SE Lab 1 Contents

SE Lab 1: Introduction to Windows Forms with C#

Windows Forms is a graphical (GUI) class library included as a part of Microsoft .NET Framework. It provides a platform to create rich desktop applications with a variety of controls like buttons, text boxes, labels, and more. Learning Windows Forms is a great way to get started with building user interfaces in C#. The following exercises are designed to help beginners understand the basics of Windows Forms, including how to drag and drop controls, handle events, and work with various controls like DataGridView, ComboBox, and more.

Let's break down the process of creating a Windows Form application, understanding the debug folder, and running the .exe file.

I. Creating a Windows Form Application

1. **Open Visual Studio:**
    - Start by opening Visual Studio. If you don't have it installed, you can download it from the Visual Studio website.
2. **Create a New Project:**
    - Go to File > New > Project.
    - Select Windows Forms App (.NET Framework) from the list of templates.
    - Give your project a name and choose a location to save it. Click Create.
3. **Design the Form:**
    - You'll see a blank form in the designer window. This is where you can drag and drop controls from the Toolbox (usually on the left side of the screen).
    - Add controls like Labels, TextBoxes, Buttons, etc., by dragging them onto the form.
    - You can set properties for each control using the Properties window (usually on the right side).
4. **Write Code:**
    - Double-click on a control (e.g., a Button) to create an event handler for it. This will open the code editor where you can write the logic for what happens when the control is interacted with.
    - For example, to handle a button click, you might write:

```
private void button1_Click(object sender, EventArgs e)
{
  label1.Text = "Button Clicked!";
}
```

5. **Run the Application:**
    - Click the Start button (green arrow) at the top of Visual Studio or press F5 to run your application. This will build and run your project, opening the form you designed.

II. Understanding the Debug Folder and .exe Runtime

1. **Build Process:**
    - When you run your application, Visual Studio compiles your code into an executable (.exe) file. This file is what actually runs your application.
    - The compiled files are placed in the bin directory of your project folder. Inside bin, you'll find Debug and Release folders.
2. **Debug Folder:**
    - The Debug folder contains the compiled files when you run your application in Debug mode. This mode is used for testing and debugging your application.
    - Inside the Debug folder, you'll find your .exe file along with other necessary files like .dlls (dynamic link libraries).
3. **Release Folder:**
    - The Release folder contains the compiled files when you build your application in Release mode. This mode is used for the final version of your application that you distribute to users.
    - To build in Release mode, change the build configuration from Debug to Release in the toolbar and then build your project.
4. **Running the .exe File:**
    - Navigate to the Debug or Release folder in your project's bin directory.
    - Double-click the .exe file to run your application outside of Visual Studio. This is how users will run your application once it's distributed.

Summary

- **Creating a Windows Form Application:** Involves designing the form, adding controls, and writing event-handling code.
- **Debug Folder:** Contains the compiled files for testing and debugging.
- **Release Folder:** Contains the compiled files for the final version of your application.
- **Running the .exe File:** Allows you to run your application independently of Visual Studio.

Understanding these steps will help students grasp the full lifecycle of a Windows Forms application, from development to deployment.

III. During Class Exercises

Exercise 1: Creating a Simple Form

This exercise introduces you to the Windows Forms environment. You'll learn how to create a new project, drag a Label control onto the form, and set its properties. This is the foundation for understanding how to build and run a simple Windows Forms application.

Objective: Create a basic Windows Form application.

1. Open Visual Studio and create a new Windows Forms App (.NET Framework) project.
2. Drag a **Label** control from the Toolbox to the form.
3. Set the text of the label to "Hello, World!".
4. Run the application to see the form with the label.

Exercise 2: Adding a Button and Handling Click Events

You'll add interactivity to your form by introducing a Button control. You'll learn how to create an event handler for the button's click event, allowing you to change the text of a Label when the button is clicked. This exercise demonstrates the basics of event-driven programming in Windows Forms.

Objective: Learn to add a button and handle its click event.

1. Open the project from Exercise 1.
2. Drag a **Button** control from the Toolbox to the form.
3. Set the text of the button to "Click Me!".
4. Double-click the button to create a click event handler.
5. In the event handler, add code to change the label's text to "Button Clicked!".
6. Run the application and click the button to see the label text change.

Exercise 3: Using TextBox and MessageBox

This exercise expands on user input by adding a TextBox control. You'll learn how to retrieve text entered by the user and display it in a MessageBox when a button is clicked. This is useful for understanding how to handle user input and provide feedback.

Objective: Learn to use a TextBox and display a message box.

