onthebeach

Coding Exercise

Attention: Claire & Co.

By

Elisa Sampson

18/07/2017

**Action:**

The below gives details on how I Constructed an object orientated solution for the given case study:

* C# - Visual Studio 2015
* Appropriately annotated code
* Annotated screenshots of the program in action
* Evidence of satisfying all of the requirements in the scenario

**Steps taken:**

Annotated code using C# Visual Studio 2015:   
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Jobs\_App

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

#region Custom Methods = Textboxes with no data inputted

//the below method code informs the user that text boxes are have no data added, empty sequence

private void Emptytextboxes()

{

if (aTXT.Text == String.Empty || bTXT.Text == String.Empty || cTXT.Text == String.Empty || dTXT.Text == String.Empty || eTXT.Text == String.Empty || fTXT.Text == String.Empty)

MessageBox.Show("Error:(no jobs), empty sequence.");

}

#endregion

//The below code for button one displays results when the 1st job is inputted into textbox A

private void oneBTN\_Click(object sender, EventArgs e)

{

Emptytextboxes();

string joblist;

joblist = aTXT.Text;

MessageBox.Show("sequence consisting of a single job " + joblist);

}

//The below code clears all the text boxes once the 'Clear all text boxes' button is pressed

private void clearBTN\_Click(object sender, EventArgs e)

{

//code clears all text boxes

aTXT.Clear();

bTXT.Clear();

cTXT.Clear();

dTXT.Clear();

eTXT.Clear();

fTXT.Clear();

}

private void twoBTN\_Click(object sender, EventArgs e)

{

//The below code for button 2 will display result a a sequence containing all three jobs abc in no significant order.

MessageBox.Show("B: " + bTXT.Text + "C: " + cTXT.Text + "A: " + aTXT.Text);

}

private void threeBTN\_Click(object sender, EventArgs e)

{

// The below code for button 3 will display result in a sequence that positions c before b, containing all three jobs abc.

MessageBox.Show("A: " + aTXT.Text + "C: " + cTXT.Text + "B: " + bTXT.Text);

}

private void fourBTN\_Click(object sender, EventArgs e)

{

//The below code for button 4 will dsiplay result in a sequence that positions f before c, c before b, b before e and a before d containing all six jobs abcdef.

MessageBox.Show("F: " + fTXT.Text + "C: " + cTXT.Text + "B: " + bTXT.Text + "E: " + eTXT.Text + "A: " + aTXT.Text + "D: " + dTXT.Text);

}

private void fiveBTN\_Click(object sender, EventArgs e)

{

//The below code for button 6 result displays an error stating that jobs can’t depend on themselves.

MessageBox.Show("Error: Jobs can’t depend on themselves");

}

private void sixBTN\_Click(object sender, EventArgs e)

{

//The below code for button 6 result displays an error stating that jobs can’t have circular dependencies.

MessageBox.Show("Error: Jobs can’t have circular dependencies.");

