

**IT 230 Coding Activity Submission**

**Name: Nathan Hallam**

**Date: February 25th, 2024**

**Class:** IT 230

**Module:**

| **1.** | Insert a copy of your of the ZIP file of all of your Visual Studio project files here so that it can be loaded and run in another Visual Studio:  ***See Attached*** |
| --- | --- |
|  |
| **2.** | using System.Windows;  namespace WPFRegisterStudent  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  Course choice;  int credits = 0;  public MainWindow()  {  InitializeComponent();  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  #region Create Courses  Course course1 = new Course("IT 145");  Course course2 = new Course("IT 200");  Course course3 = new Course("IT 201");  Course course4 = new Course("IT 270");  Course course5 = new Course("IT 315");  Course course6 = new Course("IT 328");  Course course7 = new Course("IT 330");  #endregion  #region Add Courses to Combo Box  this.cmbobx\_courses.Items.Add(course1);  this.cmbobx\_courses.Items.Add(course2);  this.cmbobx\_courses.Items.Add(course3);  this.cmbobx\_courses.Items.Add(course4);  this.cmbobx\_courses.Items.Add(course5);  this.cmbobx\_courses.Items.Add(course6);  this.cmbobx\_courses.Items.Add(course7);  #endregion  SetCreditText(credits.ToString());  }  /// <summary>  /// Adds a course to the list of registered courses given it is not already in the list.  /// </summary>  /// <param name="course">Course to add to the registered list.</param>  /// <returns>True if added to list, otherwise false.</returns>  private bool AddCourseToList(Course course)  {  if (course.IsRegisteredAlready())  return false;  lstbx\_registeredCourses.Items.Add(course);  return true;  }  /// <summary>  /// Sets the confirmation labels text  /// </summary>  /// <param name="text">String to set the labels text.</param>  private void SetConfirmLabelText(string text)  {  this.lbl\_confirmation.Content = text;  }  /// <summary>  /// Sets the credit textbox text  /// </summary>  /// <param name="text">String to set the textbox's text to.</param>  private void SetCreditText(string text)  {  this.txtbx\_ttlCredits.Text = text;  }  private void RegisterForCourse(Course course)  {  SetConfirmLabelText($"Registration Confirmed for course {course.getName()}.");  course.SetToRegistered();  credits += 3;  SetCreditText(credits.ToString());  }  #region Callbacks  private void btn\_register\_Click(object sender, RoutedEventArgs e)  {  choice = (Course)(this.cmbobx\_courses.SelectedItem);  //If equal or over max credits  if (credits >= 9)  {  SetConfirmLabelText($"You can not register for more than 9 credit hours.");  return;  }  //If course was unable to add to the list (already registered)  else if (!AddCourseToList(choice))  {  SetConfirmLabelText($"You have already registered for this {choice.getName()} course.");  return;  }  RegisterForCourse(choice);  } |
| **3.** | Explain the design of your program, the steps you took to complete it, and how you coded it:  This assignment was received in a partially complete state with some core functionalities missing that were otherwise stated in the design document. Starting out I read over the design document and the program's requirements to get a general understanding of what was to be expected from the assignment. After this I looked at what was already programmed in contrast with what had to be done. In this case the Course object was made, the programs frontend UI and the constructor for the UI was all complete. The main functionality that was missing was the callback handler for the button that registers the selected course.  Looking at the stub for the button callback I began with planning out each section of the callback that needed to occur, error checking and confirmation, adding the selected course to the registered list, and keeping track of the cumulative credits.  I started with adding the course choice to the list to see if it needed to be cast, which it already was in the stub. I then created a method that would handle adding a course to the registered list. This method would first check to see if the course has already been registered for and if not then it would add it to the list, returning a boolean true statement if successful. This would then be called from the button callback from an if statement guard clause, that if false would set the confirmation labels text accordingly and return before registering for the class.  This led me to make a method for setting the confirmation label as I feel it is better to have a single place that sets a UI variable that is called from anywhere rather than the UI variable itself being called from anywhere. The same logic is applied to setting the accumulative credit textbox’s text.  With a check on if a course can be registered, I began adding the logic to set a course to be registered. This included setting the confirmation label, setting the course registration bool to true, adding 3 to the cumulative credits and setting the accumulative credit textbox’s text. This is wrapped into its own method to keep the button callback from becoming cluttered.  This left one final check to ensure that the user would not register for more than 9 credit hours. This was implemented as an if statement in the register button callback before checking if the course could be added to the registered list. If the user's credits are equal to or more than 9 then the block would update the confirmation label and return before adding the course to the registration list and registering for the course.  Last of note is the change to the UI component names from label1, label2, label3, etc, to a more representative name for the given component. |
| **4.** | Reflect on this experience and the lessons you learned from it:  While I did not learn anything new in this assignment I was able to practice the general rule of having methods perform a single task as depicted by its given name. This allowed me to keep the code relatively cleaner than having all the required logic and error handling in the button’s callback. I was also reminded about the importance of properly naming variables as I had to go and look at the WPF UI code to determine what each component was called and promptly renamed then to represent their associated action - ie combobox1 became cmbobx\_courses |