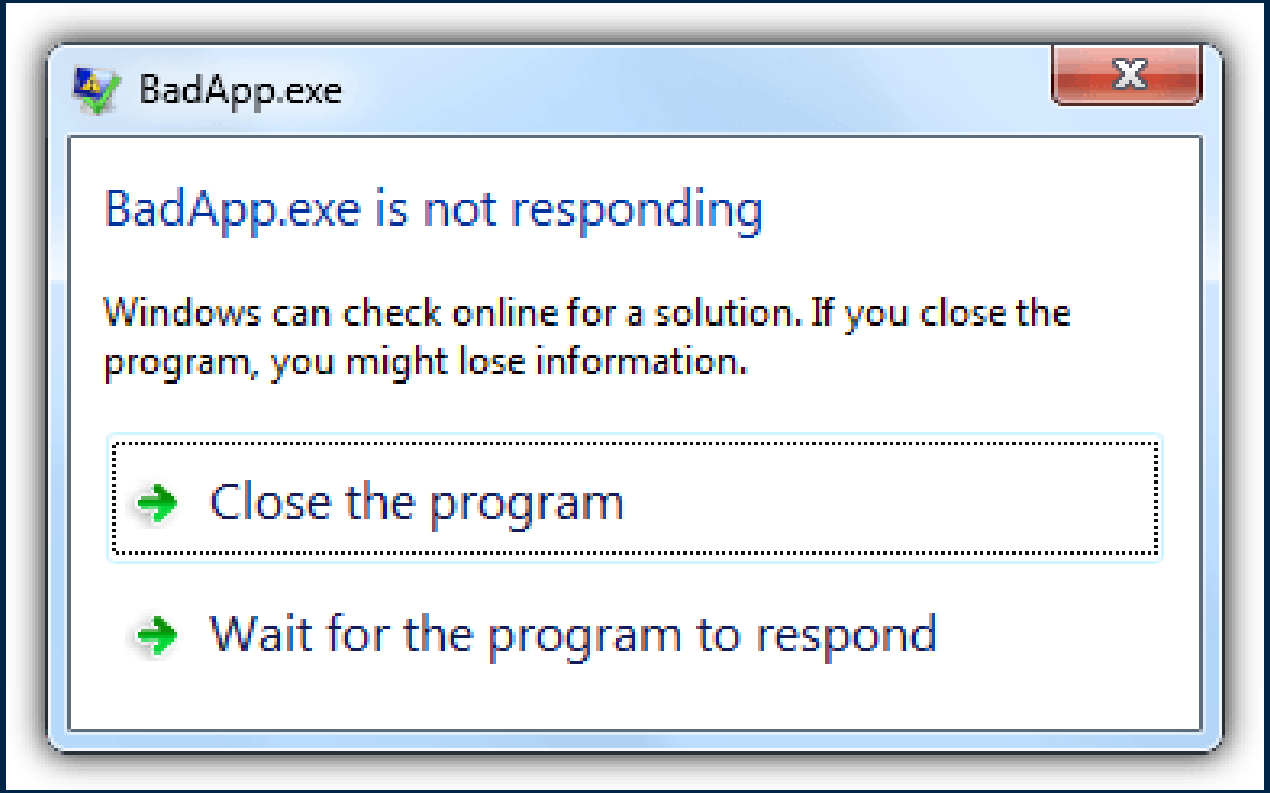
Threading and Windows Forms



MICHAEL LOFT
ML@ASE.AU.DK

Your turn

Solve exercise 1

and when done, jump to the advanced exercises if you want to

Our goals

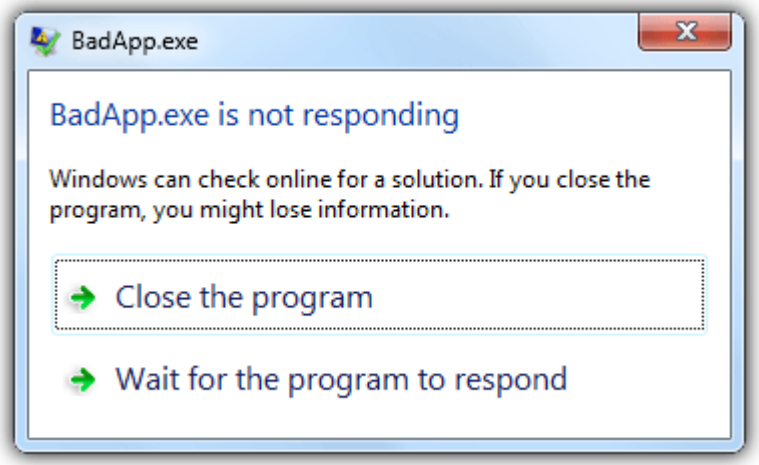
Perform time consuming work and still have a responsive UI.

Inform the user about the state of the work in progress.

React when work is complete

- Inform the user.
- Do something based on the result.

Be able to cancel the work being performed.



Introduction / Agenda

