# CSC121 Lab 14: GUI Programming

## Goals

In this lab assignment, students will demonstrate the ability to:

* Use the tkinter module to create simple GUIs
* Explore other sources beyond the textbook for Python information

NOTE: This lab document is a copyrighted work of Wake Tech Community College and the course instructor. Any posting of this document outside of Wake Tech is considered a copyright violation. Students who post these documents outside of Wake Tech are subject to academic and possible legal actions.

## Instructions

In this lab, you will use a Python standard library called tkinter for building graphical user interfaces.

Follow the instructions in each problem and submit the specified files.

Problem 1 is a program that you create from scratch that meets the problem specification. Additionally, there is an extra credit opportunity where you create a new program based on a problem specification.

## Problems

### Problem 1

In this problem, you will build a grade calculator GUI application. This application uses many elements in tkinter including Frames, Labels, Entry, and Buttons.

This GUI application will accept three values from the user:

* The average grade of all the tests taken.
* The average grade of all the labs completed.
* The average grade of all the exams taken.

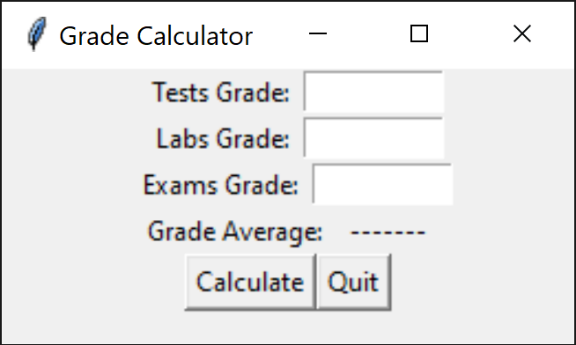
When the user hits the Calculate button, the program will calculate the overall average of the grades using the following weighting:

Table 1: Grade Weighting for Activities in application.

| **Activity** | **Weighting** |
| --- | --- |
| Tests | 20% |
| Labs | 30% |
| Exams | 50% |

When the user hits the Quit button, the program will exit.

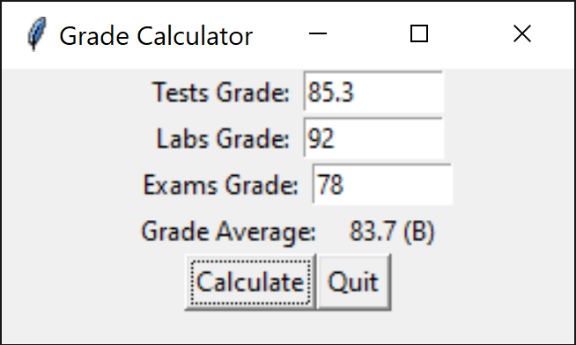
Here is what the application should roughly look like when it starts:



Here is a written version of the above graphic.

* The application has a title bar with the title "Grade Calculator".
* The first line has a Tests Grade label followed by a blank text field.
* The second line has a Labs Grade label followed by a blank text field.
* The third line has a Exams Grade label followed by a blank text field.
* The fourth line has a Grade Average label followed by another label with dashes.
* The fifth line has two buttons: A Calculate button and a Quit button.

Here is what the application looks like after the user performs a Calculate operation:



Here is a written version of the above graphic:

* The Tests Grade text field has 85.3 filled in.
* The Labs Grade text field has 92 filled in.
* The Exams Grade text field has 78 filled in.
* The field after Grade Average shows 83.7 followed by B in parenthesis.

Note that this application should gracefully catch errors in the user's input (like entering a number that cannot be converted to a float) and provide some kind of feedback. This feedback could be done by changing the label after Grade Average to "Error" or the application could pop-up an error dialog. Either is acceptable for this implementation.

When the grade is displayed, it should be displayed with 1 digit after the decimal point along with the letter grade in parenthesis. The letter grade is determined using the Wake Tech grading scale, so A is 90 and above, B is 80-89.9…, etc. with any grade below 60 as F.

