# IT 230 Final Project Part II

Ruben Garcia

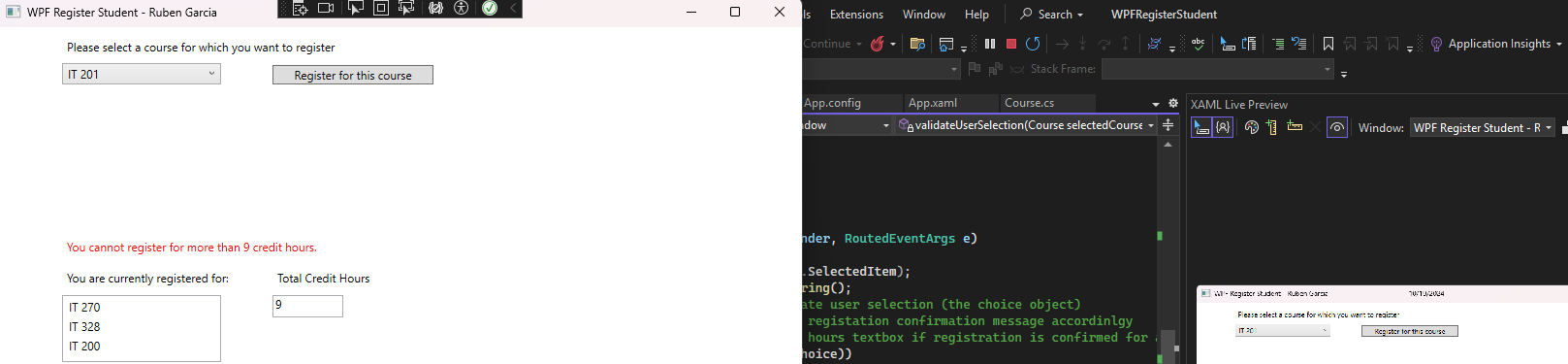
John Wetsch

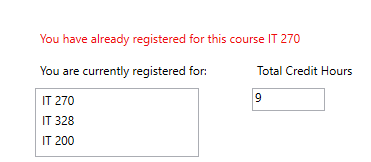
IT-230-11599-M01 Software Develop with C#.NET 2024 C-5 (Sep - Oct)

10/19/2024

Module Seven: Submission

As shown in the following output, the program is now working as intended. My name is provided alongside the date, reaffirming this is in fact my own running output. The provided screenshots help highlight the few possible outputs meant to limit the user from simply choosing random choices.





With the code essentially set up unfinished, we can pinpoint the exact segment that needs to be addressed to commence the operation. The main point which we must focus our attention on would have to be ‘MainWindow.xam.cs.’ The main reason as to why it matters is because ‘MainWindow.xaml’ is essentially the output that helps display the menus intended to be navigated. ‘MainWindow.xam.cs’ directly affects what’s being calculated and then displayed, which is what we need to complete here due to the code being unfinished in this portion.

Seeing as how we will be utilizing int, it is important to establish this early on within the given code to help formulate credits as shown in the following code:

“public partial class MainWindow : Window

{

Course choice;

int TotalCreditHours = 0;

public MainWindow()”

During this portion of the code, we need to be able to receive confirmation whenever an error is being displayed. If there’s too much credit or if there’s a class being selected more than once, then we need to be able to provide a limitation to make those inputs invalid without having to close out from the operation completely simply because it is invalid. The following code addresses this carefully, validating that information as shown in the following code:

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

string courseName = choice.ToString();

switch (validateUserSelection(choice))

{

case 0:

label3.Content = "You have already registered for this course. " + courseName;

break;

case 1:

label3.Content = "You cannot register for more than 9 credit hours.";

break;

case 2:

choice.SetToRegistered();”

Although during this portion I ran into a few errors, it wouldn’t be long until I realized that the total credit hours can simply be updated as intended shown in the video provided within the module requirements by adding “TotalCreditHours += 3;”

“ listBox.Items.Add(choice); // What essentially these lines of code equates to would be that there are registered input confirming info

label3.Content = "Registration confirmed for course..." + courseName;

TotalCreditHours += 3;

textBox.Text = TotalCreditHours.ToString();

break;

}

}

private int validateUserSelection(Course selectedCourse) //This line of code is meant to verify output and limitation being set up that way nothing exceeds the epxpected amount

{

if (selectedCourse.IsRegisteredAlready())

{

return 0;

}

else if (TotalCreditHours > 8)

{

return 1;

}

return 2;

}

}

}”

With the rest of the code, the program is essentially checking for user input to formulate valid inputs or not. The user ends up being validated even if there are invalid inputs, making it unable to close out the entire program since we are establishing and controlling the options for the user.

It is a lot easier whenever you have an existing code since you essentially have the foundation of the entire code already done, so I found myself having little problem understanding and providing solutions to the existing code. I had an easier time figuring out this code with a group of friends of mine since they were curious about my work, which made it all the easier to finish due to their suggestions and testing. The following would be the source code.:

ng 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 TotalCreditHours = 0;

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);

string courseName = choice.ToString();

// TO DO - Create code to validate user selection (the choice object)

// and to display an error or a registation confirmation message accordinlgy

// Also update the total credit hours textbox if registration is confirmed for a selected course

switch (validateUserSelection(choice))

{

case 0://We need to establish an error in order to appeal to the display error requirement and incase it calls for it

label3.Content = "You have already registered for this course." + courseName;

break;

case 1:

label3.Content = "You cannot register for more than 9 credit hours.";

break;

case 2:

choice.SetToRegistered();

listBox.Items.Add(choice); // What essentially these lines of code equates to would be that there are registered input confirming info

label3.Content = "Registration confirmed for course..." + courseName;

TotalCreditHours += 3;

textBox.Text = TotalCreditHours.ToString();

break;

}

}

private int validateUserSelection(Course selectedCourse) //This line of code is meant to verify output and limitation being set up that way nothing exceeds the epxpected amount

{

if (selectedCourse.IsRegisteredAlready())

{

return 0;

}

else if (TotalCreditHours > 8)

{

return 1;

}

return 2;

}

}

}