System.ComponentModel.BackgroundWorker

- Initiate a background thread from the UI.
- Do some work and then the thread stops.

System.Threading.Thread

- Create a thread from the main program.
- The thread keeps running, until stopped by some of your code.
- Update the UI with the ”Invoke” methods.

BackgroundWorker

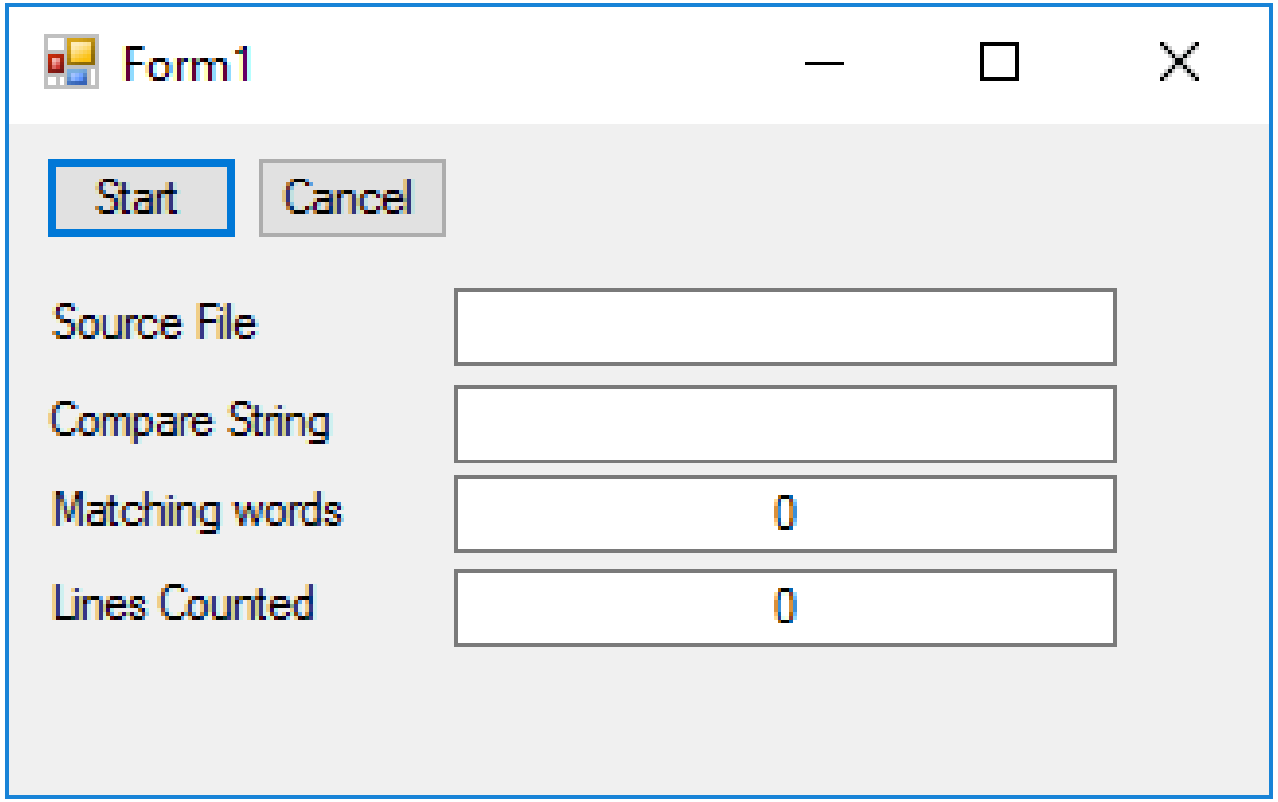
The BackgroundWorker class allows you to **run time-consuming operations** like downloads and database transactions **on a separate, dedicated thread.**

