Timed Lab 2 – Element Finder

This Timed Lab is worth 21 **Exam** points.

For this Timed Lab, you *may* use	However, you *may not*
• Course notes • Homeworks • Recitation assignments • Other course material • Any material you may find on the Internet that don't involve communicating "live" with other people.	• Communicate with other people/students in real-time via any means. This means no Facebook, email, Piazza, IM, IRC, cell phones, Google Talk, smoke signals, etc. • Share code with other students. • Look at other students work.

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistance with debugging code, program logic, etc. You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends: No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

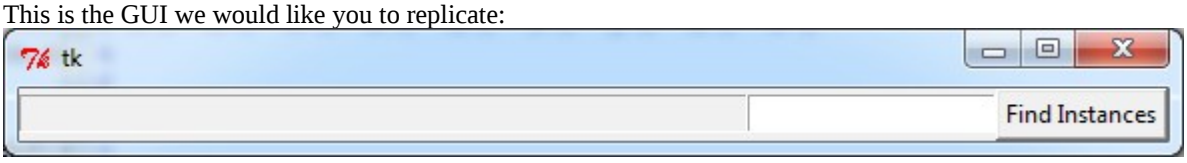
T-Square will not permit any late submissions; ensure that you submit your code to T-Square several times to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warning 10 and 5 minutes before the end of the recitation period; you should submit at these times.

In your collaboration statement, if you use code from somewhere that is *not* a class resource (i.e. not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the file.

Problem Description:

In this Timed Lab, you will be given a CSV file with certain data in it (they may be strings or numbers), as well as a certain element to look for in the file. Your responsibility is to build a simple GUI that will allow the user to specify the file to search in as well as the data to search for, and your program should return the row and column coordinates of each cell where the data in the cell **exactly matches** (not just a substring) the data specified by the user.

This is the GUI we would like you to replicate:



The first entry box is of width 60 and is readonly. The second entry box is default width and is normal. The button has text of “Find Instances” and will call the clicked method when it is clicked. The user will enter a string (it could be anything contained in the CSV file) in the second entry box. The user will then click the “Find Instances” button, which will then open a file dialog, allowing the user to select a CSV file to open. From there, the user will select the appropriate CSV file, and your program will report back all the locations of the data the user entered in the second entry box. Below is the output after the program is run on the sample file:

Use the above format to display the location of each instance. The first instance of the piece of data in the file should be first in the entry, followed by the second, etc. HINT: The above format is simply a list of tuples displayed as a string…

The user should be able to run the program again with a different file and a different piece of data to look for. However, you may assume that the user will always enter a value into the second entry box, and that the user will always select a valid CSV file in the file dialog box.

You should split your program into a class (which you may name whatever you wish) and four methods inside this class:

1. **The constructor**
   Creates all of your widgets and places them in the GUI.
2. **clicked()**
   -Creates an instance variable named data which is an empty list
   -Displays the open file dialog
   -Calls readData with the filename of the file the user selected
   -Calls findInstances
3. **readData(filename)**
   -Opens the file
   -Places each row of data into the data instance variable (data should be a list of lists when you're finished, where each list inside data contains strings)
   -Closes the file
4. **findInstances()**
   -Locates each instance of the requested data and stores this location
   -Places correctly formatted string with all instances into the first entry box in the GUI.

Make sure to add the required code to make your GUI appear when your file is run!

Grading:

+2 – **Displays GUI when file is run**
+5 – **Constructor**
  +1: Correct method header
  +2: First entry box is in correct location with width 60 and is readonly.
  +1: Second entry box is in correct location with default width and is normal.
  +1: Button is in correct location with text "Find Instances", calls clicked method when clicked.
+4 – **clicked()**
  +1: Correct method header.
  +1: Correctly asks for file using dialog box.
  +1: Correctly declares data instance variable.
  +1: Correctly calls both readData and findInstances methods.
+5 – **readData(filename)**
  +1: Correct method header.
  +2: Correctly reads in all data from file.
  +1: Stores read-in data in data instance variable.
  +1: Closes the file
+5 – **findInstance()**
  +2: Correctly considers all data in file (accounts for different sized rows).
  +2: Correctly inserts data into correct entry box with all correct locations.
  +1: Data is in correct format when inserted into entry box.