1. Open the project from Exercise 2.
2. Drag a **TextBox** control from the Toolbox to the form.
3. Modify the button click event handler to display the text from the TextBox in a MessageBox.

```
private void button1_Click(object sender, EventArgs e)
{
```

```
    string inputText = textBox1.Text;
    MessageBox.Show(inputText);
}
```

4. Run the application, enter text in the TextBox, and click the button to see the message box.

Exercise 4: Creating a Simple Calculator

In this exercise, you'll create a simple calculator application. You'll use multiple TextBox controls for input, Button controls for operations, and a Label to display results. This exercise helps you understand how to perform calculations and update the UI based on user actions.

Objective: Build a simple calculator with basic arithmetic operations.

1. Create a new Windows Forms project.
2. Add two **TextBox** controls for input numbers.
3. Add four **Button** controls for addition, subtraction, multiplication, and division.
4. Add a **Label** control to display the result.
5. Write event handlers for each button to perform the corresponding arithmetic operation and display the result in the label.

```
private void addButton_Click(object sender, EventArgs e)
{
    double num1 = Convert.ToDouble(textBox1.Text);
    double num2 = Convert.ToDouble(textBox2.Text);
    double result = num1 + num2;
    resultLabel.Text = result.ToString();
}
```

6. **// Repeat similar code for subtraction, multiplication, and division**

Exercise 5: ListBox and ComboBox

You'll learn how to populate and interact with ListBox and ComboBox controls. This exercise demonstrates how to add items to these controls and retrieve the selected item, which is essential for creating forms that require user selection from a list.

Objective: Learn to use ListBox and ComboBox controls.

1. Create a new Windows Forms project.
2. Drag a **ListBox** control and a **ComboBox** control from the Toolbox to the form.

3. Add items to the ListBox and ComboBox in the form's Load event.

```
private void Form1_Load(object sender, EventArgs e)
{
  listBox1.Items.Add("Item 1");
  listBox1.Items.Add("Item 2");
  comboBox1.Items.Add("Option A");
  comboBox1.Items.Add("Option B");
}
```

4. Add a **Button** control to display the selected item from the ListBox and ComboBox in a MessageBox.

```
private void displayButton_Click(object sender, EventArgs e)
{
  string selectedItem = listBox1.SelectedItem.ToString();
  string selectedOption = comboBox1.SelectedItem.ToString();
  MessageBox.Show($"ListBox: {selectedItem}, ComboBox: {selectedOption}");
}
```

Exercise 6: Using DataGridView

The DataGridView control is a powerful tool for displaying tabular data. In this exercise, you'll learn how to set up and populate a DataGridView with sample data. This is fundamental for applications that need to display and manage data in a grid format.

Objective: Learn to use the DataGridView control to display data.

1. Create a new Windows Forms project.
2. Drag a **DataGridView** control from the Toolbox to the form.
3. In the form's Load event, populate the DataGridView with sample data.

```
private void Form1_Load(object sender, EventArgs e)
{
  dataGridView1.ColumnCount = 3;
  dataGridView1.Columns[0].Name = "ID";
  dataGridView1.Columns[1].Name = "Name";
  dataGridView1.Columns[2].Name = "Age";
```

```
        string[] row1 = new string[] { "1", "Ky-Trung Pham", "25" };
        string[] row2 = new string[] { "2", "Thai Ky Trung", "30" };
        dataGridView1.Rows.Add(row1);
        dataGridView1.Rows.Add(row2);
    }
```

4. Run the application to see the DataGridView populated with data.

Exercise 7: Adding Rows to DataGridView

Building on the previous exercise, you'll learn how to add new rows to the DataGridView at runtime. This involves using TextBox controls for input and a Button to trigger the addition of new rows, demonstrating dynamic data manipulation.

Objective: Learn to add rows to the DataGridView dynamically.

1. Open the project from Exercise 6.
2. Add **TextBox** controls for ID, Name, and Age.
3. Add a **Button** control to add a new row to the DataGridView.
4. Write the event handler for the button to add a new row.

```
private void addButton_Click(object sender, EventArgs e)
{
    string[] row = new string[] { textBoxID.Text, textBoxName.Text, textBoxAge.Text };
    dataGridView1.Rows.Add(row);
}
```

5. Run the application, enter data in the TextBoxes, and click the button to add a new row.

Exercise 8: Editing and Deleting Rows in DataGridView

This exercise teaches you how to enable editing and deletion of rows in the DataGridView. You'll learn how to handle user interactions for modifying and removing data, which is crucial for applications that require data management capabilities.

Objective: Learn to edit and delete rows in the DataGridView.

