/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Part 5: AsyncDelegates

\*

\* Topic: Use the WaitOne method of the IAsyncResult instance to pause

\* the main thread to wait until the asynchronous call is done.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Be sure to provide an integer value on the command line. To do this:

\*

\* 1) Right-click on the project in Solution Explorer and click

\* Properties.

\*

\* 2) In the Properties window, click the Debug tab.

\*

\* 3) In the "Command line arguments" field, enter a whole number.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.Text**;**

namespaceAsyncDelegatesDemo

**{**

// Add a delegate that the signature we need to call the

// CalculateValue() method.

internaldelegatedoubleDoSomething **(**doubled1**,** doubled2**);**

internalclassComplicatedCalculator

**{**

// Member variable that represent the number of milliseconds

// to pause the thread.

privateint\_millisecondsToPause=0**;**

publicComplicatedCalculator**(**intmillisecondsToPause**)**

**{**

MillisecondsToPause=millisecondsToPause**;**

**}**

// This method represents a task that could potentially run for

// a long period of time.

internaldoubleCalculateValue

**(**doublefirstNumber**,** doublesecondNumber**)**

**{**

doubleanswer=0**;**

// Save the foreground color of the console window.

ConsoleColororiginalcolor=Console.ForegroundColor**;**

// Change the foreground color in the console.

Console.ForegroundColor=ConsoleColor.Red**;**

// Display a message that we're starting the task.

Console.WriteLine**(**"\n\tStarting the calculation task..."**);**

// Set the console color back to the original value.

Console.ForegroundColor=originalcolor**;**

// Pause for a moment.

System.Threading.Thread.Sleep**(**MillisecondsToPause**);**

// Perform the calculation.

answer=Math.Pow**(**firstNumber**,** secondNumber**);**

// Pause for another moment.

System.Threading.Thread.Sleep**(**MillisecondsToPause**);**

// Change the foreground color in the console.

Console.ForegroundColor=ConsoleColor.Red**;**

// Display a message that we're done with the task.

Console.WriteLine**(**"\n\tDone with the calculation task."**);**

// Set the console color back to the original value.

Console.ForegroundColor=originalcolor**;**

returnanswer**;**

**}**

privateintMillisecondsToPause

**{**

get **{** return\_millisecondsToPause**; }**

set

**{**

if **(**value<0**)**

**{**

thrownewArgumentException

**(**"Milliseconds must be greater than or equal to 0."**);**

**}**

\_millisecondsToPause=value**;**

**}**

**}**

**}**

classProgram

**{**

privatestaticintGetMilliseconds**(**strings**)**

**{**

intmilliseconds=0**;**

// If this call fails, milliseconds will be set to zero.

if **(**int.TryParse**(**s**,** outmilliseconds**))**

**{**

// If the user types in a low number, let's assume

// that they entered in the number of seconds and

// convert the value to milliseconds.

if **(**milliseconds<250**)**

**{**

milliseconds\*=1000**;**

**}**

**}**

returnmilliseconds**;**

**}**

staticvoidMain**(**string**[]** args**)**

**{**

try

**{**

// Display a message to show we're in Main().

Console.WriteLine**(**"Starting the program."**);**

// Get the number of milliseconds from the arguments

// passed in from the command line.

intmilliseconds=GetMilliseconds**(**args**[**0**]);**

// Create the ComplicatedCalculator object.

ComplicatedCalculatorcc=

newComplicatedCalculator**(**milliseconds**);**

// Create the delegate object.

DoSomethingmethod=newDoSomething**(**cc.CalculateValue**);**

// Call the delegate asynchronously.

IAsyncResultasynchStatus=

method.BeginInvoke**(**10.4**,** 7.451**,** null**,** null**);**

// Call WaitOne() to wait until the async task is done.

Console.WriteLine("\nWaiting for task to complete.");

// Include a timeout and check to see if it completed in time.

bool inTimeBool = asynchStatus.AsyncWaitHandle.WaitOne(10000);

if (inTimeBool)

{

Console.WriteLine("\nTask has completed.");

// Get the result of the asynchronous call.

double results = method.EndInvoke(asynchStatus);

// Display the results.

Console.WriteLine("\nThe result is: {0}", results);

}

else

{

Console.WriteLine("\nTask did NOT complete in time. There are no results. May need to stop application running using Task Manager.");

}

**}**

catch **(**Exceptione**)**

**{**

Console.WriteLine**(**"\nEXCEPTION: {0}."**,** e.Message**);**

**}**

// Pause so we can look at the console window.

Console.Write**(**"\n\nPress <ENTER> to end: "**);**

Console.ReadLine**();**

**}**

**}**

**}**