CS 3053
Project – Menu Design
Due Tuesday 2018.03.13 at the beginning of class.

Overview

In this assignment you will design and implement a menubar and a toolbar for the main window of your application. In class, we talked about menu design and Hick's Law. We also talked about icon design and did a group exercise to design an icon for a common action. Out of class, you will use what you have learned to design and implement a menubar for your application. You will list and describe key functions that can be accessed from menu items, build the menus (without implementing most functions), estimate expected total selection time for three menu items using Hick's Law, and analyze your menu design in terms of several principles of usability. You will also design and implement a toolbar for your application. To do that you will choose a subset of functions, find icons for them, design the toolbar, and assess its layout.

<u>In this assignment, **all group tasks come before individual tasks.** Make sure to leave enough time for all of your team members to complete their individual tasks, especially implementation.</u>

Group Tasks

*<u>Complete **all** group tasks before proceeding to individual tasks. Your entire team **must** participate.</u>*

#1: Review the slides from class on Menus & Hick's Law and Icons & Semiotics. Drawing from the tasks you chose to support in the Window Design assignment, discuss the specific browsing and editing functions that you think are necessary and sufficient for your application to support.

#2: List and briefly describe **10** functions that you believe are appropriate to include somewhere within the menus of a menubar in the main window of your browsing application. Don't include any common functions (like Save, Print, Copy) typically included in File, Edit, and Help menus.

#3: Design and sketch a menubar with appropriate menus, submenus, and menu items. Include all 10 functions as menu items. Your menubar sketch must clearly include the following features:

- a File menu with Open, Save, Print, and Quit items;
- an Edit menu with Cut, Copy, and Paste items;
- a Help menu with an item that links to a (tasteful!) web page related to your project theme;
- at least one menu that is an item in another menu (*menubar–menu–menu–items*);
- at least one menu separator in at least two of the menus;
- appropriate mnemonics (keyboard shortcuts) for items in the File, Edit, and Help menus; and
- at least two menu items from your 10 that have mnemonics (keyboard shortcuts).

Briefly explain the main motivations behind your menu organization and analyze your design in terms of the usability principles of simplicity, familiarity, correctness, and generalizability. Annotate your sketch as needed to aid your explanation.

#4: Out of your 10 menu items, choose **3**, including one in the deepest level of menus. For each item, estimate total expected time to select it using Hick's Law (with Raskin's constants). For each calculation, state how you chose the number of levels of menu and how you chose `n` in the Hick's Law formula for each level. Use the time from each calculation to draw conclusions about how efficient your menu organization is for selecting the 3 functions. Speculate on ways to reorganize your menus for all functions, including the common ones, to be more efficient overall.

#5: Out of your 10 menu items, choose **5** that are **specific** to tasks that involve: (1) navigating through the items in the collection; (2) filtering to see a subset of the collection; or (3) editing the characteristics of items in the collection. Find an icon to represent each of the 5 items. You may use icons from any free public source including the large icon web archives. For each icon:

- Cite the icon source and provide a link to the icon itself.
- Does the icon represent an *object*, *attribute*, *function*, or *state*? Explain briefly.
- How is a user meant to interpret the icon to discern the menu item's purpose?
- Briefly describe a user who might have difficulty interpreting the icon, and why.

#6: Design and sketch a toolbar/iconbar for your 5 task-specific menu items. Use a least one spacer (separator) to group your icons appropriately. Assign a name to each toolbar item; each name can be the same or different from the name of the corresponding menu item, as you like. Referring to the "Using Icons in Design" slide, briefly analyze *location*, *grouping*, *juxtaposition*, and *consistency* in your toolbar **as a whole**. Label and annotate your sketch correspondingly.

#7: Decide which of your Window Design implementations, or combination of them, to adopt for your entire team moving forward. Integrate the necessary code into your team's shared build. This will be the build that you all start from for the individual tasks.

