Module 5 Assignment

Description

There are two distinct sections to this assignment, each weighted equally. In the first you will create a style sheet to apply style information to an application that I have provided. In the second you will create a customer-styled "launcher application" which doesn't have any real functionality (yet).

As you go through this assignment, please keep in mind that we covered only a few of the JavaFX properties that can be styled. Explore the JavaFX CSS reference (see Resources) to get a more complete list. I do not recommend using Stack Overflow as a resource unless you can't find what you need in the reference. There is a lot of helpful information there but many of the examples posted there will break one or more of the constraints that I'm placing on you.

First steps

Make sure you've exited IntelliJ completely. Using the command line, in your solutions directory,

1. switch to the main branch and merge in your module4 branch.
2. create (and switch to) a branched name module5.

Start IntelliJ and create a new IntelliJ project using the JavaFX template. Name the project module5 and place it inside your solutions directory. **IMPORTANT:** disable the "create git repository" checkbox since you've already created a repository.

The JavaFX templates creates several files (two .java files and an fxml file). Delete all three of these files before proceeding.

Be sure to commit and push regularly as you work through this assignment.

Constraints

In both parts of this assignment you must adhere to these basic constraints. These constraints are in place in order to enforce good separation between cosmetic styling, layout and functionality and would similar to practiced followed on the typical professional project.

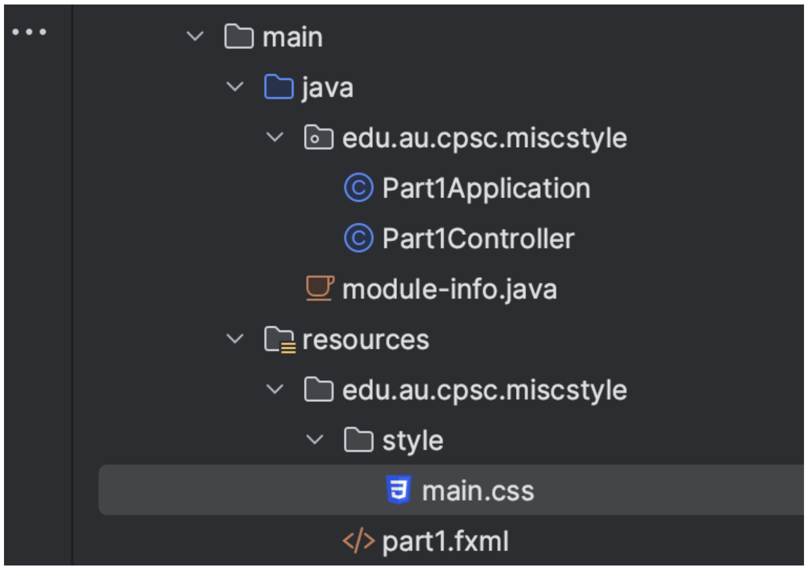
1. Do not use the Node.setStyle() method.
2. Do not call Node methods that impact the appearance of the your controls beyond their layout.
3. Do not modify any of the properties in FXML files that impact appearance of layouts or controls beyond basic layout. That is, do not modify the border, background, color, font or any similar cosmetic property. If you already made

such changes in FXML, go through each control and set these back to their defaults. You **are** permitted to include FXML settings that only impact the layout of a control (such as sizes and padding) even those many of these *can* be controlled from CSS.

Note: you can't get two developers to agree on what to place in CSS versus what to place in FXML. The separation I've made here is close to one that I use in practice and has served me well over the years. You might have difference preferences yourself but please adhere to these constraints for this assignment.

Part 1: Style basics (50 points)

