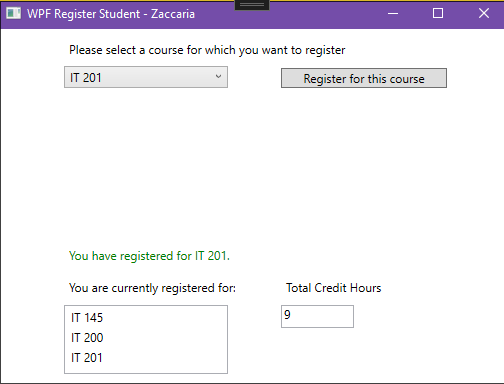
# IT 230 Coding Activity Submission Template

# 

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.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WPFRegisterStudent

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Course choice;

int totalCredit = 0; //initialize totalCredit at class level so it will get updated

public MainWindow()

{

InitializeComponent();

}

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

{

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");

this.comboBox.Items.Add(course1);

this.comboBox.Items.Add(course2);

this.comboBox.Items.Add(course3);

this.comboBox.Items.Add(course4);

this.comboBox.Items.Add(course5);

this.comboBox.Items.Add(course6);

this.comboBox.Items.Add(course7);

this.textBox.Text = "";

}

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

{

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

int credit = 3; //make credit hours variable

if (choice == null) //if there is no course selected

{

label3.Content = "You must select a course to add to the registration."; //output error message (in label3 box) telling user to select a course

label3.Foreground = Brushes.Red; //output the message in red to indicate error

return;

}

else //if there is a course selected

{

if (choice.IsRegisteredAlready() == true) //check if the course is already registered

{

label3.Content = ("You have already registered for this course."); //output error message (in label3 box) telling user they have already registered for the course

label3.Foreground = Brushes.Red; //output the message in red to indicate error

return;

}

else //if the course is not already registered

{

totalCredit += credit; //add credit hours to totalCredit

if (totalCredit > 9) //check if totalCredit is greater than 9

{

label3.Content = ("You cannot register for more than 9 credit hours."); //output error message (in label3 box) telling user they cannot register for more than 9 credit hours

label3.Foreground = Brushes.Red; //output the message in red to indicate error

return;

}

else //totalCredit is less than or equal to 9

{

choice.SetToRegistered(); //set course to registered

label3.Content = ($"You have registered for {choice.getName()}."); //output message (in label3 box) telling user they have registered for the course

listBox.Items.Add(choice); //add course to listBox

label3.Foreground = Brushes.Green; //output the message in green to indicate success

textBox.Text = totalCredit.ToString(); //output totalCredit to textBox

return;

}

}

}

}

}

}

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.

When I started writing the code for this program, I knew that there was going to be a lot of branching going on to check for the different possibilities that could invalidate a course the user is trying to register for. It made the most sense for me to break it down into steps and think about what should be checked first. I decided to make a list to make sure I covered all the branching that was going to be happening. The first branch was that the user has not selected a course, and that the user has selected a course. If the user did not have a course selected, then output an error message and end that part of the branch there. If the user did have a course selected, then move on to our second branch. The second branch is checking if the user has already registered for the selected course, or if the course has not been registered yet. If the course has already been selected, then output an error message and end that part of the branch there. If it has not already been registered, then we can move onto the third and final branch. The final branch first needs to check if they are trying to register for more than 9 credit hours (3 courses); if they are trying to exceed 9 credit hours, then output an error message and end that part of the branch there. If they are not exceeding 9 credit hours then set the course to registered so they cannot register for it again, add the course to the list box so they can see which courses they have registered for, output a message telling them they have successfully registered for the selected course, and update the total amount of credit hours they are at. Now that I had my code all laid out in plain text, it was time to get to work on the actual program. The first branch was easy, all I needed to do was write and if else statement using null to see if the user had selected anything when they pressed the register button. If they did select something I could just use the else part of the if else statement because there are no other conditions that need to be met, the user has selected something, and I can continue from that side of the branch. Next, I used another if-else statement, starting with using the IsRegisteredAlready method to check if it course has already been registered for. If it returns true, then they have already registered for that course, and I can use the getName method to write the error message and display the specific course inside the error message. Again, I can just use the else since there are no other conditions to check. We know the user has selected something, and their selected course has not been selected yet, so we can continue from that side of the branch. Now an important step that I needed to make sure was right was that the math operator responsible for adding the credit hours to the variable storing the total credit hours was not inside of the next branch at all. I needed to write it before the next branch started because no matter what happens at the next branch, the credits need to be added on first. For the final branch, I used an if-else statement again, first checking if the student has registered for more than 9 credit hours. This is why it was important to update their credit hours outside of the current branch. If it exceeds 9 credit hours, then that side of the branch ends there with an error message. I just used the else again since I did not need to check for other conditions; this way we know everything is in order for their registration since we have already weeded out all the other possible problems. First we want to use the SetToRegistered method to make sure the user cannot register for that course again, then we use the getName method again to display that the user successfully registered for their selected course, then we need to add the users selected course to the listBox that displays their selected courses, and lastly we need to display the number of credit hours in the textBox that shows the user the amount of credit hours they have based on their selected courses. For that we need to use the built in ToString method since the credit hours are being stored as integer values and to display it in a textBox it needs to be a string. After that you have completed everything you need for this assignment.

(I apologize for the double explanation, I just wanted to make sure to include my whole processes even though in the explanation it sounds repetitive)

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

A learning experience that I had while completing this activity that I did not mention in my explanation was that I learned how to look at the .xaml code and really understand what was going on. While it is not that complicated, after reading through it, I was able to alter my code into something that was not necessary, but to me it seemed to make the most sense for this assignment. I found the names for each label, and since I had the names for each label, I was now able to alter things going on with the label inside my code, opposed to hard coding and changing it in the .xaml code. As I read the line of code for the label that displays whether the user has successfully registered for a class or if there was an error, I noticed the “Foreground = (Red Square)” at the very end, which got me thinking about changing the color of the message outputted in that label depending on which branch of code the program makes it to and ends at. I already knew that the start of the code was going to be “label3.” since we used “label3.Content” to display the messages to the user. So, I assumed that we were going to need to do “label3.Foreground” which was correct, but I still did not know how to change the color. With a little Google search, I was able to find out that I can do “label3.Foreground = Brushes.(Desired Color)” to change the color of the text that was being outputted to the user. After I figured it out, I went to all the branches that ended with an error message and wrote in the code to make sure they always displayed as red text. Then I went to the output that was for the user successfully registering for a course, and I set it to make sure the text always output as green. This was something really simple, but cool and fun to mess around with, and I feel like it helped my program look more professional instead of displaying everything in red, subconsciously implying that there was an error even when it was successful.