1. Open the project from Exercise 7.
2. Enable editing in the DataGridView by setting

   `dataGridView1.AllowUserToAddRows` and `dataGridView1.AllowUserToDeleteRows` to true.
3. Add a **Button** control to delete the selected row.

4. Write the event handler for the button to delete the selected row.

```
private void deleteButton_Click(object sender, EventArgs e)
    {
      if (dataGridView1.SelectedRows.Count > 0)
      {
        dataGridView1.Rows.RemoveAt(dataGridView1.SelectedRows[0].Index);
      }
    }
```

5. Run the application, select a row, and click the delete button to remove it.

Exercise 9: Using ComboBox with DataGridView

You'll enhance the DataGridView by adding a ComboBox column. This exercise shows how to integrate different types of controls within a DataGridView, providing a more interactive and flexible data presentation.

Objective: Learn to use ComboBox within a DataGridView.

1. Open the project from Exercise 8.
2. Add a **ComboBox** column to the DataGridView.

```
DataGridViewComboBoxColumn comboBoxColumn = new DataGridViewComboBoxColumn();
comboBoxColumn.HeaderText = "Options";
comboBoxColumn.Items.AddRange("Option 1", "Option 2", "Option 3");
dataGridView1.Columns.Add(comboBoxColumn);
```

3. Run the application to see the ComboBox in the DataGridView.

Exercise 10: Using DateTimePicker and ProgressBar

Objective: Learn to use DateTimePicker and ProgressBar controls.

1. Create a new Windows Forms project.
2. Drag a **DateTimePicker** control and a **ProgressBar** control from the Toolbox to the form.
3. Add a **Button** control to start a progress simulation.
4. Write the event handler for the button to update the ProgressBar based on a timer.

```
private void startButton_Click(object sender, EventArgs e)
{
  progressBar1.Value = 0;
```

```
        Timer timer = new Timer();
        timer.Interval = 100; // 100 ms
        timer.Tick += (s, args) =>
        {
            if (progressBar1.Value < 100)
            {
                progressBar1.Value += 10;
            }
            else
            {
                timer.Stop();
                MessageBox.Show("Progress complete!");
            }
        };
        timer.Start();
    }
```

5. Run the application, select a date, and click the button to see the progress bar update.

Notes for summary report

The content in these exercises helps you build basic skills. Follow the instructor's instructions to complete them during your Lab lecture. You do not need to submit these exercises. Instead, prepare a summary report on what you learned with less than A4 page for Exercise 1-10 (no screenshot), and including section IV. HomeWorks with following requirements:

Requirements for Exercises (exercise 11-20)

Copyright Label:

- Each exercise should include a label on the left or right side with the text: “Developed by [Your Full Name]”.
- For example: “Developed by Pham Thai Ky Trung”.

Screenshot of Output:

- After completing each exercise, run the application (press F5) and take a screenshot of the output.
- Include the screenshot in your summary report.

IV. HomeWorks

Some more advanced exercises for Windows Forms controls. If you can finish III. During Class Exercises then you can continue with the following exercises.

These exercises (11->20) are designed to help students understand and utilize advanced Windows Forms controls in C#. Each exercise focuses on a specific control, providing practical examples and explanations to enhance learning.

1. **TreeView Control:** Learn to display hierarchical data using the TreeView control, often used for file systems or organizational structures.
2. **ListView Control:** Understand how to display a list of items with different views (details, large icons, small icons, list) using the ListView control.
3. **TabControl:** Create tabbed interfaces with the TabControl, allowing different content and controls on each tab.
4. **MenuStrip:** Learn to create menus for your application using the MenuStrip control, organizing commands and options.
5. **ContextMenuStrip:** Implement context menus that appear on right-click using the ContextMenuStrip control.
6. **ToolStrip:** Create toolbars with buttons and other items using the ToolStrip control, enhancing user interaction.
7. **StatusStrip:** Display status information at the bottom of your form using the StatusStrip control.
8. **SplitContainer:** Create resizable panels with the SplitContainer control, useful for complex layouts.
9. **FlowLayoutPanel:** Arrange controls in a horizontal or vertical flow using the FlowLayoutPanel control, which automatically wraps as needed.
10. **TableLayoutPanel:** Arrange controls in a grid layout with the TableLayoutPanel control, providing precise control over the layout.

Here you go:

Exercise 11: Using TreeView Control

The TreeView control is used to display a hierarchical collection of labeled items, often used for file systems or organizational structures.

Objective: Learn to use the TreeView control to display hierarchical dataCreate a new Windows Forms project.

