CS 342 Homework#5 Fall 2021

JavaFX Basics: Using FXML

Due Date: Friday, 11/5, by 11:59PM

Description:

In this homework, you will replicate the GUI you made for homework #3 utilizing FXML and CSS style sheets. You can use the FXML sample Maven project on Blackboard to work in and submit. You must have at least one of each: a controller class, a .fxml file and a .css file. The user interface must render only using this approach. The appropriate GUI elements must be clickable and have methods that are called when the user interacts with it. These methods must be defined in a controller class and attached to the widgets in the .fxml files.
If you submit a working version of homework #3 that creates and manages the GUI with fxml instead of programmatically, you will receive extra credit on this homework.
If you do not want to submit a working version of homework #3 using fxml, when widgets are clicked you can simply have the methods attached print something to the terminal. We will only concentrate on the successful creation of a user interface using fxml and css.
Below, is the write up of homework #3 for your reference….

Implementation Details (this is the description from homework #3):

You will use the Maven project provided in the homework link to work on this assignment. You will also need to use the documentation for JavaFX BorderPane, VBox, TextField and Button. It can be found here:

https://docs.oracle.com/javase/8/javafx/api/index.html?help-doc.html

First, change the title of the app to "(your first and last name) Homework 3". Your GUI should have two buttons (use the JavaFX class Button) with the labels "button 1" and "button 2" as well as two text fields (use the JavaFX class TextField). Utilizing a BorderPane (a class in JavaFX), set one TextField to the center and the other TexField to the right in the BorderPane. Set the buttons in an VBox to the left inside the BorderPane.

The TextField in the center should display prompt text letting the user know what to do with it; the prompt should say "enter text here then press button 1". You will need to look at the documentation for TextField to find the proper method to do this.

The TextField on the right should not be editable (user can not type in it). It should start out displaying the message “final string goes here”. This string should NOT be a prompt. Consult the documentation to find the correct methods.

Create and attach an EventHandler to “button 1” utilizing an anonymous class. When clicked:

- you will get the text from the text field in the center,
- concatenate that text with the string “ : from the center text field!”
- set the text field to the right with the new string
- disable button 1 so it can not be pressed
- change the string displayed on button one to “pressed” (instead of “button 1”).

Create and attach an EventHandler to “button 2” utilizing a lambda expression. When clicked:

- you will clear the text from both TextFields,
- replace the string “final string goes here” in the TextField to the right
- enable the first button so it can be pressed again
- replace the string displayed on that button with “button one”.

You will notice when you complete the above, that your user interface does not look very good. Look into the documentation for BorderPane and figure out how to improve the look of your app and implement it.

Electronic Submission:

Put the Maven template folder with your files in a .zip and name it with your netid + Homework5: for example, I would have a submission called mhalle5Homework5.zip, and submit it to the link on Blackboard course website.

Assignment Details:

Late work on a homework is **NOT ACCEPTED.** Anything past the deadline will result in a zero.

We will test all homework on the command line using Maven 3.6.3. You may develop in any IDE you chose but make sure your homework can be run on the command line using Maven commands. Any homework that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.