This application has two features not mentioned in the textbook:

* This application has a custom title, "Grade Calculator" in the application title bar. To get this custom title, you use the title method of the main window:  
    
  self.main\_window.title('Grade Calculator')
* This application starts with a custom size of width 260 by height 125. To get this custom size, you use the geometry method of the main window:  
    
  self.main\_window.geometry('260x125')

Create a file named **Lab14P1.py** and implement your GUI application in that file. Review the textbook for details on how to create each element and how to support the updates of the grade average label using a callback function. The textbook also provides details for creating a quit button using the destroy method.

For this problem you are **REQUIRED** to implement this solution using the **class-based approach** shown in the textbook and described in the video lectures. Solutions that are submitted using a function-based approach will be lose 50% credit for their implementation.

Solutions are also **REQUIRED** to use the **pack geometry manager** for laying out the UI as was also shown in the textbook and described in the video lectures. Solutions that are submitted using a different geometry manager will lose 50% credit for their implementation.

Submit the program file **Lab14P1.py** to Blackboard for credit.

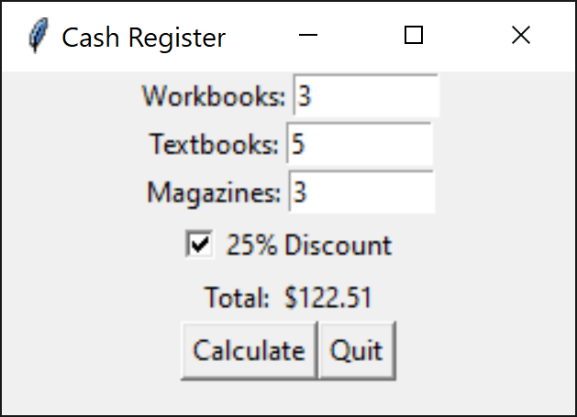
### Extra Credit Opportunity – 15 or 20 points

You have the opportunity to attempt extra credit for this lab. This option provides a specification for a simple cash register GUI application that you can follow directly. If implemented correctly you will get 15 extra credit points.

This application allows a cashier to enter the number of workbooks, textbooks, and magazines a customer purchases. It also provides a checkbox for indicating whether the customer should get a 25% discount. When the calculate button is pressed, the total purchase amount is calculated.

* Workbooks are 8.50 each.
* Textbooks are 24.00 each.
* Magazines are 5.95 each.

Here's what the application looks like when running and filled with user input:



Here is a written version of the above graphic:

* The main window title bar is labeled "Cash Register".
* The first line has a Workbooks label followed by a text field which contains the number 3.
* The second line has a Textbooks label followed by a text field which contains the number 5.
* The third line has a Magazines label followed by a text field which contains the number 3.
* The fourth line has a checkbox that has been labeled "25% Discount".
* The fifth line has a Total label followed by a label with the value "$122.51".
* The sixth line has a Calculate and Quit button.

Note: The geometry used in the graphic was '250x150', but you may use whatever geometry best works for your application.

