Exercise sheet 3

Interactive Systems - SoSe 24

Prof. Dr. Michael Rohs, Jan Feuchter, M.Sc.

All exercises that are not explicitly declared as group assignments must be completed individually and handed in individually. Identical submissions will be treated as plagiarism.

Submission until Monday, April 22nd at 11:59 pm via https://assignments.hci.uni-hannover.de/SoSe2024/ ISy. The submission must consist of a single zip file containing all necessary files. Please remove umlauts from file names.

Pack your solution (pdf file and zip file of the exported project) into a zip file for uploading.

Task 1: No Chrome (10 points)

In the lecture it was shown that No Chrome interfaces try to avoid widgets.

a) What is the philosophy behind "No Chrome"?

b) Compare the OpenOffice or Word application with the Sublime Text Editor. Which wid- gets are used in each case? How are different functions made available to the user?

c) Name 2 advantages and 2 disadvantages of No Chrome interfaces. Name a possible example of No Chrome interfaces from your everyday life.

Task 2: Loading bar (10 points)

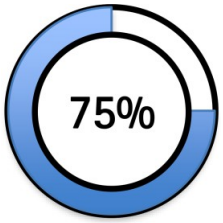
Progressbars in JavaFX and their feedback methods were presented in the lecture. Please read the following paper:

Harrison et al. Faster Progress Bars: Manipulating Perceived Duration with Visual Augmentations. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.
Available at: https://dl.acm.org/citation.cfm?id=1753556 Answer the

following questions about the paper:

a) Briefly describe the study design and which factors were investigated.

b) You have the task of designing a round progress indicator, as shown below, based on the results from the paper. The shape should not be changed for this. It is only about the presentation of the progress indicator (blue area). How would you present the progress indicator? How much faster (in %) can users theoretically perceive the loading bar?



Task 3: Listeners, Properties, Bindings (12 points)

In this task, the behavior of a dialog for setting fonts is to be implemented. The structure of the UI is predefined (see Stud.IP: FontSelect.zip). Use bindings wherever possible, otherwise use listeners. In the case of listeners, use lambda expressions and no anonymous classes. Try to write the code as short as possible. The following aspects should be implemented. (These aspects can also be found as comments in the source code of the template).

a) set font to focused list item

b) filter list as text is entered in search field (ignore case, show all fonts whose name contains the search text)

c) update the countView to show the number of fonts remaining in the list

d) update the percentageView to show the percentage of fonts remaining)

e) update font size as sizeSlider is modified

f) update underline through checkbox

g) update strikethrough through checkbox

h) update font color with radio buttons

i) update font weight with radio buttons

j) update font posture with radio buttons

k) update font blurring as slider is modified

The JavaFX documentation can be found at: https://openjfx.io/javadoc/11/ Export your solution as a zip file from IntelliJ.