#8: Write up your work on each of the parts of tasks #1–#7. Compose this together. Be clear, objective, detailed, and thorough, yet succinct. *In grading we will be looking in particular for: thematically appropriate functions (#2); a complete, well-annotated menubar sketch followed by sensible design motivations and a well-reasoned usability analysis (#3); correctness of Hick's Law calculations including suitable choices of how to calculate (#4); constructive speculation of how to improve efficiency (#4); reasonable icon choices with helpful reflection about users (#5); a complete, well-annotated toolbar sketch followed by a well-reasoned design analysis (#6).*

Your writeup should be between 1.5 and 2.5 single-spaced pages of writing, not including your Hick's Law calculations. Use regular paragraphs and standard formatting (12 point font, 1 inch margins, etc.) Start the first page with a few lines stating your team number, name/logo, and list of member names. Attach scans/photos of your two sketches, followed by the details of your Hick's Law calculations. Refer to the sketches and calculations in your writeup appropriately.

To **turn in** your group work, go to the "Group - Menu Design" assignment in Canvas to submit your results as a PDF. Only one team member needs to turn in the group component.

Individual Tasks

All individual tasks must be completed entirely on your own.

#9: Start from a copy of your team's integrated build from task #7. Duplicate the stage4 main() class, call it `Stage5.java`, and modify `build.gradle` to have a new `createScript()` line. Whenever you build, the executable `stage5` should appear in `build/install/base/bin`.

I extended `Resources.java` with a new `getImage()` method to load an `ImageIcon` from an image file. You can use this method to easily create icons from image files located somewhere inside the `edu.ou.cs.hci.resources` package, a lot like you did for strings in the Personas & Scenarios assignment. See the `HelloPanel` constructor in `Stage5.java` for an example.

#10: Read about the `Action` interface and `AbstractAction` class in the Swing API. Using these capabilities isn't strictly necessary to complete the following tasks. However, it is likely to make your code much more modular, making it easier to organize, share, maintain, and extend. After implementing a set of `AbstractAction` subclasses, you can simply add them to `JMenu`s and `JToolbar`s. They automatically register themselves as `ActionListener`s when added. You can even reuse the same instance of any of your `AbstractAction` subclasses in multiple `JMenu`s, `JToolbar`s, and `JButton`s!

#11: Implement a menubar for your application's main window. Build the menubar with menus, submenus, and menu items in accordance with your team's menubar design. Use instances of the `JMenuBar`, `JMenu`, and `AbstractAction` (or `JMenuItem`) classes. Include all 10 of your team's functions and all of the features in the list in task #3. See below for what to do when the user selects Quit. For other items, print a message with the name and description of the item's function to the console when the user selects it, but don't actually implement its function.

#12: Implement a toolbar and add it to the layout of your application's main window. Populate the `JToolbar` with labeled icons using instances of `AbstractAction` in accordance with your team's toolbar design. Include separators and use the item names your team chose in task #6. Print a message with the name and description of the item's function to the console when the user selects it, but don't actually implement its function.

For tasks #11 and #12 you may create new classes and even packages. Organize them inside the `edu.ou.cs.hci.stages` package. Document your code thoroughly and appropriately.

When your app starts, show only your latest frame. When the user tries to exit by closing the frame, trigger your Quit action. Have your Quit action in turn trigger the action for each of your team's 10 functions to get them to print their messages in some reasonable order. Capture those messages in a `menu-actions.txt` file and put it in the `Results` subdirectory.

Take a screenshot of your frame while highlighting one of the items in the deepest level of your menus. Trim it, turn it into a PDF, and put it in the `Results` subdirectory as `screenshot.pdf`.

To **turn in** your individual work, *first test your project* <u>*using Gradle on the command line*</u> *to make sure it builds and runs as intended.* Run `gradle clean` to reduce the project size. Append your 4x4 to the `project` directory; mine would be `project-weav8417`. Zip the renamed directory. Submit your zip file to the "Individual - Menu Design" assignment in Canvas.