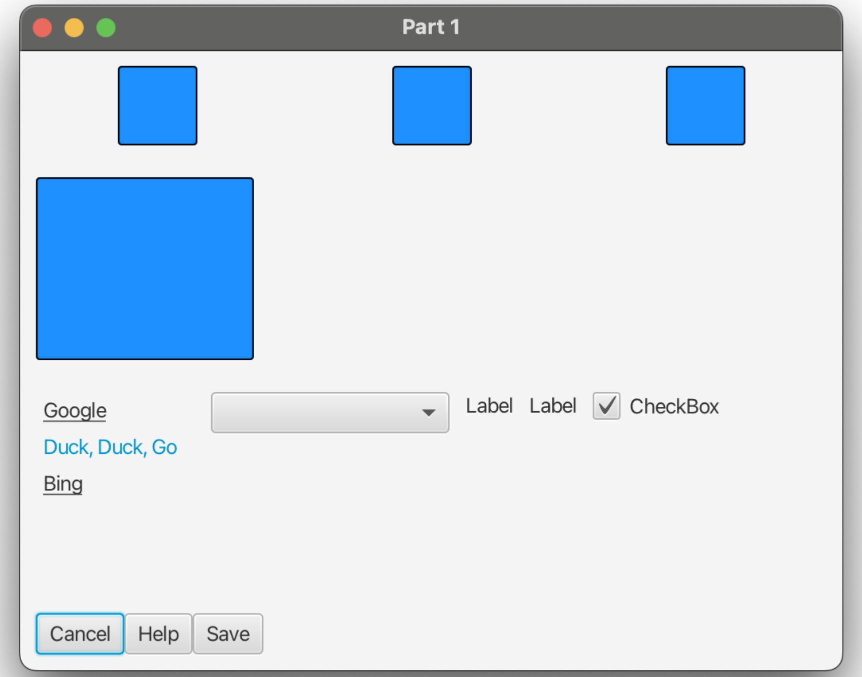
I'll guide you through the steps to get set up but they're pretty straightforward so, if you'd prefer to figure it yourself, here is the project structure you will have once these steps are completed:



1. Create a package named edu.au.cpsc.miscstyle.
2. Create a resources directory named edu/au/cpsc/miscstyle (note, these are directories, not packages even though IntelliJ displays them in a more package-like notation). To create this directory, right-click on Resources, select New->Directory then type the complete path /edu/au/cpsc/miscstyle.
3. Download Part1Application.java Download Part1Application.java, Part1Controller.java Download Part1Controller.java and part1.fxml Download part1.fxml and move the Java files into your package and the FXML file into the resources directory that you created (you can drag-and-drop the files from Windows File Manager/MacOS Finder).
4. Add the following lines to your module-info.java just before the closing brace:

```
opens edu.au.cpsc.miscstyle to javafx.fxml;
exports edu.au.cpsc.miscstyle;
```

5. Create a directory called style inside resources (full path edu/au/cpsc/miscstyle/style) and create a file named main.css inside this style directory.
6. Verify that you can run Part1Application. The following window should appear:



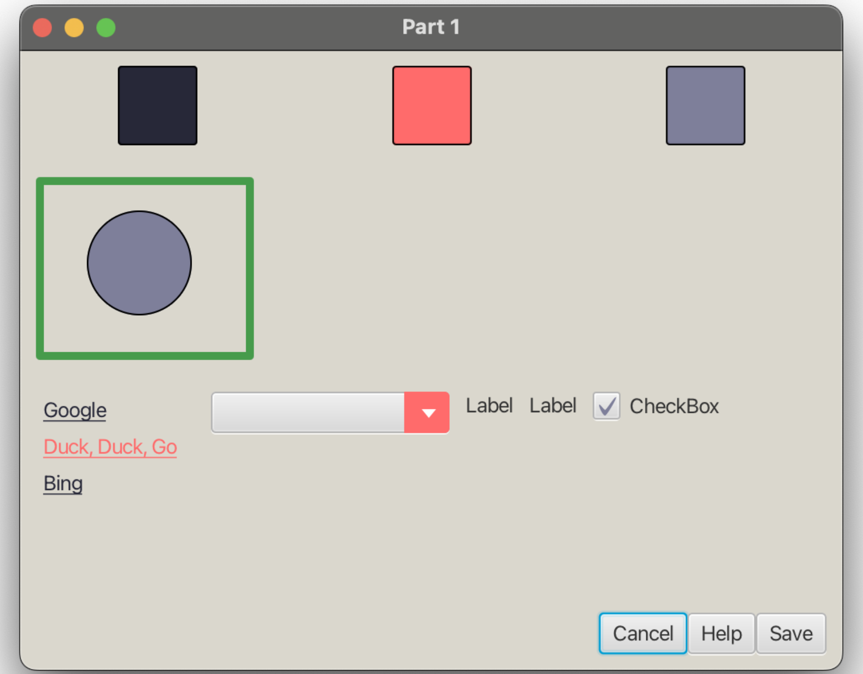
Throughout this project, do not modify any of the three supplied files. All of your work will be in main.css.

