**Object Oriented Software Development**

**Practical 2**

Part1 : An introduction to WPF

The aim of this session is to introduce you to Visual Studio and the concepts of forms, controls, properties and events.

**Tasks**

Go back through the lecture notes, use them to assist you in the following tasks.

1. Start Visual Studio (if you are presented with a dialogue allowing a choice of languages select c#). Now create a new C# & WPF project, call the solution “SD2week1” and your project, “hello”. Run the project and make sure that it displays the default blank window.
2. Modify your project to include a button on the form, which displays a message when clicked. Use the following code to display the message

MessageBox.Show("You have clicked on the button");

1. Download and unzip the example projects from Moodle, and place the two folders (myFirstApp and Example2) in the solution folder alongside the “hello” folder. Right click on the solution in the solution explorer window and select add->existing project, select the .csproj files within the new folders to add the projects to your solution.
2. Examine and run both of the two new projects (to select a project to run, right click it in the solution window and select “set as startup project”).

Make sure that you understand how the examples work. To aid your understanding use the Visual Studio debugger to assist you. To add a breakpoint click in the left-hand margin alongside the line where you wish to stop, the location of the red dot below:



Run the application, when the breakpoint is reached you may continue execution line by line using F10.

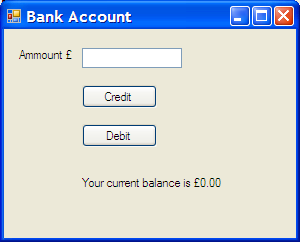
1. All of the controls in the tool box are themselves Objects, they have properties and methods. The ability to use WPF controls is crucial if you are to create Windows applications.

Look at each of the controls in the following table, complete a brief description of the controls’ function and describe 2 properties and an event associated with that control (Use different properties/events for each control). For help use Google to find out more about .Net and the toolbox.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ControlName** | **Description/function** | **Property 1**  **Name and description** | **Property 2**  **Name and description** | **Event**  **Name and trigger** |
| **Button** |  |  |  |  |
| **TextBox** |  |  |  |  |
| **ListBox** |  |  |  |  |
| **RichTextBox** |  |  |  |  |
| **Image** |  |  |  |  |

**Please pick different properties and events for each control.**

1. Create a new C# WPF in your new application create a form that looks like this:



Make sure that you give your controls meaningful names!

Add event handling code so that pressing the Credit button will add an amount to the balance, pressing the Debit button will deduct an amount from the balance.

You will need to store the balance in a variable declared within your class, you should declare it as follows:

private double balance = 0;

You will also need to convert from a string to double in order to add/subtract the text in the text box to/from the balance, the following code demonstrates how to achieve this:

string sNum = "2";

double num = Double.Parse(sNum);

* 1. Modify your bank account programme so that any transaction that would result in a negative balance is not carried out and an error message displayed
  2. Add a ListBox control to your bank account programme and add an entry to the list each time a transaction is carried out e.g. “£3 deposited, balance = £12”.

1. Create a WPF application with the following forms:

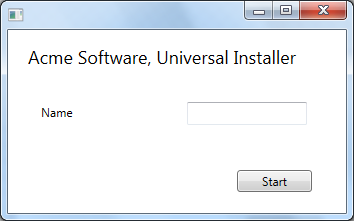


Figure 1 FormStart

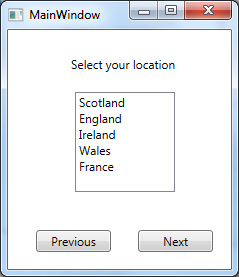


Figure 2 FormLocation

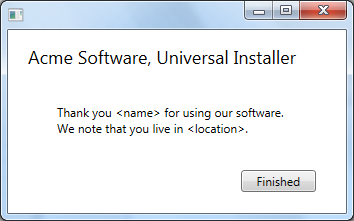
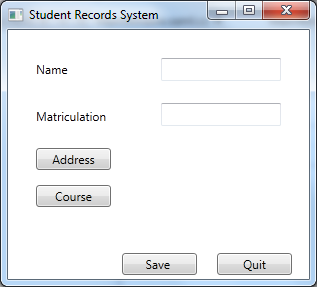


Figure 3 FormEnd

Clicking the Start button should close FormStart and open FormLocaton. Create **one** event handler attached to both of the buttons on FormLocation, when the previous button is clicked the application should close FormLocation and open FormStart, When the next button is clicked the application should close FormLocation and open FormEnd, which should display the details as entered on the previous two forms.



**Part 2: Event Driven Programming**

1. Create a WPF application with the following forms:

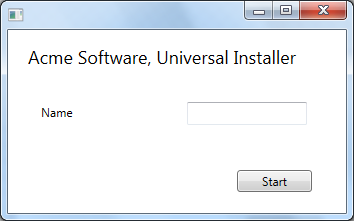


Figure 1 FormStart

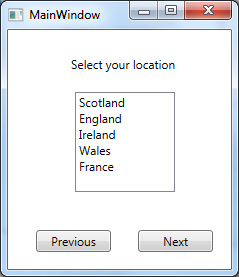


Figure 2 FormLocation

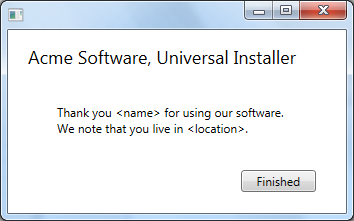
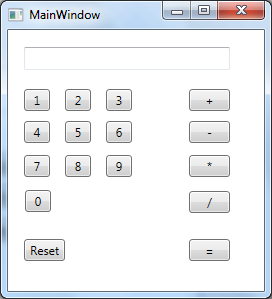


Figure 3 FormEnd

Clicking the Start button should close FormStart and open FormLocaton. Create **one** event handler attached to both of the buttons on FormLocation, when the previous button is clicked the application should close FormLocation and open FormStart, When the next button is clicked the application should close FormLocation and open FormEnd, which should display the details as entered on the previous two forms.

1. Create a reminder application. This application should create a form with a list box and an add button on it. When the add button is clicked the user should be prompted to enter a message (string) and a time in seconds, after the appropriate number of seconds have elapsed then then the message will be displayed in a message box. A list box should be displayed on the form containing the messages to be displayed and the time delay entered. *Hint: use the Timer object to fire events at the appropriate times.* The application must be capable of containing multiple events.
2. Create a simple calculator for children based on a single form. It should have buttons for the digits 0 – 9 and the operations + / \* - and enter.

A text box at the top of the form will show what has been entered.



**One** event handler should be written for all of the buttons. Initially the “=” button will be disabled. The user should use the 0-9 buttons to enter a number which will be shown in the text box. When an operation button (+/\*-) is pressed the digits in the text box will be converted into an integer number and stored in a variable, and the text box cleared to allow a second digit to be entered. The operations buttons will be disabled and the “=” button enabled. The user can then enter a second number and hit equals to show the answer. Pressing reset should reset the form to the initial configuration.