Module 3 Lab Assignment.

In this lab assignment, you will develop a simple user registration application. All form controls and variables that you include in your project MUST have a proper and relevant name. For each improper naming, 0.05 point will be deducted.

If you have any questions, doubts, or if you need any assistance about this assignment, please feel free to post in Module 3's *Slack* channel.

You will use ***TabControl*** to collect various user data through two different pages, **Profile Info** and **Address Info**, and print the entered user data in the **Summary** page.

Grading Schema

Task	**Points**
Building the interface	0.5
Printing the summary page	0.8
Transition among the pages	0.6
Printing the final popup message	0.1

TASK #1: Please add TabControl to the form. The control should have 3 tab-pages as shown in Figure 1. TabControl and its tab-pages should have proper names.

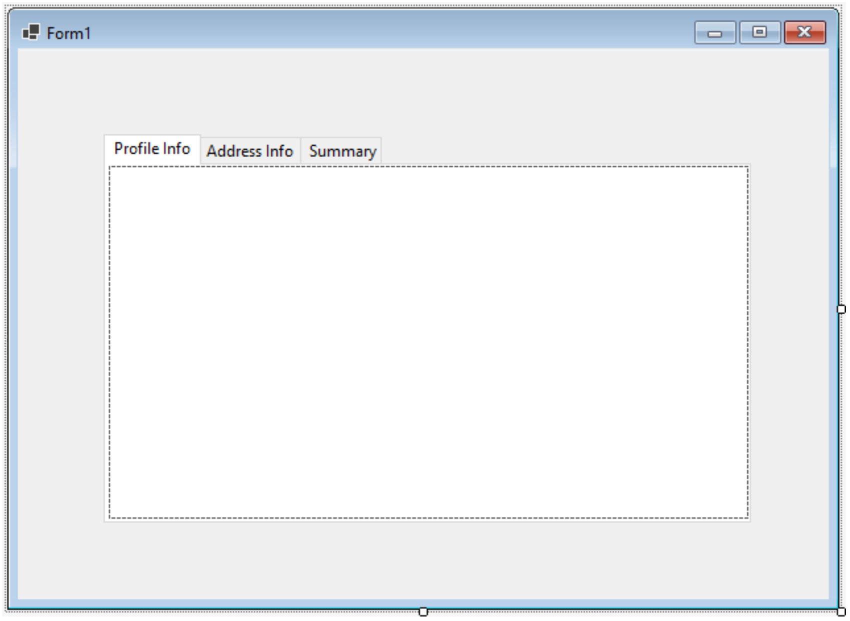


Figure 1. Adding a Tab control to the form.

Next, design the Profile Info page as shown in Figure 2. The page should have first name, last name, and email fields, as well as a Next button. This button, when clicked, should display the Address Info page. To add this functionality, please write the necessary code inside the button click event handler.

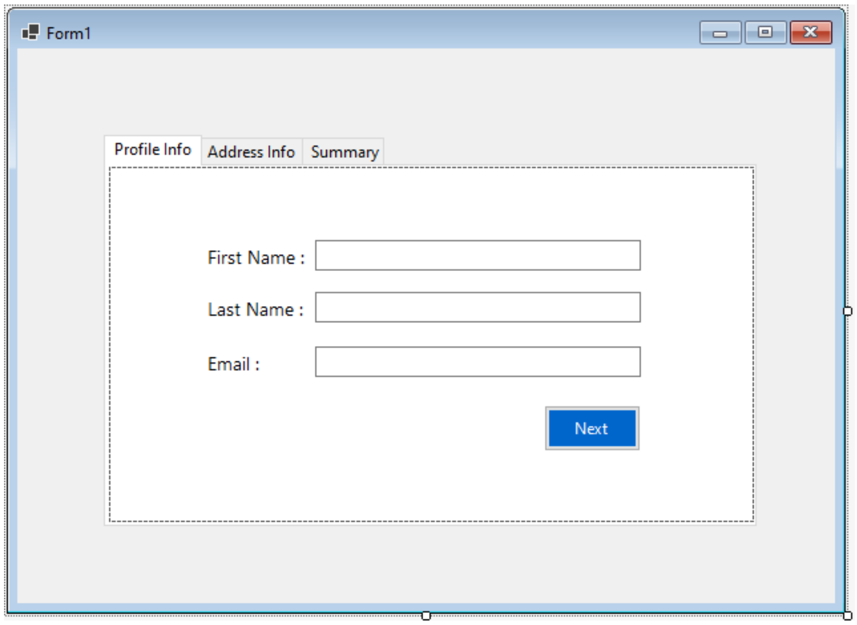


Figure 2. Designing the Profile Info tab page.

