Thread and GUI

```
/*
Professional Windows GUI Programming Using C#
by Jay Glynn, Csaba Torok, Richard Conway, Wahid Choudhury,
Zach Greenvoss, Shripad Kulkarni, Neil Whitlow

Publisher: Peer Information
ISBN: 1861007663
*/
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Threading;

namespace Wrox.WindowsGUIProgramming.Chapter9
{
///

/// Summary description for Form1.
///
///
struct _threadstart
{
public ThreadStart threadstart;
public DateTime dt;
}

public class frmWasteTime : System.Windows.Forms.Form
{
private System.Windows.Forms.Button butWasteTime;
private System.Windows.Forms.TextBox textBox1;
///
```

```
/// Required designer variable.
///
private System.ComponentModel.Container components = null;
private System.Windows.Forms.ListBox lbThreads;
private System.Windows.Forms.TextBox txtNoOfThreads;
private System.Windows.Forms.Button btThreadStart;

private System.IAsyncResult m_EndInvoke = null;

public frmWasteTime()
{
//
// Required for Windows Form Designer support
//
InitializeComponent();

//
// TODO: Add any constructor code after InitializeComponent
call
//
this.Show();
//TakeTime();
}

///

/// Clean up any resources being used.
///
protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}
```

```
#region Windows Form Designer generated code
///

/// Required method for Designer support – do not modify
/// the contents of this method with the code editor.
///
private void InitializeComponent()
{
this.butWasteTime = new System.Windows.Forms.Button();
this.textBox1 = new System.Windows.Forms.TextBox();
this.lbThreads = new System.Windows.Forms.ListBox();
this.txtNoOfThreads = new System.Windows.Forms.TextBox();
this.btThreadStart = new System.Windows.Forms.Button();
this.SuspendLayout();
//
// butWasteTime
//
this.butWasteTime.Location = new System.Drawing.Point(88, 32);
this.butWasteTime.Name = “butWasteTime”;
this.butWasteTime.TabIndex = 0;
this.butWasteTime.Text = “Waste Time”;
this.butWasteTime.Click             +=             new
System.EventHandler(this.butWasteTime_Click);
//
// textBox1
//
this.textBox1.Location = new System.Drawing.Point(184, 24);
this.textBox1.Name = “textBox1”;
this.textBox1.TabIndex = 1;
this.textBox1.Text = “”;
//
// lbThreads
//
this.lbThreads.Location = new System.Drawing.Point(8, 64);
this.lbThreads.Name = “lbThreads”;
this.lbThreads.Size = new System.Drawing.Size(280, 277);
this.lbThreads.TabIndex = 2;
```

```
//
// txtNoOfThreads
//
this.txtNoOfThreads.Location = new System.Drawing.Point(8,
352);
this.txtNoOfThreads.Name = “txtNoOfThreads”;
this.txtNoOfThreads.TabIndex = 3;
this.txtNoOfThreads.Text = “”;
//
// btThreadStart
//
this.btThreadStart.Location = new System.Drawing.Point(128,
352);
this.btThreadStart.Name = “btThreadStart”;
this.btThreadStart.Size = new System.Drawing.Size(160, 23);
this.btThreadStart.TabIndex = 4;
this.btThreadStart.Text = “Start”;
this.btThreadStart.Click += new
System.EventHandler(this.btThreadStart_Click);
//
// frmWasteTime
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(296, 390);
this.Controls.AddRange(new System.Windows.Forms.Control[] {
this.btThreadStart,
this.txtNoOfThreads,
this.lbThreads,
this.textBox1,
this.butWasteTime});
this.Name = “frmWasteTime”;
this.Text = “Lock Form”;
this.Paint += new
System.Windows.Forms.PaintEventHandler(this.painting);
this.ResumeLayout(false);

}
```

```
#endregion

///

/// The main entry point for the application.
