CS 4063/5063
Homework: Prototype D
Due Thursday 2021.04.07 at 11:00pm.

All homework assignments are individual efforts, and must be completed entirely on your own.

In this assignment you will continue to develop a movie collection application in JavaFX. You will learn how to write basic HTML and CSS, and use it to implement a window of simple information about the application. You will also learn how to use absolute positioning to create highly precise widget layouts, and apply that approach to create a more fine-tuned version of the Editor Pane. More generally, you will learn how to develop quality layouts that follow the Gestalt Principles of Design (GPoD) by engaging in an iterative, incremental design and analysis process.

Refining the Prototype

In the PrototypeA assignment, you implemented an `EditorPane` for editing the attributes of one movie in the collection at a time. To implement your pane design, you probably relied on the subclasses of `javafx.scene.layout.Pane` to define a hierarchy of panes and widgets. You probably also weren't able to converge on the exact layout you intended to recreate from your design. This is a common problem in implementing complicated, widget-heavy UIs like that of the `EditorPane`. An alternative approach is to use *absolute positioning* to specify the exact location (and sometimes also the exact size) of each component in a UI. Unlike relative layout approaches, which focus on supporting flexibility in screen size and user preference, absolute layout approaches target high memorability and efficiency through stability and precision. These qualities are beneficial for repetitive, complex data entry, especially in critical applications.

In this assignment, you will use absolute positioning to fine-tune the layout of the `EditorPane`. First, you will implement the spatially precise redesign of the `EditorPane` that you created and analyzed in the DesignD assignment. You will then experiment with precise layout adjustments with the goal of converging on a highly usable layout that stands up well to GPoD analysis.

Implementing the Prototype

Start by putting a copy of your `DesignD.bmpr` file in the `Results` directory. *(We need your design file for comparison with your prototype UI. You can create a design file to include now even if you didn't finish the DesignD assignment.)*

The implementation work in this assignment is split into two parts. The first part is to write HTML and CSS to implement your About Pane design from the DesignD assignment. To do that, you will edit the `index.html` and `style.css` files in the `about` directory inside the `prototyped` package. Before you start, review the slides from class on HTML+CSS, take the superb tutorials at https://www.w3schools.com, and refer to the reference pages there as you work. (Note that I have already added the necessary code to actually load and display your HTML+CSS inside a window. To show the window, select the Movies/About... menu item in the menubar of any view.)

The second part is to rewrite the layout code in a new `EditorPane2` class. It's an exact copy of the `EditorPane` class except for the `buildLayout()` method. The method provides a couple of examples of absolute positioning code to get you started. As you will discover, writing code in the absolute positioning approach is relatively easy. Most of the actual work is making repeated adjustments to positions, sizes, and styles to achieve the desired quality of layout. See the APIs of `javafx.scene.Node` and `javafx.scene.layout.Region` for helpful methods.

The `prototyped` package includes my "solution" to the PrototypeC assignment. You are welcome to use my designs, but I encourage you to use your own if they work well for you. **If you decide to use your own**, carefully replace the my widget, layout, and menu code with the code from your solution. Be careful to preserve new code that manages the About menu item and window in the `View` class. You will also need to copy all but your layout code from your `EditorPane` class into the `EditorPane2` class. Don't forget to copy over the `about` directory.

The places to add code are as follows:

#1 (in `index.html`) — Add HTML code to specify the content and layout of the About page.

#2 (in `style.css`) — Add CSS code to specify styling of elements in the About page.

#3 (in `View`) — Adjust the width and height of the About window to match your page design.

#4 (in `View`) — Change the background color of the About window to match your design.

#5 (in `EditorPane2`) — Implement your final DesignD layout using absolute positioning.

I recommend following the order above. You will probably work on #1 and #2 simultaneously. For #5, attention to detail is crucial. Keep readability in mind and document your code helpfully.

Take a screenshot of the window showing your About page. Put it in the `Results` directory as `about.png` or `about.jpg`.

Fine-Tune the Editor Pane Implementation

Before fine-tuning, take a screenshot of a window showing your *Editor2* tab in an informative graphical state. Put it in the `Results` directory as `editor-v1.png` or `editor-v1.jpg`.

The work here is much the same as in Design D. Start by reviewing the slides from class on the GPoD. After completing TODO #5, experiment with the positions, sizes, and styles of your UI components, both individually and in groups, in your `EditorPane2` layout code. Focus much more on bottom-up, fine-grained adjustments at the level of pixels, than on top-down shuffling of widget placements in the layout overall. Examine how visual relationships *within* and *between* different combinations of widgets *reinforce* and/or *interfere* with your ability to correctly identify how they are meant to be grouped functionally. As you do that, think about how well those combinations follow the principle, corollary, and guideline of each of the nine GPoD covered in class. Keep experimenting and refining until you are confident you have converged on a layout that seems likely to significantly increase the ability of most experienced users to remember the layout easily and perform editing tasks in it efficiently. *As before, this process is open-ended. A couple of hours will hopefully be enough to see noteworthy, if not substantial, improvements.*

After fine-tuning, take a screenshot of a window showing your *Editor2* tab in an informative graphical state. Put it in the `Results` directory as `editor-v2.png` or `editor-v2.jpg`.

Next, identify two specific aspects of your final layout that are likely to substantially increase memorability and efficiency for expert users, and one other aspect that is likely to decrease one or both of those things. For each of those two+one, analyze how specific design choices either follow or break particular GPoD. Factor at least five of the nine GPoD into your analyses. Write up a brief description (< 450 words) of your fine-tuning process and analysis. Put your writeup in the `Results` directory as `analysis.txt`.

Turning It In

Turn in a complete, cleaned, renamed, zipped **COPY** of your `PrototypeD` directory:

- Put a copy of your `DesignD.bmpr` file in the `Results` directory.
- Put your three screenshot files and `analysis.txt` file in the `Results` directory.
- Go into the `ou-cs-hci` directory.
  - Make sure it contains all of the modifications and additions that you wish to submit.
  - Run `gradlew clean` to reduce the size of your build.
  - If you're using an IDE, remove any IDE-specific files (such as the Eclipse `bin` directory).
- Append your 4x4 to the `PrototypeD` directory; mine would be `PrototypeD-weav8417`.
- Zip your entire renamed `PrototypeD` directory (including `About`, `Build`, and `Results`).
- Submit your zip file to the Homework - Prototype D assignment in Canvas.

These steps will make your submissions smaller and neater, which speeds up grading a lot.

To score the assignment, we'll be looking at how many elements in your About and Editor pane designs appear as components in your prototype; how well the prototype reflects those designs' layout and style; how clearly your code (Java, HTML, CSS) is organized and documented; the character and amount of improvement in your editor design; and the cogency, thoroughness, and depth of your process description and GPoD analysis. The maximum score is 20 out of 20.