Hint: How to switch between tab pages of TabControl in the code? We use the **`SelectedTab`** *property of TabControl. An example would be: myTabControl.SelectedTab = tabPageName;*

At this point you should run your application and test if the Next button functions as desired.

TASK #2: Now, you need to design the Address Info page. Onto this page, please add a ComboBox control, which will be used to allow users to select the address type. Also, add a multiline textbox to allow users to enter the details of their address. The ComboBox should have two items: Home and Business. These items can be added through the ***Items*** property of the ComboBox. See Figure 3 below.

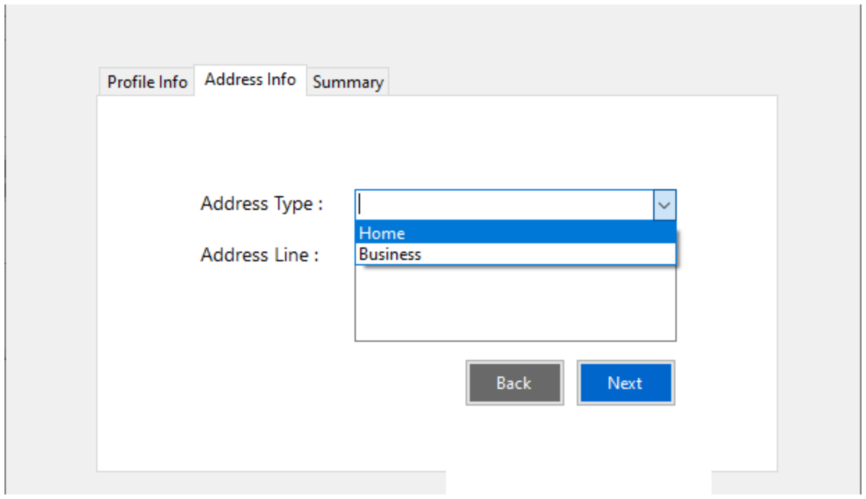


Figure 3. Designing the Address Info tab page.

Next, please add the Back and Next buttons to the Address Info tab page as shown in Figure 5.

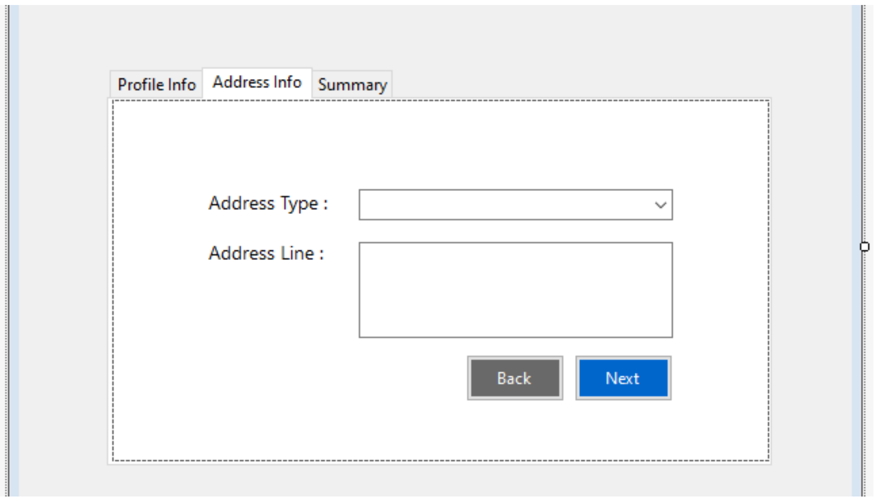


Figure 4. Adding the buttons to the Address Info tab page.

When clicked, the Back button should navigate back to the Profile Info page, and the Next button should navigate to the Summary page. Please write the necessary code for this navigation and test your application.

TASK #3: In this task, you need to build the Summary tab page. First, adding a label control with some default text as shown in Figure 5 below.