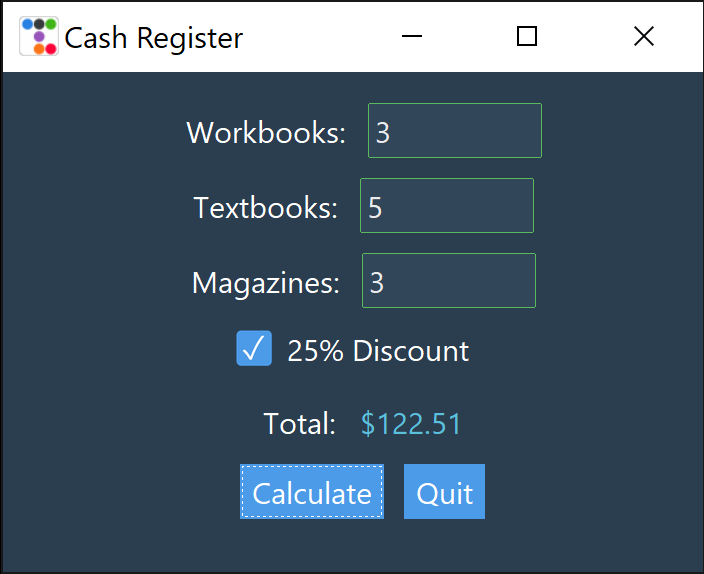
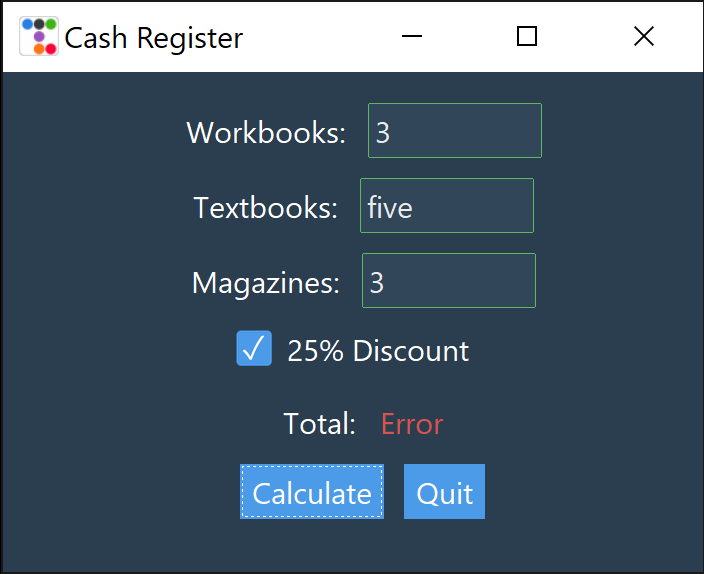
Create a file named **Lab14Extra.py** and implement your GUI application in that file. As with Problem 1, the information for creating an application like this is mostly in the textbook except for the window title and geometry information. That information is provided in Problem 1.

Note also that this program should gracefully catch invalid data input by the user such as not entering an integer to indicate the number of an item.

As with Problem 1, you are REQUIRED to implement this solution using the class-based approach and using the pack geometry manager shown in the textbook and described in the video lectures. Solutions that are submitted using a function-based approach or other geometry managers will receive **no** extra credit.

Submit **Lab14Extra.py** to Blackboard for credit.

**Want 5 additional points for this problem?** Investigate the ttkboostrap library and produce the same application using the additional GUI styling from that library. Here's an example of this same application using the ttkboostrap library in addition to tkinter:

If you choose this option, submit **Lab14Extra-ttkb.py** to Blackboard for credit.

Here's some resources for investigating ttkbootstrap:

* [Python Tkinter GUI Design Using tkkbootstrap – Complete Course](https://youtu.be/0tM-l_ZsxjU)  
  [Plain text URL: https://youtu.be/0tM-l\_ZsxjU]
* [Official ttkbootstrap documentation](https://ttkbootstrap.readthedocs.io/en/latest/)  
  [Plain text URL: https://ttkbootstrap.readthedocs.io/en/latest/]

NOTE: To use the ttkbootstrap library, you will need to pip install that library. It is expected that you would also need to adjust the geometry of your GUI to show all the controls. You might also find it better to not specify the geometry as ttkbootstrap may adjust it automatically. It is okay to use any theme or styling you want!

## Grading Rubric

### Grading rubric for Problem 1 (100 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [95 points]
* Solutions that do not use a class-based approach LOSE 50% of the grade.
* Solutions that do not use the pack geometry manager LOSE 50% of the grade.

### Grading rubric for Extra Credit – Problem Spec Option (15 or 20 points)

* Program is well-formatted with a correct header, executes correctly, and produces correct results [15 points]
* Program ALSO uses the ttkboostrap library for extra styling [5 more points]
* Solutions that do not use a class-based approach will receive NO extra credit.
* Solutions that do not use the pack geometry manager will receive NO extra credit.