BCS 450 C# Lab – Task and Critical Section

Overview

You will write TWO programs that use C# Tasks.

- The first program will be a simple console application that uses a Task.
- The second program will be a WPF application that does mathematical calculations in parallel.

Part 1 – Console Application project

Create a C# console application in Visual Studio. Name the project Lab-Task-Console.

Part 2 – Use a Simple Task

1. Write a method named Message() that prints a simple one line message.
2. Create an instance of Action and store Message() inside of it.
3. Use a Task in Main to execute the Message() method.
4. Run the program. Does the message appear on screen?

Part 3 – Add a WriteLine to Main

1. Add a Write to the end of Main that says “Main finished”.
2. Run the program. Does the message from Message() appear? If both messages appear (from Main() and Message()) then what order do they appear in?

Part 4– Add a Wait to Main

1. Add a call to Wait right after the call to Start in Main. This will cause the main thread to wait at that line until the Task is finished.
2. Run the program. What is the order of the output statements now?

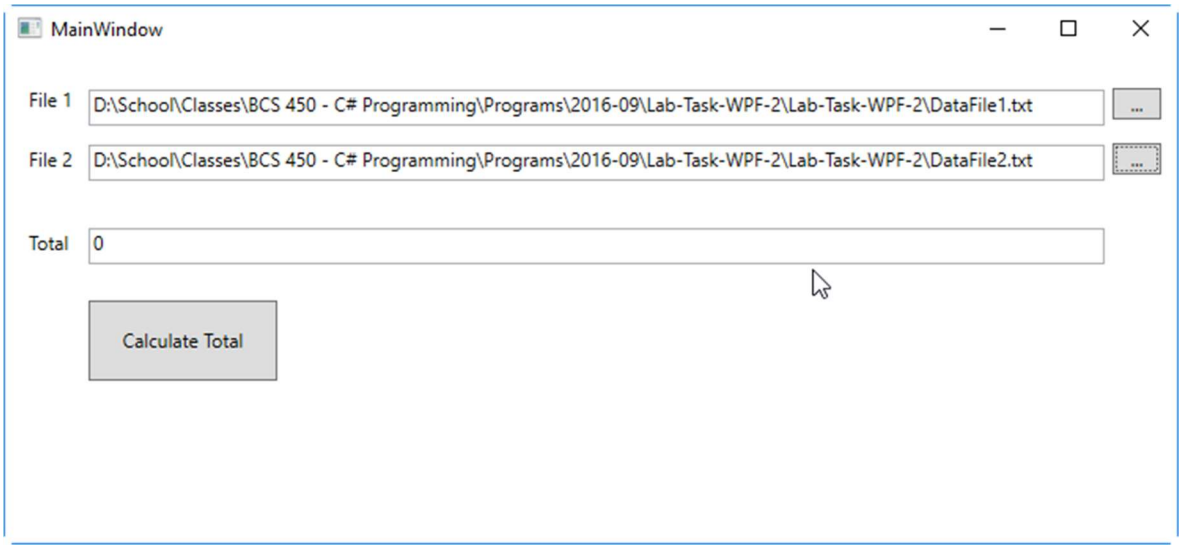
Part 5 – WPF Application project

Create a C# WPF application project in Visual Studio. Name the project Lab-Task-WPF.

This program will add all the numbers stored in two different files. You will use C# Tasks to do the summing of numbers in each file. You will need to define a critical section for the code that updates the total being displayed in the window (see below).

Part 6 – WPF GUI

1. Create the following GUI using WPF:



Important! – Make sure to initialize the Total TextBox with the value "0".

2. Create a method named CalculateFileSum in the Window class.
    a. Should have one parameter to pass in a filename. No return value. Here is the method header:
    void CalculateFileSum(string filename)
    b. The CalculateFileSum method should open the file and add all numbers in the file (read each number and add to a running total). This method will be used by another thread (see details below).
    c. Once the file sum is calculated it should add the file sum to the value in the Total textbox.
    IMPORTANT: This operation needs to use the TextBox Dispatcher and be thread-safe. Read both items i and ii before proceeding.
        i. Dispatcher. The worker thread needs to access the Total TextBox on the UI thread. This operation is not directly allowed. You must use the Dispatcher on the TextBox to get the value that is currently in the TextBox and to set it to a new value (check lecture slides for help).
        ii. Thread safe. The operation that gets or sets the TextBox value needs to be thread-safe. There should be a critical section around the code that accesses the Total TextBox (check lecture slides for help). Use the C# lock statement to create the critical section. Use a delegate with an anonymous method to get the total from the TextBox on the UI thread. For example:
        string textTotal = "";
        Action a = delegate () { textTotal = textBoxTotal.Text; }; // Define action
        textBoxTotal.Dispatcher.Invoke(a); // Use dispatch to run the action
3. Add an event handler for each … button press. Each event handler should display an open file dialog and take the filename the user enters and put it in the appropriate textbox. Use the OpenFileDialog class to do this.
4. Add an event handler for a button press on the Calculate Total button.
    a. Create local string variables in this method to store filename1 and filename2.
    b. When the button is pressed it should create 2 Tasks.

c. One task should call CalculateFileSum passing in **the local variable filename1**.
d. The other task should call CalculateFileSum passing in **the local variable filename2**.
e. The result of each call to CalculateFileSum should be added to the value stored in the Total textbox (as described in the CalculateFileSum specifications).

Part 7 – Test the Program

Create two input files containing numeric data on separate lines and run the program to make sure it works properly.