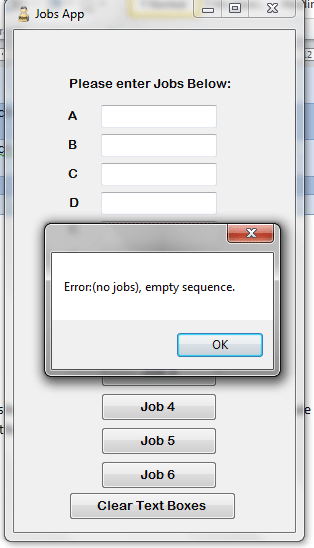
}

}

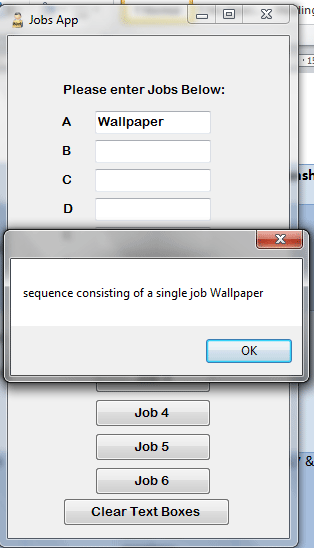
}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Id | Test Purpose | Expected Results | Actual Results | Screenshot |
| 1 | Given you’re passed an empty string (no jobs), the result should be an empty sequence. | Empty sequence result to be displayed | Empty sequence result to be displayed | 1 |
| 2 | Given the following job structure: a => The result should be a sequence consisting of a single job a. | The result should be a sequence consisting of a single job a. | The result should be a sequence consisting of a single job a. | 2 |
| 3 | Given the following job structure: a => b => c => The result should be a sequence containing all three jobs abc in no significant order | The result should be a sequence that positions c before b, containing all three jobs abc. | The result should be a sequence that positions c before b, containing all three jobs abc. | 3 |
| 4 | Given the following job structure: a => b => c c => The result should be a sequence that positions c before b, containing all three jobs abc. | The result should be a sequence that positions c before b, containing all three jobs abc. | The result should be a sequence that positions c before b, containing all three jobs abc. | 4 |
| 5 | Given the following job structure: a => b => c c => f d => a e => b f => The result should be a sequence that positions f before c, c before b, b before e and a before d containing all six jobs abcdef. | The result should be a sequence that positions f before c, c before b, b before e and a before d containing all six jobs abcdef. | The result should be a sequence that positions f before c, c before b, b before e and a before d containing all six jobs abcdef. | 5 |
| 6 | Given the following job structure: a => b => c => c The result should be an error stating that jobs can’t depend on themselves | The result should be an error stating that jobs can’t depend on themselves | The result should be an error stating that jobs can’t depend on themselves | 6 |
| 7 | Given the following job structure: a => b => c c => f d => a e => f => b The result should be an error stating that jobs can’t have circular dependencies | The result should be an error stating that jobs can’t have circular dependencies | The result should be an error stating that jobs can’t have circular dependencies | 7 |
| 8 | Additional Feature:  Clear all textboxes | Clear all textboxes | Clear all textboxes | 8 |

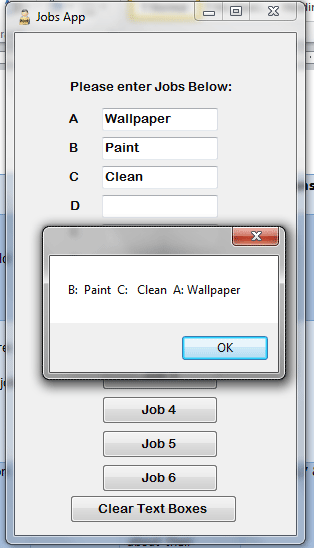
Screenshot 1:



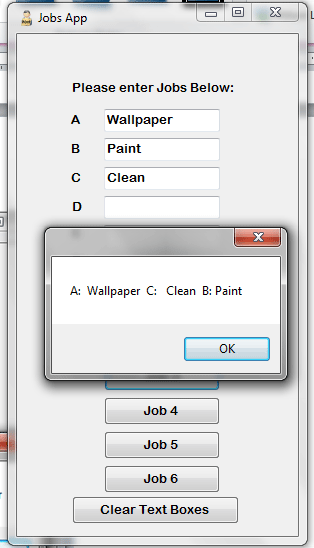
Screenshot 2:



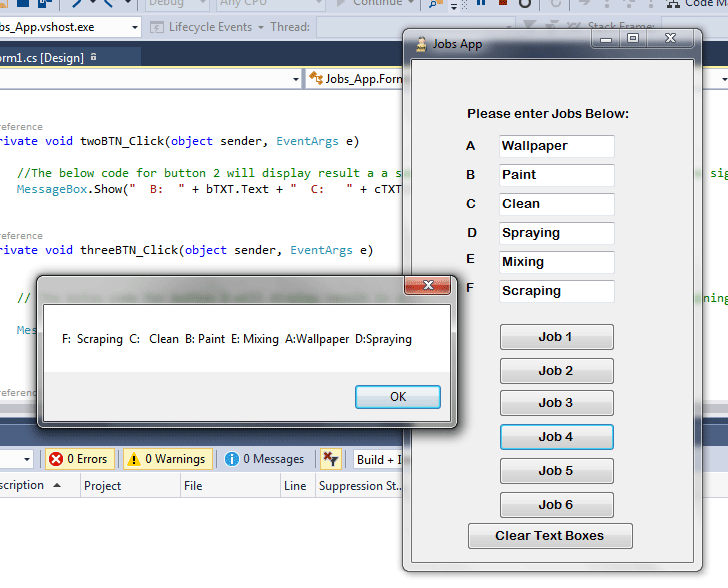
Screenshot 3:



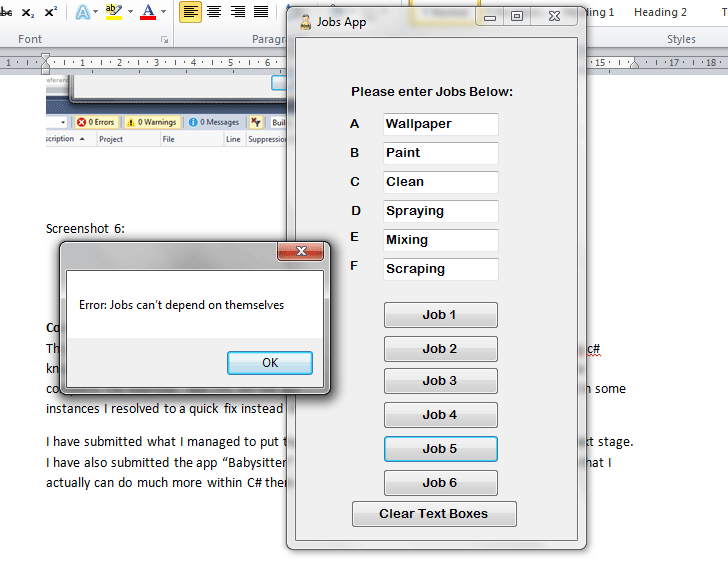
Screenshot 4:



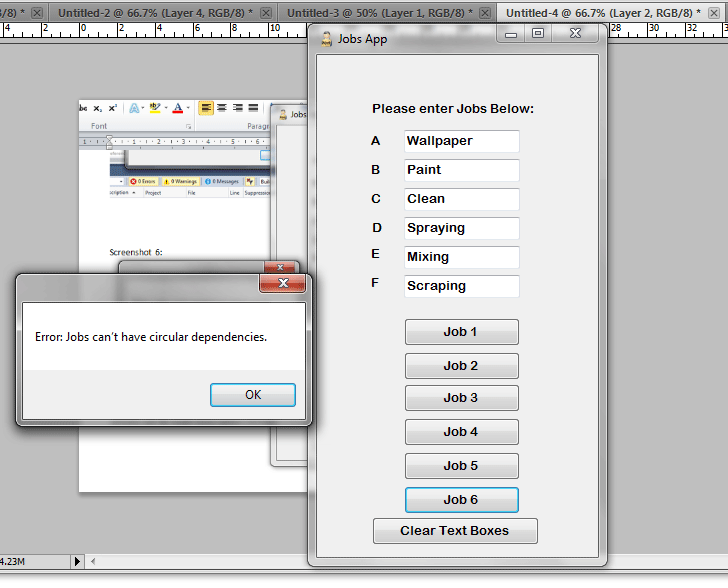
Screenshot 5:



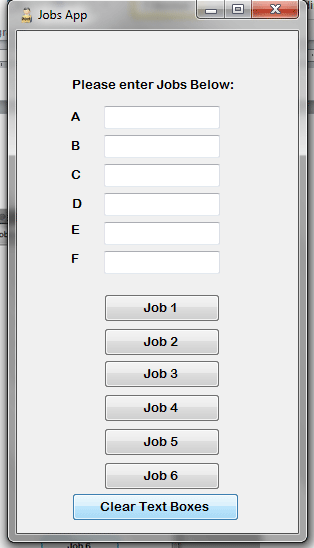
Screenshot 6:



Screenshot 7:



Screenshot 8:



**Conclusion:**   
This coding exercise started out simple but as the app developed it became apparent my my c# knowledge is not as vast as I had hoped. The given time scale should have been sufficient to complete the exercise. But this did not stop me from trying to get the results required and in some instances I resolved to a quick fix instead of using the OOP features like lists & arrays etc.

I have submitted what I managed to put together hoping this will be enough to get to the next stage. I have also submitted the app “Babysitter” I designed as a project, in the hopes you can see that I actually can do much more within C# then what I have produced in this exercise.