Create a BackgroundWorker and **listen for events** that report the **progress** of your operation and signal when your operation is **finished**.

You can create the BackgroundWorker programmatically or you can drag it onto your form.

Adapted from: https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.backgroundworker?view=netframework-4.7

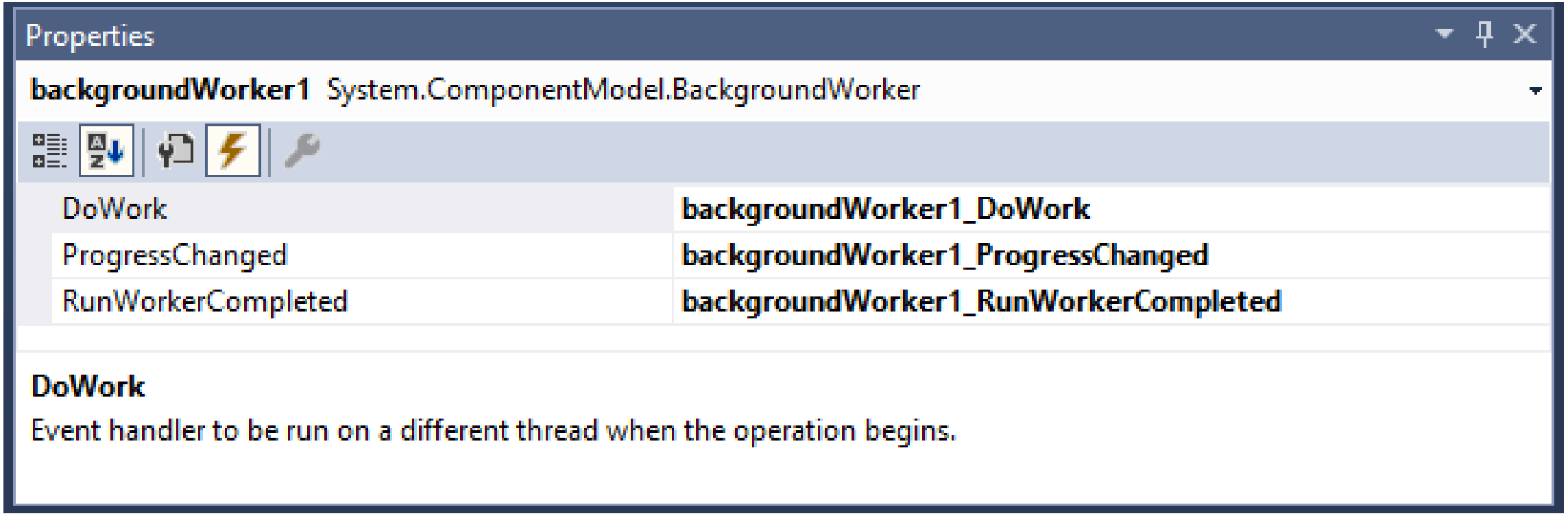
The word counter



https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/threading/walkthrough-multithreading-with-the-backgroundworker-component

Creating a BackgroundWorker

You can create the BackgroundWorker programmatically or drag it onto your form from the **Components** tab of the **Toolbox**. If you create it in the Windows Forms Designer, it will appear in the Component Tray, and its properties will be displayed in the Properties window.



