpListBox(string msg)
{
    CSharpListBox.Items.Add(msg);
}

private void CSharpButton_Click(object sender, EventArgs e)
{
    CsWorker.RunWorkerAsync();
}

private void Form1_Load(object sender, EventArgs e)
{
    SlnWorker = new BackgroundWorker();

    SlnWorker.DoWork += new DoWorkEventHandler(SlnOnWork_DoWork);
    SlnWorker.WorkerReportsProgress = true;
    //SlnWorker.ProgressChanged += new ProgressChangedEventHandler(SlnProgressChanged);

    CsWorker = new BackgroundWorker();
    CsWorker.DoWork += new DoWorkEventHandler(CsOnWork_DoWork);
    CsWorker.WorkerReportsProgress = true;
    //VBWorker.ProgressChanged += new ProgressChangedEventHandler(VBProgressChanged);
}
}
}

```