///
[STAThread]
static void Main()
{
Application.Run(new frmWasteTime());
}

private void TakeTime()
{
int j = 0;
for(int i=0;i<320-j;i++) { System.Diagnostics.Debug.Write(i);
} j++; } private void butWasteTime_Click(object sender,
System.EventArgs e) { IAsyncResult m_EndInvoke2 = null;
ThreadStart threadstart = new ThreadStart(TakeTime);
m_EndInvoke = threadstart.BeginInvoke(new
AsyncCallback(MethodBeginInvoke), String.Copy("test"));
ThreadStart threadstart2 = new ThreadStart(TakeTime);
m_EndInvoke2 = threadstart.BeginInvoke(new
AsyncCallback(MethodBeginInvoke), null);
//m_EndInvoke.AsyncWaitHandle.WaitOne();
threadstart.EndInvoke(m_EndInvoke);
threadstart.EndInvoke(m_EndInvoke2); /*while(true) {
if(m_EndInvoke.IsCompleted) MessageBox.Show("finally!!!");
break; }*/ } private void
MethodBeginInvoke(System.IAsyncResult ar) {
if(ar.CompletedSynchronously) MessageBox.Show("TakeTime()
called synchronously"); else MessageBox.Show("TakeTime()
called asynchronously"); } protected override void
OnPaint(PaintEventArgs pe) {
System.Diagnostics.Debug.Write("Paint event called"); }
private void painting(object sender,
System.Windows.Forms.PaintEventArgs e) {
System.Diagnostics.Debug.Write("Paint event called"); }
```

```
//System.Threading.ThreadStart[] threads; //DateTime[] dateTimeThread; _threadstart[] th; DateTime dtStart; DateTime dtEnd; private void btThreadStart_Click(object sender, System.EventArgs e) { th = new _threadstart[Convert.ToInt32(txtNoOfThreads.Text)]; //threads = (ThreadStart[])Array.CreateInstance(typeof(ThreadStart), Convert.ToInt16(txtNoOfThreads.Text)); //dateTimeThread = (DateTime[])Array.CreateInstance(typeof(DateTime), threads.Length); dtStart = DateTime.Now; for(int i = 0; i < th.Length; i++) { th[i].threadstart = new ThreadStart(TakeTime); th[i].threadstart.BeginInvoke(new AsyncCallback(ThreadFinished), i); th[i].dt = DateTime.Now; } /*while(true) { for(int i = 0; i < threads.Length; i++) { if((threads[i].ThreadState == System.Threading.ThreadState.Suspended) && finished[i].Equals(false)) { //put something into the listbox lbThreads.Items.Insert(i, "Thread "+i+" finished - "+(DateTime.Now.Subtract(dateTimeThread[i]))); finished[i] = true; } Refresh(); } }*/ } void FinalCallback(string blank) { dtEnd = DateTime.Now.Subtract(new TimeSpan(dtStart.Ticks)); lbThreads.Items.Add("Total Thread Time: "+((dtEnd.Minute*60)+dtEnd.Second).ToString()+" seconds"); lbThreads.Items.Add("Average Thread Time: "+(((dtEnd.Minute*60)+dtEnd.Second)/th.Length)+" seconds"); } void UpdateListbox(string listboxText) { lbThreads.Items.Add(listboxText); } delegate void ItemAdd(string item); private void ThreadFinished(IAsyncResult ar) { ItemAdd ia = new ItemAdd(UpdateListbox); ItemAdd ib = new ItemAdd(FinalCallback); int i = (int)ar.AsyncState; string state = "Thread "+(i+1)+" finished - "+(DateTime.Now.Subtract(th[i].dt)).ToString(); //put something into the listbox if(lbThreads.InvokeRequired) { lbThreads.Invoke(ia, new Object[] {state}); } else { lbThreads.Items.Add(state); } if(i==(th.Length-1)) lbThreads.Invoke(ib, new Object[]{""}); } } } [/csharp]
```