1. Drag a **TreeView** control from the Toolbox to the form.
2. Populate the TreeView with nodes in the form's Load event.

```
private void Form1_Load(object sender, EventArgs e)
{
  TreeNode rootNode = new TreeNode("Root");
  TreeNode childNode1 = new TreeNode("Child 1");
  TreeNode childNode2 = new TreeNode("Child 2");
  rootNode.Nodes.Add(childNode1);
  rootNode.Nodes.Add(childNode2);
  treeView1.Nodes.Add(rootNode);
}
```

3. Run the application to see the hierarchical structure.

Exercise 12: Using ListView Control

The ListView control can display items in various views such as details, large icons, small icons, and list.

Objective: Learn to use the ListView control to display a list of items with different views.

1. Create a new Windows Forms project.
2. Drag a **ListView** control from the Toolbox to the form.
3. Set the View property of the ListView to Details.
4. Add columns and items to the ListView in the form's Load event.

```
private void Form1_Load(object sender, EventArgs e)
{
  listView1.View = View.Details;
  listView1.Columns.Add("Column 1", -2, HorizontalAlignment.Left);
  listView1.Columns.Add("Column 2", -2, HorizontalAlignment.Left);

  ListViewItem item1 = new ListViewItem("Item 1");
  item1.SubItems.Add("Subitem 1");
  ListViewItem item2 = new ListViewItem("Item 2");
  item2.SubItems.Add("Subitem 2");
```

```
    listView1.Items.Add(item1);
    listView1.Items.Add(item2);
}
```

5. Run the application to see the ListView in details view.

Exercise 13: Using TabControl

The TabControl allows you to create a tabbed interface, where each tab can contain different controls and content.

Objective: Learn to use the TabControl to create tabbed interfaces.

1. Create a new Windows Forms project.
2. Drag a **TabControl** from the Toolbox to the form.
3. Add tabs and controls to each tab.

```
private void Form1_Load(object sender, EventArgs e)
{
    TabPage tabPage1 = new TabPage("Tab 1");
    TabPage tabPage2 = new TabPage("Tab 2");
    Label label1 = new Label() { Text = "Content of Tab 1", Location = new Point(20, 20) };
    Label label2 = new Label() { Text = "Content of Tab 2", Location = new Point(20, 20) };
    tabPage1.Controls.Add(label1);
    tabPage2.Controls.Add(label2);
    tabControl1.TabPages.Add(tabPage1);
    tabControl1.TabPages.Add(tabPage2);
}
```

4. Run the application to see the tabbed interface.

Exercise 14: Using MenuStrip

The MenuStrip control allows you to create menus for your application, providing a way to organize commands and options

Objective: Learn to use the MenuStrip control to create menus.

1. Create a new Windows Forms project.
2. Drag a **MenuStrip** control from the Toolbox to the form.

3. Add menu items and sub-items.

```
private void Form1_Load(object sender, EventArgs e)
{
  ToolStripMenuItem fileMenu = new ToolStripMenuItem("File");
  ToolStripMenuItem openItem = new ToolStripMenuItem("Open");
  ToolStripMenuItem saveItem = new ToolStripMenuItem("Save");
  fileMenu.DropDownItems.Add(openItem);
  fileMenu.DropDownItems.Add(saveItem);
  menuStrip1.Items.Add(fileMenu);
}
```

4. Run the application to see the menu.

Exercise 15: Using ContextMenuStrip

The ContextMenuStrip control allows you to create context menus that appear when the user right-clicks on a control.

Objective: Learn to use the ContextMenuStrip control to create context menus.

1. Create a new Windows Forms project.
2. Drag a **ContextMenuStrip** control from the Toolbox to the form.
3. Add context menu items.

```
private void Form1_Load(object sender, EventArgs e)
{
  ToolStripMenuItem cutItem = new ToolStripMenuItem("Cut");
  ToolStripMenuItem copyItem = new ToolStripMenuItem("Copy");
  ToolStripMenuItem pasteItem = new ToolStripMenuItem("Paste");
  contextMenuStrip1.Items.Add(cutItem);
  contextMenuStrip1.Items.Add(copyItem);
  contextMenuStrip1.Items.Add(pasteItem);
  textBox1.ContextMenuStrip = contextMenuStrip1;
}
```

4. Run the application, right-click on the TextBox to see the context menu.

Exercise 16: Using ToolStrip

The ToolStrip control allows you to create toolbars with buttons, drop-downs, and other controls.

Objective: Learn to use the ToolStrip control to create toolbars.

