CS 4063/5063
Homework: Prototype B
Due Friday 2020.02.21 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 an MVC application using JavaFX. Specifically, you will learn how to model items in a collection, display the collection in a table, support single selection of an item in the table, and update an item summary in response to selection. You will also link the table to the `EditorPane` from PrototypeA to allow users to edit the selected item.

Prototyping Your Refined Design

In the PrototypeA assignment, you laid out and connected a variety of common widget types to build an `EditorPane` for editing the attributes of a single movie. In this assignment, you will build a `CollectionPane` for browsing a collection of movies, focusing on two key components: a `TableView` that shows the entire collection, and a sidebar **summary** of the selected movie.

You will start by completing a `Movie` class that represents all 17 of the editable movie attributes. Then, for the `TableView`, you will implement `TableCell` classes that render the contents of a `TableColumn` for each of the three attributes that you chose in DesignB. You will also support direct editing of attribute values inside cells of some columns using custom `EventHandler`s. Finally, you will implement your **summary** from DesignB to display data for the selected movie.

Preparing for Implementation

Prepare by browsing and reviewing the relevant packages and classes in 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 into actual coding, and keep the API handy for reference while you code. Start with the `TableView` , `TableColumn`, `TableCell`, `EventHandler` classes, then consult the `javafx.scene.control.cell`, `javafx.util.converter` packages as you proceed.

Implementing Your Prototype

In the DesignB assignment, you created a refined Collection wireframe. In this assignment, you will implement the **table** and **summary** areas of your wireframe as a horizontal prototype. Start by putting a copy of your `DesignB.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 DesignB assignment.)*

To make implementing easier, you'll add code to designated places. In the `Build/ou-cs-hci` source tree, go into the `edu.ou.cs.hci.assignment.prototypeb` package and modify the `Model.java`, `pane/EditorPane.java`, and `pane/CollectionPane.java` files.

The `EditorPane` class contains my "solution" to the PrototypeA assignment. You are welcome to use it, but I encourage you to use your own version if it works well for you. **If you decide to use your own version**, carefully replace the widget code in my `EditorPane` with the widget code from yours. Be careful to preserve the code that deals with movie data and the selection index! You will also need to modify your widget code to use the new property keys in `Model`.

The places to add code for your **table** and **summary** are as follows:

#0 (in `Movie`) — Add members for all attributes.

#1 (in `Movie`) — Create properties for all attributes.

#2 (in `Movie`) — Add access and modify methods for all attributes.

#3 (in `EditorPane`) — Update all attributes when the movie selection changes.

#4 (in `CollectionPane`) — Implement event handler classes for editing directly in cells.

#5 (in `CollectionPane`) — Implement cell classes for your three attributes.

#6 (in `CollectionPane`) — Implement factory classes for creating cells on demand.

#7 (in `CollectionPane`) — Implement column builder methods for your three attributes.

#8 (in `CollectionPane`) — Uncomment lines to add columns to the table.

#9 (in `CollectionPane`) — Implement your **summary** of the currently selected movie.

I recommend following the order above. For each of #0–#8, closely examine the nearby code provided for the *title* and *image* attributes. You can usually copy and modify that code to implement the other attributes. Keep readability in mind and document your code helpfully.

You shouldn't need to change any of the other classes. Compiling the build will create a script called `prototypeb` (in `build/install/base/bin`) for running your program. *As you test, keep in mind that the way we're implementing editing is still shallow. The panes should all show the same selected movie, but they won't share any results of local editing with each other.*

Turning It In

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

- Take a screenshot of a window showing your *Collection* tab in an interesting graphical state.
- Put the screenshot in the `Results` directory as `collection.png` or `collection.jpg`.
- Do the same for your *Editor* tab, as `editor.png` or `editor.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` <u>and</u> delete the `bin` directory.
- Append your 4x4 to the `PrototypeB` directory; mine would be `PrototypeB-weav8417`.
- Zip your entire renamed `PrototypeB` directory (including `About`, `Build`, and `Results`).
- 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 your refined design appear as components in your prototype, how well the prototype reflects the design's layout and style with five attributes, whether each selection and editing interaction has the expected effect, and how clearly your code is organized and documented. The maximum score is 20 out of 20.