# IT 230 6-2 Final Project Part II Milestone I Coding Activity: Create Classes for Final Project

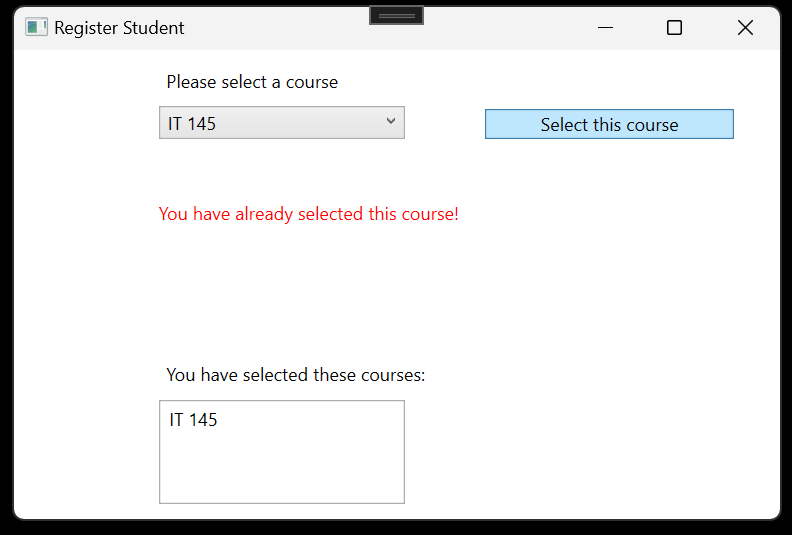
Justin Markwell

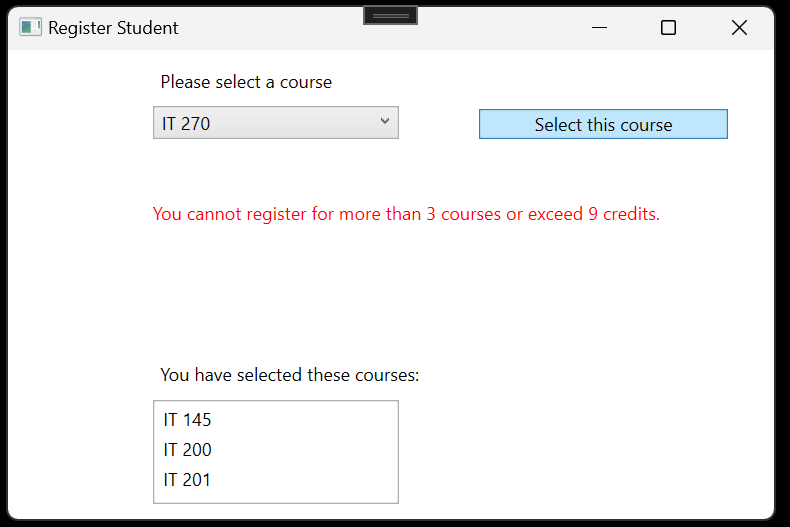
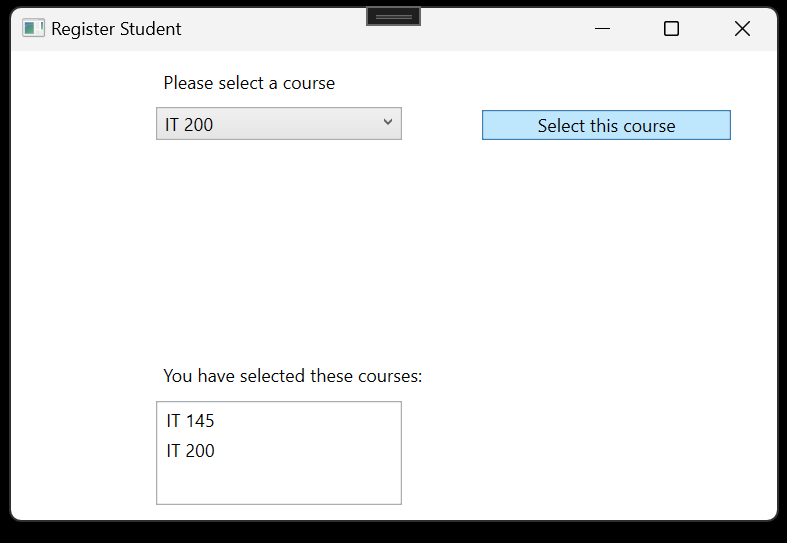
Professor: Zoe **Likoudis**

April 8, 2025

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. The program is fully functional without error
   2. Data results are accurate for the given problem

A screenshot of a computer

AI-generated content may be incorrect.



