***NOTE: Because this is the last week of the course, your homework assignments and peer reviews are due before the last day of the course.  Check and verify the last date of the course to ensure you submit on time.  As a result, this assignment does not require a lot of coding and most of the code is written for you already.***

***NOTE:*** ***This is a WPF application and requires Windows to run.  If you are not using Windows, you will need to either modify the code for your platform or open the .cs file and make changes to the code.  You may not be able to test the functionality.  There are many free options available such as a trial version of Windows on the Microsoft TechNet Evaluation Center and most platforms support virtual machine environments.***

In this assignment, you will be asked to modify some code, which is similar to the Module 9 assignment.  Your changes will be designed to take advantage of tasks.  A starter project has been created for you and you can [download it here](https://d37djvu3ytnwxt.cloudfront.net/assets/courseware/v1/81933cd5d0945b79eb6874f53d118d46/asset-v1:Microsoft+DEV204x+1T2017+type@asset+block/Mod11_Assignment.zip).  This starter project is the same one used in the demo for this unit with some modifications.

In this project file, the creation of student objects is being handled in the button click event and is included for you already.  There is a call to a method called AddToCollection(Student student).  This method makes use of the Student object that you pass in to it. The problem is, using the new Action syntax that was used in the demo will not work if your action method requires a parameter.  As a result, you must use the Lambda expression variant of creating a task.  Review the module content to see how to do this.

Inside the AddToCollection method you will notice a line of code that causes the thread to sleep for 5 seconds.  This is placed in here to create a delay in the program.   The delay is intended to simulate an application that is being used by multiple users adding students to a database.  If the database and network are busy enough, they will cause a bottleneck and could result in the UI being non-responsive.

Run the application in its current state.  Add student information and click the Create Student button.  Notice that the UI is unresponsive for 5 seconds.  Again, this delay is only simulating an unresponsive application due to the thread sleep, but is intended to show how a blocking thread can affect UI responsiveness.

There are many ways to handle this in a real-world application but in this instance you will create a separate task for this process.   This way, the UI thread will not be blocked and the user can immediately start to enter more text in the fields.

This assignment only requires you to create the necessary code to use a task to unblock the UI.

Do NOT simply remove the Thread.Sleep command.  You must correctly implement a task to achieve the desired effect.

Paste your code in the peer review.  You only need to paste the code of the method, not the entire .cs file.  Some of your peers are not using Windows and may only create the code changes to implement tasks, which is ok for this assignment.   No events or UI code are necessary for grading this assignment.  Just the code for using a Task.