Introduction to Java

CS9053

Tuesday 6 PM – 8:30 PM

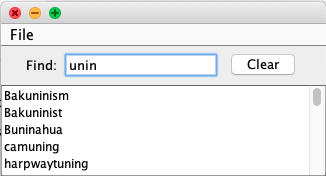
Prof. Dean Christakos

October 28, 2020

Due: November 4, 2020

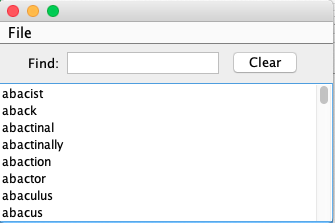
Part I:

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:

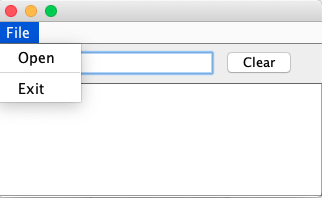


As you can see, “unin” matches all the words that appears in the text box. The text box should update constantly as the user types. Pressing Enter should not be necessary. (Hint: this requires you to use a listener that receives every change to the text field; see the [JTextField](https://docs.oracle.com/javase/8/docs/api/javax/swing/JTextField.html)class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

**Part II**

Here we are going to go over some new material in GUIs: the concept of a Canvas.

In DrawSpace.java, as you can see, we create three classes DrawingCanvas1, DrawingCanvas1, and DrawingCanvas1 of size 150x150 that extends the Canvas class, and they are embedded in a 600x200 JFrame. It currently does nothing. You are going to make it do something.

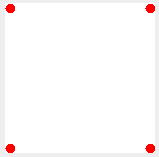
As you can see, we have overloaded the method paint(Graphics g). You might notice that it executes on start and something when you move the window around. This runs automatically, and you never call the method paint() directly.

To draw a circle, we can use the method fillOval on the Graphics object, g, in paint, documented here: <https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html#drawOval-int-int-int-int->

To set the color of what you will draw, you would use g.setColor and one of the static Color values. Eg, Color.RED or Color.BLACK, and so on, as you can see in the documentation.

Question 1

Draw four circles of radius 10 in each of the four corners of the canvas on the left, DrawingCanvas1. A Canvas’s coordinate 0,0 starts in the upper left of the Canvas. Coordinate 150,150 (or whatever the dimension of the Canvas) is in the lower right. The output should look like this:



Question 2:

Now you are going to work with DrawingCanvas2. You are going to create a MouseListener class and assign it to your DrawingCanvas2 using addMouseListener.

Create a subclass CanvasMouseListener that extends MouseListener. You’ll notice that the compiler gives an error. Eclipse will allow all of these methods to be created, but the important one is public void mousePressed(MouseEvent e). A MouseEvent has methods getY() and getX() which shows where the event occurred. Play with it, printing out the x and y coordinates to get an idea of how this works.

**The goal for question 2:** When you click on the canvas, you will fillOval at the coordinate the user has clicked at. You have to figure out how you will do that, but I will give you plenty of hints and guidance:

What you want to do is create a memory space that records what the canvas should draw. The canvas is 150x150, so you will want variables that represent what is to be drawn at each coordinate in the canvas.

The getX() and getY() MouseEvent in mouseClicked should set which coordinate has been clicked on in a variable, and then mouseClicked should call repaint(). repaint() calls the