CS 3053
Homework: Prototype B
Due Thursday 2019.03.07 at 11:59pm.

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

In this assignment you will learn how to develop an interactive application using JavaFX. You will learn how to write code to lay out controls in a scene, set the appearance of controls using FXCSS, modify the scene in response to interaction, and animate interaction transitions.

Learning about JavaFX and FXCSS

The slides from class on "App Design (JavaFX)" provide an overview of JavaFX capabilities. For a more in depth look, see Oracle's "Getting Started with JavaFX" tutorial here:

https://docs.oracle.com/javase/8/javafx/get-started-tutorial/index.html

The `PrototypeB` download includes several new examples inspired by the sample applications from the tutorial: `FXHelloWorld`, `FXCSSExample`, `FXIceSkating`, and `FXMediaView`. Try out the examples and have a look at the code. Also spend some time browsing the JavaFX API:

https://docs.oracle.com/javase/8/javafx/api/toc.htm

Learning new libraries is a major part of software development! Get a feel for what's available in the API before diving in to actual coding, and keep the API handy for reference while you code.

Exploring an Example

The `fxmvc` program displays an assortment of common JavaFX controls, using a wide variety of styling and layout options. It has the same MVC structure as the `swingmvc` program. Browse through the classes in the `edu.ou.cs.hci.application.fxmvc` package, starting with `Application.java`, to see how the `fxmvc` program works. Run it, interact with controls, and trace through the code to see how interacting with various controls affects the scene. (Also note how assets, including `.css` files, can be stored in packages alongside classes, as well as in the `resources` package for access via the `Resources.java` class).

Implementing your Prototype

In the DesignB assignment, you created mockups of tabs in an imaginary UI. In this assignment, you'll implement parts of the UI as a horizontal prototype. To make this easier, you'll start from a simplified version of the `fxmvc` code. The download provides a copy for you to work on; go into the `edu.ou.cs.hci.assignment.prototypeb` package and modify the files in it. Focus on `Model.java`, `View.css`, and the three `Pane` classes discussed below. You shouldn't need to change any other files. Compiling will create a script called `prototypeb` to run your program.

To complete the assignment, do any two of the following modifications. (If you do all three, your total score will be determined by the highest scoring two.)

[Difficulty: Easy] In the `MediaPane`, complete the control bar to reproduce the functions in the DesignB "trailer" tab: play/pause/stop, volume (as icon and slider), mute, time/duration, and rate (as a popup). Reproduce the *general* layout and style of the design, including the way that time/duration is also shown as a thin red-on-gray bar above the other controls. Connect the controls to properties of the `MediaPlayer` instance to let the user adjust playback interactively.

[Difficulty: Medium] In the `TablePane`, recreate the *general* layout, style, and functions of the in the DesignB "posters" tab. First, complete the left side to reproduce the accordion. Complete the right side to reproduce the movie information pane (feel free to use placeholder data for pane items that have no data). You should be able to complete both with an `Accordion`, some `Button`s or similar, and a few `Label`s. Second, modify the `TableView` to have four columns like in the design. The `list-movies-plus.txt` file has the necessary data. Third, connect up the parts to prototype basic functionality: (1) populate the table with data only when *Favorites/Documents* is selected; (2) populate the movie information pane with data about the currently selected item in the table ; and (3) allow the user to tag the selected item in the table with any subset of color tags. Use the MVC model to store the tag information for each movie in the list.

[Difficulty: Hard] In the `CoverPane`, reproduce the *general* layout and interactive operation of the cover flow in the DesignB "posters" tab. First, have the list of movies always progress from left to right, with the currently selected item always shown in the middle of the cover flow. Second, allow the user to navigate over the images interactively using the keyboard and mouse. With the keyboard, allow the user to step forward or backward by one, go to the beginning, or go to the end. With the mouse, allow the user to select any visible image. Following interactions, animate a transition to move the newly selected image to the center position. (Watch out for interactions that happen during transitions!) Third, add **new** effects (**not** like DesignB) to style the cover flow and the images in it. Have fun and feel free to be creative, but stick to reasonably appropriate transitions and effects for a sequence of movie poster images. Your **effects must be visibly different** from the ones that were copied over from the `CyclePane` example.

You may add Java files or even subdirectories for subpackages as you like. You probably won't need to. Organize your code inside the classes and CSS file as you like, but keep readability in mind. Avoid very long methods, group related methods into sections, and document your code helpfully. Also remove any unused code left over in the three `Pane` classes before you turn it in.

The main goal is to prototype the UI with enough functionality to replay the task flow from the DesignB assignment. Keep it simple, but feel free to experiment with effects and animations.

Turning It In

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

- Take a screenshot of your application window when it's in an interesting graphical state.
- Put the screenshot in the `Results` directory as `snapshot.png` or `snapshot.jpg`.
- 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 Eclipse, run `gradlew cleanEclipse` and delete the `bin` directory.
- Append your 4x4 to the `PrototypeB` directory; mine would be `PrototypeB-weav8417`.
- Zip your entire renamed `PrototypeB` directory.
- Submit your zip file to the Homework - Prototype B 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 the imaginary UI appear as controls in your prototype panes, how well the prototype reflects the design's *overall* layout and

general style, whether each interaction has the specified effect (including to modify values in the model and update views correspondingly), the utility and appeal of transitions and effects, and how clearly your code is organized and documented. The maximum score is 20 out of 20.