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.

//Additional optimized changes to MainWindow

public partial class MainWindow : Window

{

Course choice;

List<Course> selectedCourses = new List<Course>(); //Track selected courses

int totalCredits = 0; //Track total credits

public MainWindow()

{

InitializeComponent();

}

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

{

//used an array to hold the courseNames here to make it easier to handle listing names

string[] courseNames = { "IT 145", "IT 200", "IT 201", "IT 270", "IT 315", "IT 328", "IT 330" };

List<Course> availableCourses = new List<Course>();

//loop through and set name and credits for each available courses

foreach (string courseName in courseNames)

{

Course course = new Course();

course.SetName(courseName);

course.SetCredits(3); //All courses are currently 3 credits

availableCourses.Add(course);

}

//Populate the ComboBox with course choices

foreach (Course course in availableCourses)

{

this.comboBox.Items.Add(course);

}

}

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

{

if (comboBox.SelectedItem == null) return;

choice = (Course)(this.comboBox.SelectedItem);

//Check if the course is already selected

if (selectedCourses.Contains(choice))

{

errorMessageText.Text = "You have already selected this course!";

errorMessageText.Foreground = Brushes.Red;

}

//Check if the user is trying to register more than 3 courses or exceed 9 credits

else if (selectedCourses.Count >= 3 || totalCredits + choice.GetCredits() > 9)

{

errorMessageText.Text = "You cannot register for more than 3 courses or exceed 9 credits.";

errorMessageText.Foreground = Brushes.Red;

}

else

{

//Add course and update total credits

selectedCourses.Add(choice);

totalCredits += choice.GetCredits();

this.listBox.Items.Add(choice);

errorMessageText.Text = ""; //Clear error message

}

}}}

//Requirements for Course class

namespace CreateClassesObjs

{

public class Course

{

//Hold the course name

private string name;

//credits for each course

private int credits;

//Set the course name

public void SetName(string courseName)

{

name = courseName;

}

//Get the course name

public string GetName()

{

return name;

}

//Set Credits for course

public void SetCredits(int courseCredits)

{

credits = courseCredits;

}

//Get credits for course-in case some course in future is not exactly 3

public int GetCredits()

{

return credits;

}

//Override ToString() >>display course name in the UI

public override string ToString()

{

//return the name field

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.

### **Design and Development Process**

To further complete the course registration application, I designed a WPF project that allows users to select from a list of courses while enforcing key limitations:

Students cannot register for the same course twice

Total credits cannot exceed 9.

I began by creating a Course class that stores both the name and credit value of each course. The MainWindow then handles the logic for selection and validation.

In the Window\_Loaded event, I used a string array to store course names and a List<Course> to instantiate each course with a default of 3 credits. These were added to the ComboBox using a loop. When a course is selected, I check if it’s already been registered or if the credit limit has been reached. If either condition is met, an error message is shown using a red TextBlock. Otherwise, the course is added to the user’s list of selected courses.

To support a scalable UI, I wrapped the layout in a ViewBox so it scales nicely with window resizing, or in fullscreen.

### **Debugging Process and Steps**

**Identified and Resolved Issues:**

I ensured method names like SetName and GetCredits followed Pascal Case to meet C# naming conventions.

Improved the validation logic in the Button\_Click event to clearly handle duplicate course selections and credit overflow.

Adjusted how ComboBox items were added by directly storing and referencing the Course objects, which made it easy to display course names via the overridden ToString() method.

**Made sure to add a null check on the selected item to validate having a course selected**

**Enhanced Usability:**

Added clear error messages with a red font using a TextBlock (errorMessageText) to guide user actions.

Used layout controls like Viewbox to make sure the UI scales well on different screen sizes.

**Commenting and Readability:**

Added concise, descriptive comments throughout the codebase to explain purpose and functionality to support any future maintenance.

**Tested the Program:**

Ran various scenarios to verify all conditions:

Selecting 3 valid courses successfully

Attempting to exceed credit limit

Trying to register the same course twice

### **Reflection on the Learning Experience**

This project helped reinforce good object-oriented design by separating course data into its own class. It also gave me a deeper understanding of WPF event-driven programming and dynamic UI updates. Working with validation and error handling in a user interface context was especially valuable.

I learned how important it is to enforce constraints not just in logic but visually for the user. The project also emphasized how thoughtful structure and comments contribute to maintainable code. Overall, this was a great opportunity to apply clean design principles and improve my UI development skills.