Introduction to Java

CS9053

Tuesday 6 PM – 8:30 PM

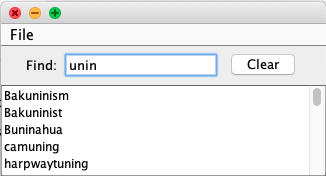
Prof. Dean Christakos

October 28, 2020

Due: November 4, 2020

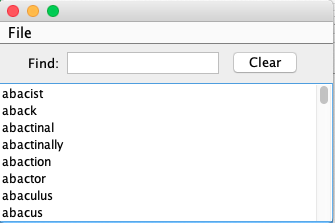
Part I:

You’re going to write an application where you type a word into a text field and matches words in a dictionary. Whatever is currently in the text field updates what appears in the text box. For example:

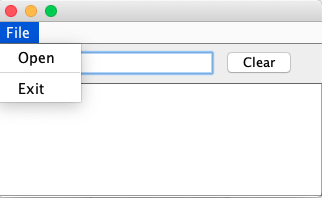


As you can see, “unin” matches all the words that appears in the text box. The text box should update constantly as the user types. Pressing Enter should not be necessary. (Hint: this requires you to use a listener that receives every change to the text field; see the [JTextField](https://docs.oracle.com/javase/8/docs/api/javax/swing/JTextField.html)class overview for a hint about which listener to use. It was also mentioned in lecture).

When the query is blank, the list box displays the entire word list:



The menu bar should look like this:



When the program is run, it starts empty. You should then be able to load a list of words from the file words, provided in the Java project for this midterm. The first 100 words are in the file words100, if you want to experiment because that takes far less time to load.

If none of the words contain the query, the text box should be empty.

The Clear button should clear the query field, restoring the list box to displaying all words again.

An outline is contained in the file WordFinder.java. To search, you will use the class WordList.java. WordList.java is loaded using the method WordList.load, which takes a FileInputStream. All this will be handled in the OpenFileListener, the structure of which, including the file selection process, is written. You need to figure out how to get the results out of WordList, based on your search term.

Hint: to scroll a JTextArea all the way to the top, use the method setCaretPosition(0)

Hint: you can provoke an action in a Java Swing objection using the method postActionEvent();