Lab 12 C212 FA22

Last modified 2022-11-16 4:40 PM

Summary

The goals of this lab are as follows:

- Part A - BMI Calculator
- Part B - Shape Configurer
- Part C - Show us what you've done
    - Get checked off before leaving lab

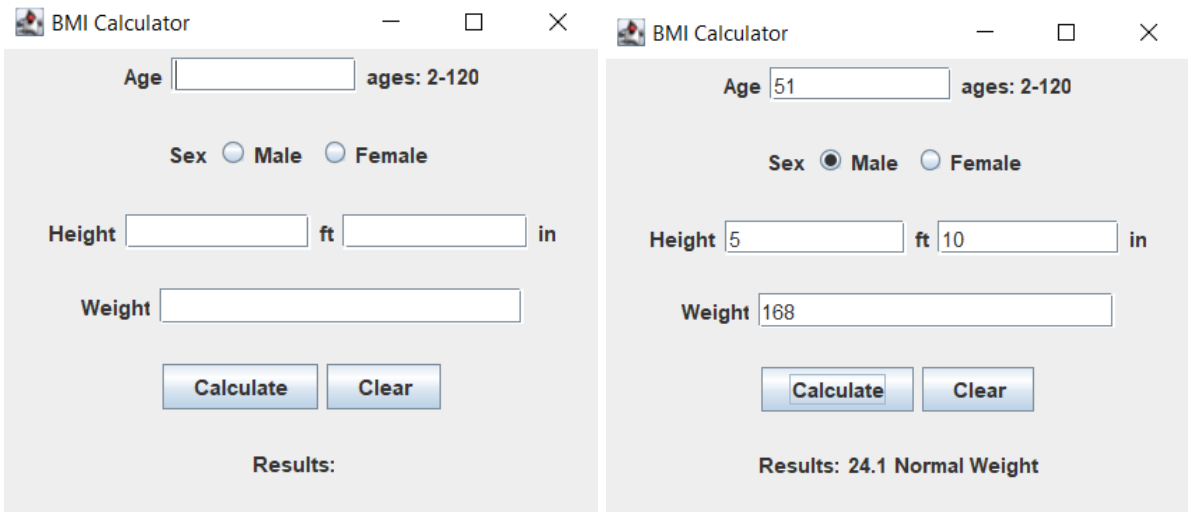
Since Assignment8 requires you to use a zipped project for starter code, this week's lab's starter code is also a zipped project folder. The general theme of this lab is preparing you for Assignment8 while also teaching you some new things with GUI that don't appear there.

Part A - BMI Calculator

In Part A you'll be creating a BMI Calculator GUI-based program like the following. The calculation is handled for you in the starter code, you just need to manage the GUI.

This exercise is designed to be similar/applicable to the current assignment, where elements are defined at the lowest-possible scope and references to them are passed in to the components that need to be able to access them. This is an alternative to the approach we used in lab last week, where everything is defined statically – this is bad practice. We want to control the scope of our variables and restrict their access to only what is necessary. Additionally, we use methods to group creation of our different panels in this exercise, which also matches what you'll have to do in A8.

Here is what the finished product should look like, first as the default GUI then with some data entered and the result calculated:

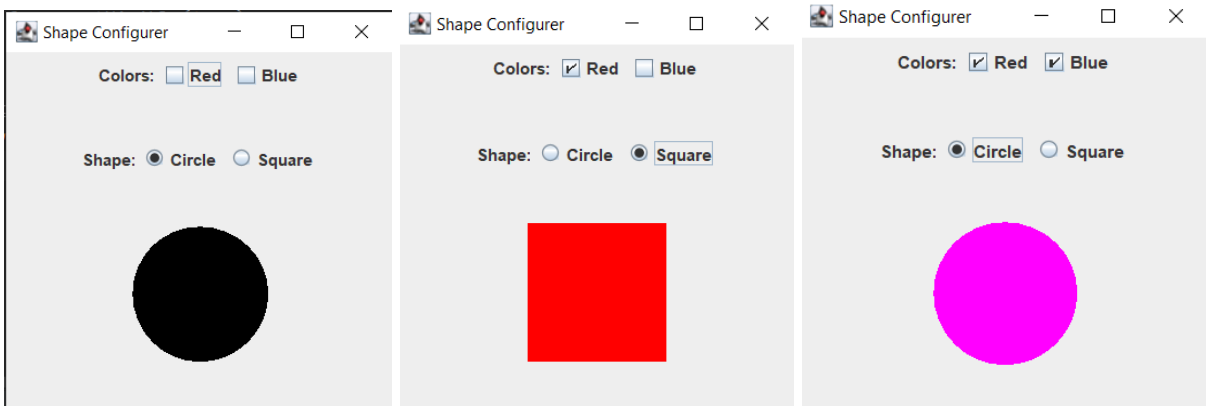


It works as follows:

- main() gets a frame from getFrame() and makes it visible. This is done for you.
- getFrame() creates the frame, sets its properties, creates a main panel which holds other panels, creates some elements that need to be accessed in multiple places, creates sub-panels by calling the corresponding methods, passing in references to the elements it needs to function, and finally does some last steps with the mainPanel and frame and returns the frame.
- Each individual row panel starts with a basic row panel. In case you want to play around with the layout of every row at the same time, you can modify getBasicRowPanel(). Otherwise, each row panel just adds the elements that appear in that row, using parameters if it is provided them or defining the elements if not. For this task, each method already includes in its header the parameters it will need, just like the assignment. Then, it returns the panel.
    - The calculateRowPanel() has items with attached state changed listeners, so it needs to be given those with event handlers, but otherwise works mostly the same. Since it needs to read the values in all the inputs, it takes in a lot of parameters, and it also has the two output elements to overwrite each time calculate or clear is pressed. Calculations are mostly handled by helper methods in another file that separate the business logic from this file – again, like the assignment.

Part B - Shape Configurer

In Part B you'll be creating a GUI-based program like the following, which shows the program at different states based on user choices of zero or one or many colors and exactly one shape:



Part B is about using a different kind of event listener: the ItemListener interface and its method itemStateChanged. This fires when the state of an element changes, such as whether or not a button like a checkbox or radio button is selected. Notice how there is no submit button! In this task, we will automatically update the shape any time a button is selected/unselected for a more responsive user experience.

The DrawnShapeComponent class, a subclass of JComponent, has a constructor this time. Yes, this is possible! This lets us configure how our drawings will work based on variables and logic for that instance of DrawnShapeComponent.

We leave some of the GUI stuff to you to do from scratch as practice, though a lot is done for you. We'll give some guidance with the new event listeners and focus on that here, as well as how we can dynamically draw the shape differently using instance variables in DrawnShapeComponent.

Event listeners and their event handlers have some information you can extract from the event object, often seen as the e in `ActionEvent e` or, in this case, `ItemEvent e`. For example, you can determine what element caused the event to fire. For our purposes, though, we don't need to know, because the DrawnShapeComponent on-screen gets replaced with a brand new instance of DrawnShapeComponent each time anything changes. This is meant to match DrawnImageComponent from A8, which is replaced each time a new transformation is applied to the input without pressing "reuse". Therefore, we re-instantiate the DrawnShapeComponent by either changing the color settings or shape settings, depending on which kind of button was pressed. We don't care WHICH button (e.g. red or blue) was pressed, only what type of button – this is why we have two different event listeners, one for checkboxes and one for radio buttons – and we just read in the currently selected elements in both cases.

The process you need to follow is guided in the lab starter code using TODOs.

Part C - Show us what you've done

As announced recently on Canvas, we want to try and expedite grading, especially for labs as they should be a way to recap and practice recent lecture topics so you're ready for the assignment.

To that end, assuming you're working on the lab at your lab section, show your TAs your final product and they'll check you off and give you the full points. Otherwise, submit your work to Canvas before the deadline.