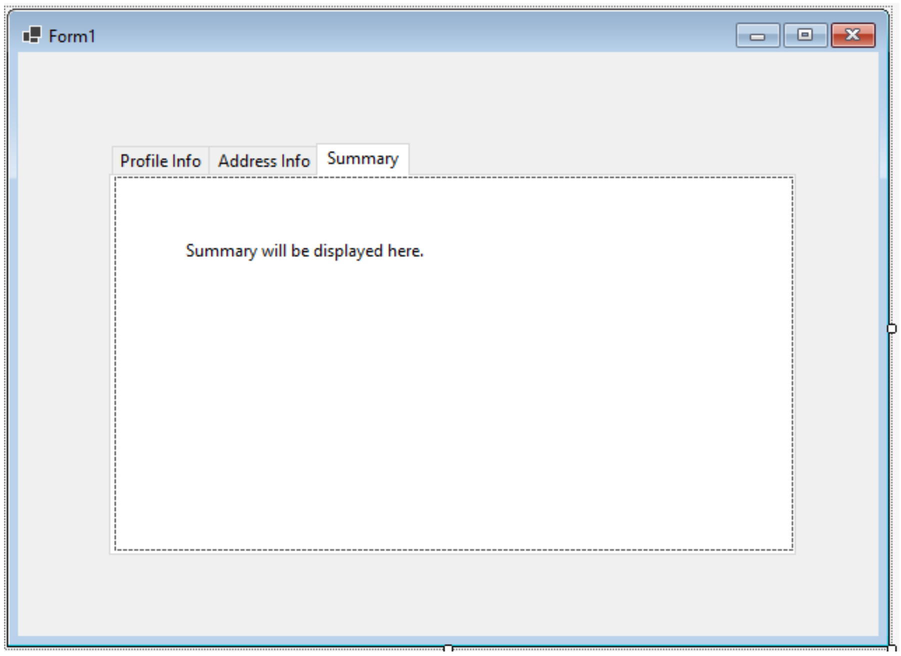


Figure 5. Creating the Summary tab page.

Next add the Back and Complete buttons as shown in Figure 6.

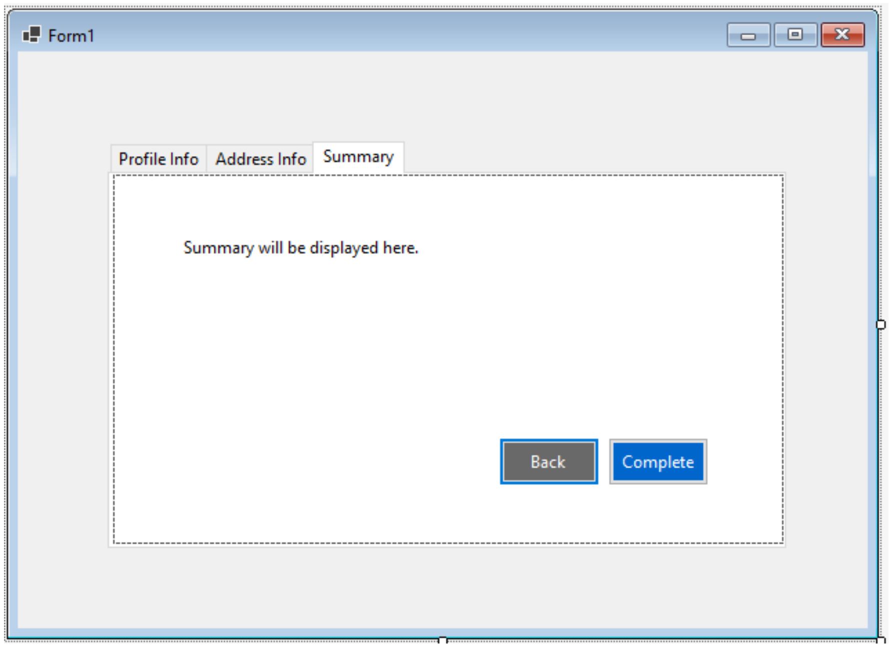


Figure 6. Creating the Summary tab page.

While the Back button should navigate the user back to Address Info page, the Complete button should print a popup message “Registration is completed.” (See Figure 10).

TASK #4: Now, you need to write the necessary code to create the summary text in the Summary tab page. This code should be executed when the user clicks on Next within the Address Info page to transition to the Summary page. Summary text should be displayed using the label control inside the Summary tab page.

To create the summary text, we will use the `StringBuilder` class, which belongs to the `System.Text` namespace. Therefore, to be able to use StringBuilder in your code, you should have the `using System.Text;` statement at the top part of your code file. Add this statement if it is missing.

To use `StringBuilder`, you need to define a `StringBuilder` object using the following structure. Definitions of class objects will be covered in detail in later weeks.

```
StringBuilder summary_sb = new StringBuilder();
```

During this task, you will use two methods of `StringBuilder:` `Append()` and `AppendLine()`. Both methods add strings to the end of the current string value of the `StringBuilder` object. With `Append()` the cursor stays in the same line after adding a string. That means, the next string you will add will be inserted to the end of the current line, and no new lines will be added. For example, the code provided below would print “Erkan Er” in a single line:

```
summary_sb.Append(“Erkan”);
summary_sb.Append("  ");
summary_sb.Append(“Er”);
```

Preferably, you can use the chained syntax in `StringBuilder` expressions. The code below will work the same way as the code shown above.

