# IT 230 Coding Activity Submission Template

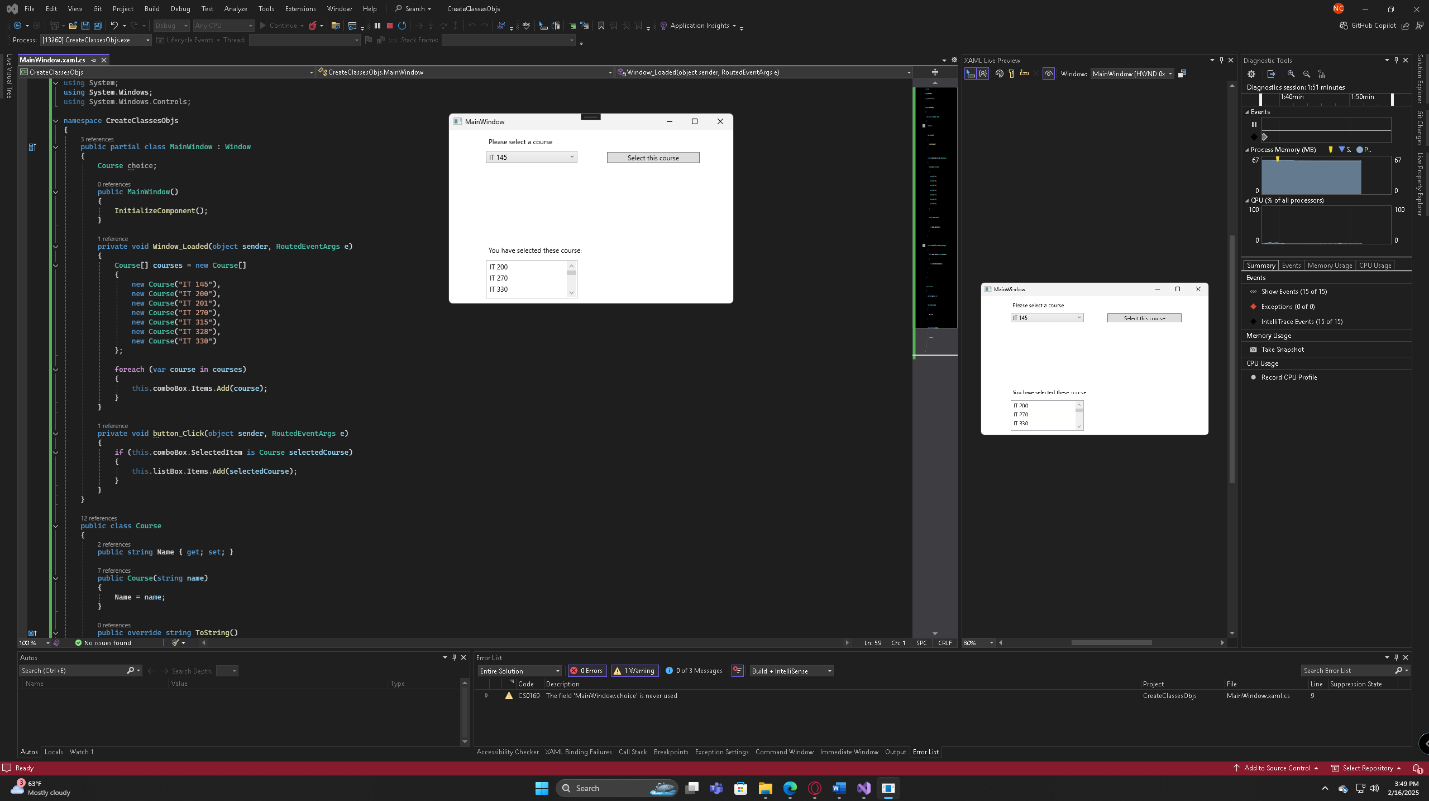
Submit your work on the coding activities for Modules One, Two, Three, Four, and Six in this document. In addition to this document, you should submit a ZIP file containing all your Visual Studio project files and source code that can be run in Visual Studio on a different computer.

For each coding activity, complete the following steps:

* Download and rename this document to meet the file naming conventions requested in the assignment instructions.
* Fill in the required information below by replacing the bracketed text with the relevant information.
* Submit this document and your ZIP file for grading and feedback. Your ZIP file should follow the same naming conventions.

Document your work in the coding activity by completing each of the following items:

1. Provide a screenshot of the output that resulted from running your program successfully in Visual Studio. See the coding assignment instructions for an example of what should be included in the screenshot. Your screenshot must include the following elements:
   1. Your last name as the first printed text on the screen
   2. Verification that the program is fully functioning and data results are accurate for the given problem



1. Copy and paste the source code text you wrote for this assignment from the \*.cs file into the space below. Only providing the \*.cs files or a screenshot does not meet the requirements for this part of the assignment. Code should be logically organized. It should also follow proper syntax and conventions noted in the Coding Activity Guidelines and Rubric.

using System;

using System.Windows;

using System.Windows.Controls;

namespace CreateClassesObjs

{

public partial class MainWindow : Window

{

Course choice;

public MainWindow()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

Course[] courses = new Course[]

{

new Course("IT 145"),

new Course("IT 200"),

new Course("IT 201"),

new Course("IT 270"),

new Course("IT 315"),

new Course("IT 328"),

new Course("IT 330")

};

foreach (var course in courses)

{

this.comboBox.Items.Add(course);

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

if (this.comboBox.SelectedItem is Course selectedCourse)

{

this.listBox.Items.Add(selectedCourse);

}

}

}

public class Course

{

public string Name { get; set; }

public Course(string name)

{

Name = name;

}

public override string ToString()

{

return Name;

}

}

}

1. Show that you understand the task by explaining the design of your program in the space below. Include the process and steps you took to write your code. Explain how you arrived at the solution to the problem and completed the activity.

The program is a WPF application designed to allow users to select a course from a drop-down list (ComboBox) and add it to a ListBox. The primary goal is to manage and display course objects dynamically. To achieve this, I first created a Course class with a Name property and an overridden ToString() method to ensure that the ComboBox displays the course name correctly instead of the default object reference.

Next, I structured the program by initializing an array of Course objects, each representing a different course, and populating the ComboBox using a loop. This approach improved code efficiency compared to manually creating multiple course variables. When the window loads, the ComboBox is filled with course options, allowing the user to select.

To handle user interaction, I implemented an event for the button click. When the button is clicked, the program retrieves the selected course from the ComboBox and adds it to the ListBox. A null check ensures that an item is selected before attempting to add it, preventing potential errors.

Throughout the process, I focused on writing clean, maintainable code while following object-oriented principles. The final implementation successfully allows users to select and add courses dynamically, demonstrating an understanding of object creation, UI component manipulation, and event handling in C#.

1. Reflect on your learning experience and what you learned from completing the activity.

Completing this activity gave me a deeper understanding of object-oriented programming concepts in C#, particularly in how to create and manage objects dynamically within a WPF application. I learned how to properly structure a class with properties and methods, as well as the importance of overriding the ToString() method to ensure meaningful output in UI components like a ComboBox.

Additionally, I gained experience working with event-driven programming by handling user interactions through button clicks. This reinforced my understanding of how to retrieve and manipulate selected data from UI elements. I also improved my ability to write cleaner and more efficient code by using an array to store course objects instead of declaring multiple individual variables.

Another key takeaway was the importance of error handling and checking for null values before performing operations, which helps prevent runtime errors. Overall, this activity helped strengthen my problem-solving skills and my ability to build interactive applications using C#.