Run time-consuming operations on a separate thread

```
private void backgroundWorker1_DoWork(object sender,
DoWorkEventArgs e)
{
    // This event handler is where the actual work is done.
    // This method runs on the background thread.

    // Get the BackgroundWorker object that raised this event.
    System.ComponentModel.BackgroundWorker worker;
    worker = (System.ComponentModel.BackgroundWorker)sender;

    // Get the Words object and call the main method.
    Words WC = (Words)e.Argument;
    WC.CountWords(worker, e);
}
```

To set up for a background operation, add an event handler for the DoWork event.

Call your time-consuming operation in this event handler.

Run time-consuming operations on a separate thread

```
private void StartThread()
{
    // This method runs on the main thread.
    this.WordsCounted.Text = "0";

    // Initialize the object that the background worker calls.
    Words WC = new Words();
    WC.CompareString = this.CompareString.Text;
    WC.SourceFile = this.SourceFile.Text;

    // Start the asynchronous operation.
    backgroundWorker1.RunWorkerAsync(WC);
}
```

To start the operation, call RunWorkerAsync.

```
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
{
    // This event handler is where the actual work is done.
    // This method runs on the background thread.

    // Get the BackgroundWorker object that raised this event.
    System.ComponentModel.BackgroundWorker worker;
    worker = (System.ComponentModel.BackgroundWorker)sender;

    // Get the Words object and call the main method.
    Words WC = (Words)e.Argument;
    WC.CountWords(worker, e);
}
```

Set as the DoWork event handler on the background worker.

Called on a background thread at some point in time.

Argument to the DoWork event handler.

```
private void StartThread()
{
    // This method runs on the main thread.
    this.WordsCounted.Text = "0";

    // Initialize the object that the background worker calls.
    Words WC = new Words();
    WC.CompareString = this.CompareString.Text;
    WC.SourceFile = this.SourceFile.Text;

    // Start the asynchronous operation.
    backgroundWorker1.RunWorkerAsync(WC);
}
```

Called when the Start button is clicked.

Reporting progress

```
// Object to store the current state, for passing to the caller.
public class CurrentState
{
    public int LinesCounted;
    public int WordsMatched;
}

public void CountWords(
    System.ComponentModel.BackgroundWorker worker,
    System.ComponentModel.DoWorkEventArgs e)
{
  // do some work
  ...
  CurrentState state = new CurrentState();

  state.LinesCounted = LinesCounted;
  state.WordsMatched = WordCount;
  worker.ReportProgress(0, state);

  // do some more work
  ...
}
```

ReportProgress

takes a percentage complete and a custom object as arguments.

Listen for progress reports

```
private void backgroundWorker1_ProgressChanged(object sender,
                                ProgressChangedEventArgs e)
{
    // This method runs on the main thread.
    Words.CurrentState state =
        (Words.CurrentState)e.UserState;
    this.LinesCounted.Text = state.LinesCounted.ToString();
    this.WordsCounted.Text = state.WordsMatched.ToString();
}
```

To receive notifications of progress updates, handle the ProgressChanged event.

Listen for events that signal when your operation is finished

```
private void backgroundWorker1_RunWorkerCompleted(object sender,
                                RunWorkerCompletedEventArgs e)
{
    // This event handler is called when the background thread
finishes.
    // This method runs on the main thread.
    if (e.Error != null)
        MessageBox.Show("Error: " + e.Error.Message);
    else if (e.Cancelled)
        MessageBox.Show("Word counting canceled.");
    else
        MessageBox.Show("Finished counting words.");
}
```

To receive a notification when the operation is completed, handle the RunWorkerCompleted event.

Cancel the operation

```
private void Cancel_Click(object sender, EventArgs e)
{
    // Cancel the asynchronous operation.
    this.backgroundWorker1.CancelAsyncAll();
}
```

The RunWorkerCompleted event handler will be called after cancellation.

Called when the Cancel button is clicked.

Your turn

Solve exercise 2

Your turn

Solve exercise 3, 4, 5 and 6

and when done jump to the advanced exercises if you want to

References and image sources

Images:

BadApp not responding: https://www.raymond.cc/blog/forcefully-close-full-screen-application-or-game-with-superf4/

Computer keyboard: http://stockmedia.cc/computing_technology/slides/DSD_8790.jpg

AARHUS UNIVERSITY