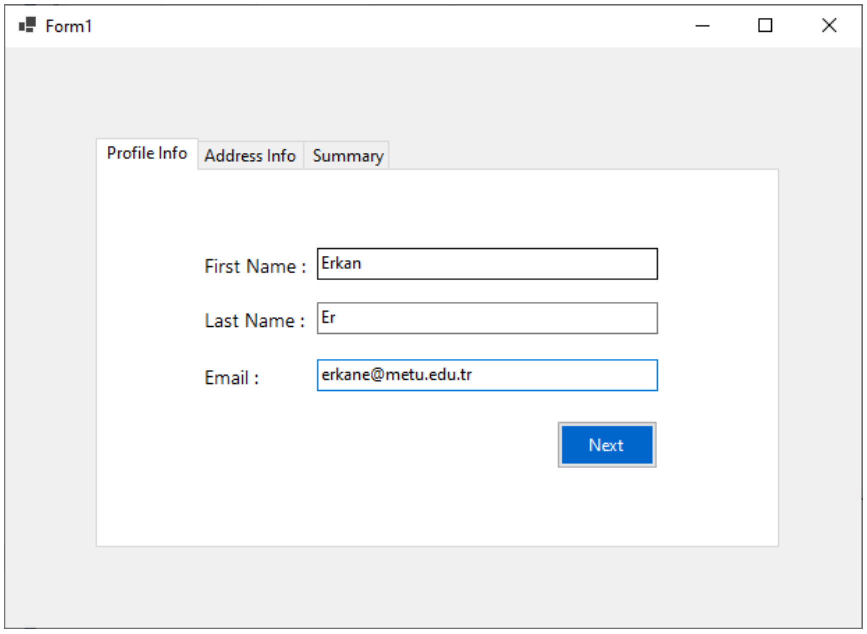
```
summary_sb.Append(“Erkan”).Append("  ").Append(“Er”);
```

On the other hand, after adding a string using `AppendLine()` the curser moves to the next line. Any text you insert afterwards will be appear in the next line. For example, the code provided below would print each string in separate lines.

```
summary_sb.AppendLine(“Erkan”);
summary_sb.AppendLine(“Er”);
```

Please test these codes in your application to see how they work in action.

Given the sample input shown in Figure 7 and Figure 8, the Summary page should produce the output shown in Figure 9.