OK, now take a few minutes and look through the FXML file to see element ID's and CSS classes. You can open it in Scene Builder, if you prefer, or just read the XML directly. Add style information to your main.css file in order to accomplish the following visual changes:

1. Set the background color of the element with CSS ID background to #dad7cd.
2. Set the fill color of the elements with ids foreground, highlight and accent to #272838, #ff6b6b and #7e7f9a respectively.
3. Create a border around the Rectangle inside a Group inside a VBox. The border should be 5px wide, stroked inside the rectangle's bounds and have a color with RGB components 70, 155, 75.
4. Set the fill for the Rectangle in step 3 to a completely transparent color (so you can see through it). Note: you cannot use -fx-opacity here because that will also make the border of the Rectangle transparent.

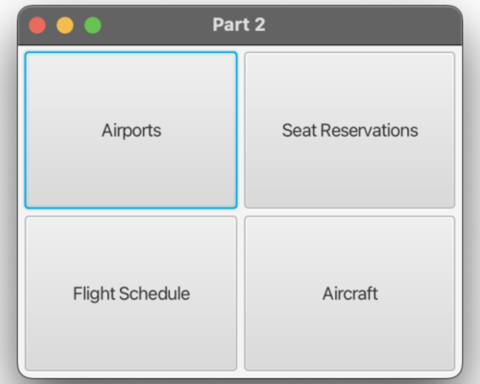
5. You should now be able to see a Circle behind the Rectangle. Set its fill color to #7e7f9a.
6. There are three Hyperlink objects in the scene. Two of them (Google and Bing) have been set to the "visited" state. Modify all Hyperlinks to have the following properties:
    - Text color #ff6b6b and underlined (notice that non-visited Hyperlinks are normally not underlined).
    - Text color for visited Hyperlinks #272838.
7. ComboBoxes are an example of elements with substructure. Look the the JavaFX CSS reference's description of this control. (You might have to look at one of its superclasses to find the substructure...just follow the links!). Notice that it has two sub-elements one with CSS class arrow-button and one with CSS class arrow. Change the background color of all arrow buttons on all ComboBoxes to #ff6b6b and the background color of all arrows on all ComboBoxes to white. Do not impact elements with CSS class arrow that are not in ComboBoxes. I have placed two labels next to the ComboBox that have these CSS classes and these labels should not be impacted by your changes.
8. CheckBoxes also have substructure. Study the documentation and change the background of the checkmark to #7e7f9a.
9. There is an HBox (CSS id bottom-row) containing three buttons. Modify the alignment of the elements in that HBox so that they are centered vertically and right-aligned horizontally.

When you are done the window should look like this:



Part 2: Launcher (50 points)

For this part you will need a package named edu.au.cpsc.launcher. Create this package and the corresponding resources directory (following similar steps to those in part 1). Create classes LauncherApplication, and LauncherController, an FXML file named launcher-app.fxml, and a CSS file named main.css (place the CSS file in a directory named style, as usual). Modify LauncherApplication so that it loads your FXML and opens the primary stage. (You can copy and modify code from another Application.) Your controller should just be an empty class. Edit the FXML (in Scene Builder) so that you get a 2x2 grid with 4 buttons looking something like:



Assign a CSS id to each button: airports-btn, seat-reservation-btn, flight-schedule-btn, aircraft-btn. Add your main.css style file as a stylesheet for the root GridPane. Style this application :) You have some latitude but there are some requirements, of course. You must:

1. make use of a downloaded font which must be included properly in your resources directory. Use this font on your buttons. You can increase/decrease the font size as you see fit. As you change the font size, make sure that your application looks OK when it is launched (you might need to modify LauncherApplication to get the size right.
2. use at least 4 image assets (download icons from your favorite site or create your own!). When I was toying with my solution I did the following:
    1. Added icons to the buttons.
    2. Made the buttons partially transparent (via -fx-opacity).
    3. Used an "image fill" background to place a logo image behind the buttons which was partially visible through them. Set the opacity of this filled image so that it wasn't overwhelming.

    The result looked hideous providing further evidence that I should never be put in charge of the appearance of an application :(Feel free to use these images any way you like but if you can't think of anything you're welcomed to do the same thing I did.

That's it! Feel free to add whatever else you like. For example, if you prefer to have a logo image at the top of the window you can add an ImageView to your GridPane but be sure to set the image from within CSS!

From a grading point of view we're simply going to run your program and bask in its beauty. If it runs and you meet the 2 requirements above, you'll get full credit.

Submitting

Commit and push one last time and verify that your files look OK on GitHub (make sure you switch to the correct branch when looking at it on GitHub). Next submit this assignment on Canvas by pasting the URL of your GitHub repository into the supplied input. This will be the same URL that you submitted in Module 1. Note that Canvas will try to display a preview of it but that preview may show an error since the repository is private. Don't worry, as long as your URL is correct, everything is OK.