1. Create a new Windows Forms project.
2. Drag a **ToolStrip** control from the Toolbox to the form.
3. Add buttons and other items to the ToolStrip.

```
private void Form1_Load(object sender, EventArgs e)
{
  ToolStripButton newButton = new ToolStripButton("New");
  ToolStripButton openButton = new ToolStripButton("Open");
  ToolStripButton saveButton = new ToolStripButton("Save");
  toolStrip1.Items.Add(newButton);
  toolStrip1.Items.Add(openButton);
  toolStrip1.Items.Add(saveButton);
}
```

4. Run the application to see the toolbar.

Exercise 17: Using StatusStrip

The StatusStrip control allows you to display status information at the bottom of your form

Objective: Learn to use the StatusStrip control to display status information.

1. Create a new Windows Forms project.
2. Drag a **StatusStrip** control from the Toolbox to the form.
3. Add status labels and other items to the StatusStrip.

```
private void Form1_Load(object sender, EventArgs e)
{
  ToolStripStatusLabel statusLabel = new ToolStripStatusLabel("Ready");
  statusStrip1.Items.Add(statusLabel);
}
```

4. Run the application to see the status strip.

Exercise 18: Using SplitContainer

The SplitContainer control allows you to create a form with resizable panels, useful for creating complex layouts

Objective: Learn to use the SplitContainer control to create resizable panels.

1. Create a new Windows Forms project.
2. Drag a **SplitContainer** control from the Toolbox to the form.
3. Add controls to each panel of the SplitContainer.

```
private void Form1_Load(object sender, EventArgs e)
{
  Label label1 = new Label() { Text = "Panel 1", Dock = DockStyle.Fill };
  Label label2 = new Label() { Text = "Panel 2", Dock = DockStyle.Fill };

  splitContainer1.Panel1.Controls.Add(label1);
  splitContainer1.Panel2.Controls.Add(label2);
}
```

4. Run the application to see the resizable panels.

Exercise 19: Using FlowLayoutPanel

Objective: Learn to use the FlowLayoutPanel control to arrange controls in a flow layout. **Explanation:** The FlowLayoutPanel control arranges its child controls in a horizontal or vertical flow, automatically wrapping as needed.

1. Create a new Windows Forms project.
2. Drag a **FlowLayoutPanel** control from the Toolbox to the form.
3. Add controls to the FlowLayoutPanel.
4. private void Form1_Load(object sender, EventArgs e)

```
{
  Button button1 = new Button() { Text = "Button 1" };
  Button button2 = new Button() { Text = "Button 2" };
  Button button3 = new Button() { Text = "Button 3" };
  flowLayoutPanel1.Controls.Add(button1);
  flowLayoutPanel1.Controls.Add(button2);
```

```
        flowLayoutPanel1.Controls.Add(button3);
    }
```

5. Run the application to see the flow layout.

Exercise 20: Using TableLayoutPanel

The TableLayoutPanel control arranges its child controls in a grid of rows and columns, providing precise control over layout.

Objective: Learn to use the TableLayoutPanel control to arrange controls in a grid layout.

1. Create a new Windows Forms project.
2. Drag a **TableLayoutPanel** control from the Toolbox to the form.
3. Add controls to the TableLayoutPanel and configure rows and columns.

```
private void Form1_Load(object sender, EventArgs e)
    {
        // Set up the TableLayoutPanel
        tableLayoutPanel1.ColumnCount = 2;
        tableLayoutPanel1.RowCount = 2;
        tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 50F));
        tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 50F));
        tableLayoutPanel1.RowStyles.Add(new RowStyle(SizeType.Percent, 50F));
        tableLayoutPanel1.RowStyles.Add(new RowStyle(SizeType.Percent, 50F));
        // Add controls to the TableLayoutPanel
        Button button1 = new Button() { Text = "Button 1" };
        Button button2 = new Button() { Text = "Button 2" };
        Button button3 = new Button() { Text = "Button 3" };
        Button button4 = new Button() { Text = "Button 4" };
        tableLayoutPanel1.Controls.Add(button1, 0, 0); // Add button1 to column 0, row 0
        tableLayoutPanel1.Controls.Add(button2, 1, 0); // Add button2 to column 1, row 0
        tableLayoutPanel1.Controls.Add(button3, 0, 1); // Add button3 to column 0, row 1
        tableLayoutPanel1.Controls.Add(button4, 1, 1); // Add button4 to column 1, row 1
    }
```

4. **Run the Application:**
    - Press F5 or click the Start button to run the application.
    - You should see a form with a TableLayoutPanel containing four buttons arranged in a 2x2 grid.

Congratulations. You are now a Windows Form Controls specialist as these exercises cover a range of advanced controls, helping you build more sophisticated and user-friendly Windows Forms applications.

Hope you enjoy Lab 1. If you need further details or additional exercises, feel free to ask:

Email: tg_phamthaikytrung@tdtu.edu.vn