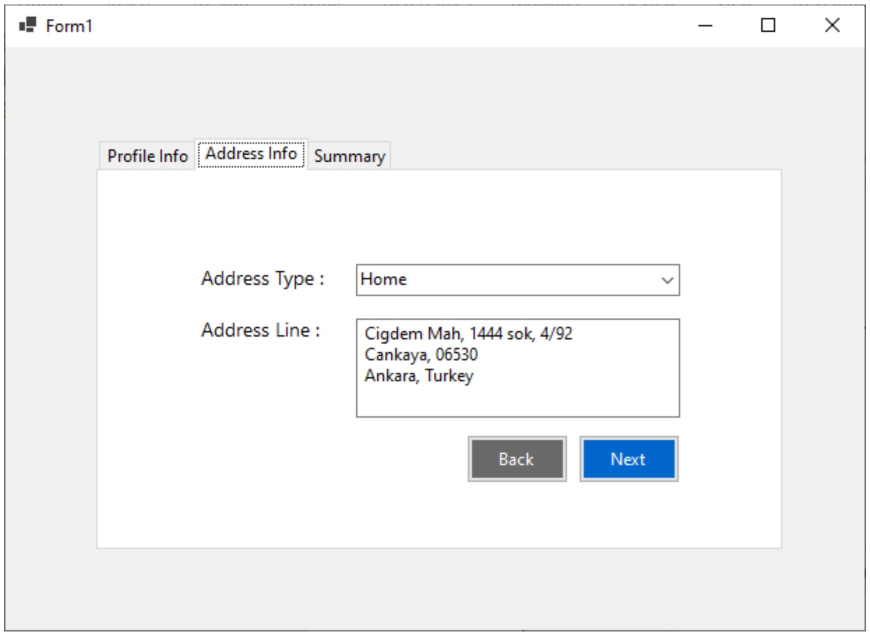


First Name : Erkan

Last Name : Er

Email : erkane@metu.edu.tr

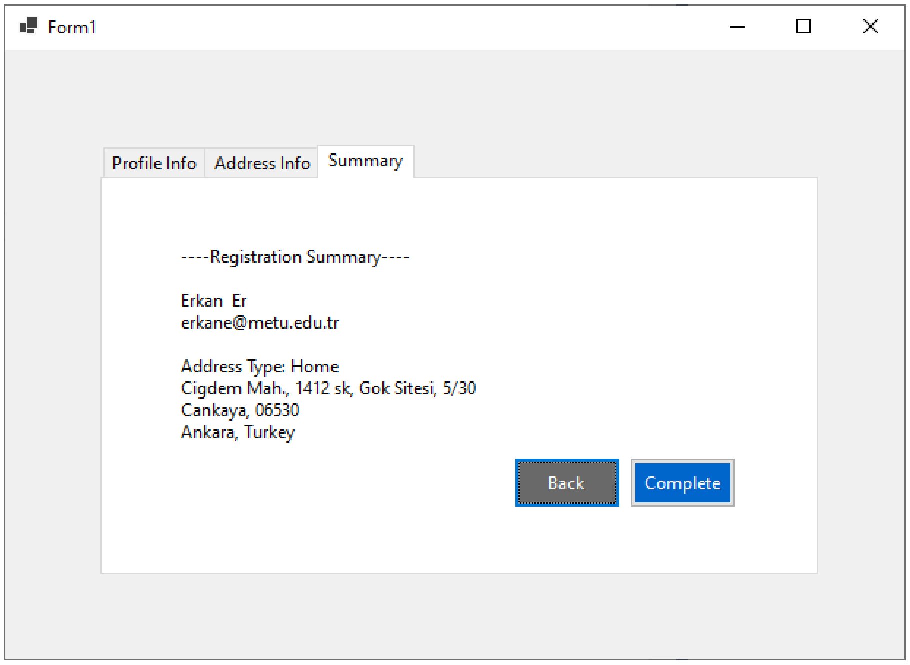
Figure 7. Profile Info page with sample data.



Address Type : Home

Address Line : Cigdem Mah, 1444 sok, 4/92
Cankaya, 06530
Ankara, Turkey

Figure 8. Profile Info page with sample data.

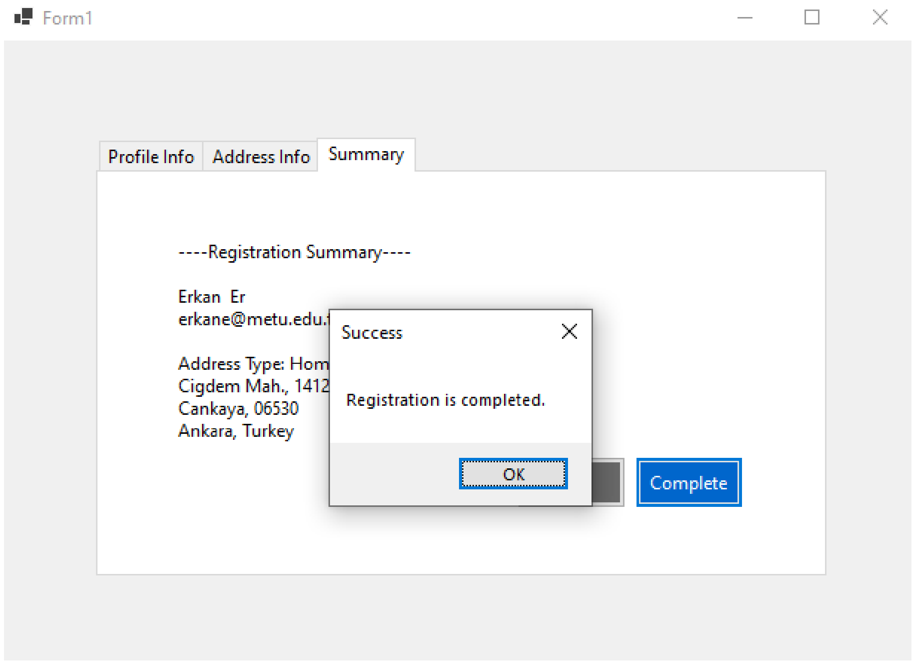


```
----Registration Summary----

Erkan  Er
erkane@metu.edu.tr

Address Type: Home
Cigdem Mah., 1412 sk, Gok Sitesi, 5/30
Cankaya, 06530
Ankara, Turkey
```

Figure 9. Summary page with sample output.



```
----Registration Summary----

Erkan  Er
erkane@metu.edu.

Address Type: Hom
Cigdem Mah., 1412
Cankaya, 06530
Ankara, Turkey
```

Success

Registration is completed.

Figure 10. After clicking on the Complete message.