CS 4063/5063
Homework: Prototype A
Due Tuesday 2020.02.04 at 11:00pm.

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

In this assignment you will learn how to develop basic Model-View-Controller applications using JavaFX and Gradle. Specifically, you will learn how to write code to lay out *nodes* in a *scene* on a *stage*, adjust their individual appearances and behaviors, handle different kinds of interaction events, manage a central shared data model, and compile and run applications using Gradle.

Learning about Gradle

All of the prototype assignments will involve writing in JavaFX and building and running using Gradle. Visit https://docs.gradle.org/current/userguide/userguide.html to get a sense of what Gradle does and how it works. You **don't** have to install Gradle on your system. You don't even have to do any Gradle scripting. I've done that for you.

In the `PrototypeA` download, go to the `ou-cs-hci` directory in `Build`. Read the `about.txt` file to learn about running Gradle tasks on the command line and the tasks you're most likely to need. Feel free to take a peek at the `build.gradle` script file. The *Alternative Start Scripts* section at the end describes how to create additional scripts to run alternative main() classes. You shouldn't need to, but the possibility exists. Don't otherwise change `build.gradle`!

Using Gradle with Eclipse

Gradle and Eclipse play fairly well together. First, make sure you're using a version of Eclipse with the BuildShip plugin installed. Most recent Eclipse for Java distributions come with it. Second, run `gradlew eclipse` in the `ou-cs-hci` directory to prepare it for import into Eclipse. Third, import the `ou-cs-hci` directory as a *Gradle / Existing Gradle Project* using the Eclipse import wizard. Eclipse will add a *Gradle Tasks* pane to the editing window for running Gradle commands. (Other IDEs are less well supported, but can still be used to edit code.)

Building and Running Applications

To compile the build, type `gradlew installDist` on the command line. You need to be in the `ou-cs-hci` directory for this to work. Then go into the `build/install/base/bin` directory and run any of the resulting programs by typing its name. Each program has a `.bat` version for running on Windows. (Note that the `gradlew run` shortcut only launches the `base` program!)

The source code packages are organized to hold code for your prototype assignments as well as example applications that we'll see in class. The download for each prototype assignment will provide an updated `ou-cs-hci` directory with all necessary classes and program scripts.

Exploring an Example in JavaFX

The `fxmvc` program shows off JavaFX's capabilities. It provides examples of creating, styling, laying out, and connecting components in a simple MVC application. *In this assignment, focus on the Simple and Gallery panes.* Review the slides on common JavaFX widgets, MVC, and the View Lifecycle. Study the classes in the `edu.ou.cs.hci.application.fxmvc` package, starting with `Application.java`. Run it, interact with controls, and trace through the code to see how interacting with various controls affects the scene. Notice how some components are effectively coupled to each other by depending on the same data value in the model.

Also notice 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. (Bundling assets in Java applications isn't always straightforward. The `edu.ou.cs.hci.resources` package is designed to make it easier. Put any files you need in subdirectories of that package. Gradle is configured to copy any non-`.java` files in the source tree into the same places as the compiled `.class` files. This means that your assets will accompany your application and can be easily accessed by your program, regardless of where it is running. The `fxmvc` code has several examples of accessing both image and text files this way.)

Implementing your Refined Design

In the DesignA assignment, you created a Refined mockup of a movie metadata editor. In this assignment, you will implement that wireframe as a horizontal prototype using JavaFX. Start by putting a copy of your `DesignA.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 Design A assignment.)*

To make implementing easier, you'll modify a simplified copy of the `fxmvc` program code. In the source tree in `Build/ou-cs-hci`, go into the `edu.ou.cs.hci.assignment.prototypea` package and modify the classes in it. Focus on `Model.java` and `pane/EditorPane.java`. You shouldn't need to change any of the other classes. Compiling the build will create a script called `prototypea` (in `build/install/base/bin`) for running your program.

The goal is to prototype the widgets, layout, and shallow interactivity of the widgets in your Refined mockup. At this stage of the design process, reproduce the *general* style of the design, but don't bother with fine details. Don't include your mockup Comments. Don't forget to refer to the code in the Gallery pane of the `fxmvc` code if you need help implementing various widgets!

Feel free add Java files or even subdirectories for subpackages as you like. You probably won't need to. Organize your code inside the classes 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, left over code before you turn it in.

Turning It In

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

- Put a copy of your `DesignA.bmpr` file in the `Results` 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 code 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 `PrototypeA` directory; mine would be `PrototypeA-weav8417`.
- Zip your entire renamed `PrototypeA` directory.
- Submit your zip file to the Homework - Prototype A 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 *overall* layout and *general* style, whether each interaction has the expected effect (to modify a value in the model and update the view correspondingly), and how clearly your code is organized and documented. The maximum score is 20 out of 20.