Module 3 Assignment

Description

In this project, you will create a user interface for displaying airport information. Since we haven't covered ListView or TableView yet, the user will enter an ID for an airport and it will display it.

First Steps

Using the command line, in your solutions directory:

1. Switch to the main branch and merge in your module2 branch.
2. Create (and switch to) a branched name module3.

Create a new IntelliJ project using the JavaFX template. Name the project module3 and place it inside your solutions directory. IMPORTANT: disable the 'create git repository' checkbox since you've already created a repository.

Domain Model

Download the CSV format database of airport information. Open the file in a spreadsheet program (Excel, Numbers, Google Sheets, etc.). Create a Java class named Airport that has one instance variable for each column except the last, which you should break into 2 instance variables. Use Java naming conventions so, for example, the column named elevation_ft should result in an instance variable named elevationFt or elevationInFeet. Choose reasonable types for each column. For example, the elevation_ft column contains integer values but some are missing, so int would not work as a type, but Integer would (since ints cannot be null but Integers can). For the coordinates column, split it into two instance variables: coordinatesLongitude and coordinatesLatitude, in that order.

Use IntelliJ to generate getters and setters for all of these variables.

Write a static method with the following signature:

public static List<Airport> readAll() throws IOException

This method should read the contents of this file (make sure it is placed in the correct location as discussed in the previous module) and return a list of airports as a result. An internet search engine will help you find instructions for reading CSV files. If that doesn't help, try ChatGPT or similar. (Note: do not use ChatGPT to generate your entire class, this would be a violation of our class policies, just for help on this method.)

Data User Interface

Use Scene Builder to create a user interface with two sections: Data and Map.

The Data section should have a cleanly laid-out display of only the following fields (note that the first three fields are search fields so you might want to display them in their own group):

- ident -- read/write TextField
- iata code -- read/write TextField
- local code -- read/write TextField
- type -- read-only TextField
- name -- read-only TextField
- elevation -- read-only TextField
- country -- read-only TextField
- region -- read-only TextField
- municipality -- read-only TextField

Note that you might want to read the rest of this homework description before you commit to your layout. I recommend sketching it before writing code.

When the user presses enter in any of the first three TextFields, search for the corresponding airport and fill in all of the fields with their correct values. In addition, update the Map to display the airport (Map information follows).

Add a button labeled 'Search'. This button operates similarly to the input fields; clicking on it searches based on the first non-blank input field it finds. For example, if ident is blank but iata code and local code have text in them, it would search via iata code.

Map User Interface

Add a WebView to your user interface. Whenever you update your TextFields to display an Airport, show a weather view of the site. For example, if the user entered PIT in the iata code input, display Pittsburgh International Airport, which is at latitude 40.49150085, longitude -80.23290253. Our program would tell the web view to go to the URL:

https://www.windy.com/?40.49150085,-80.23290253,12

Note that the last number here (12) is a 'zoom level.' The free version of windy has limited data at higher zoom levels, so feel free to modify this as you see fit. Also, note that the JavaFX WebView does not support WebGL, so some of the nicer features of this weather service will not work.

As you test your application, if you don't see the correct location on the map, perhaps your latitude and longitude variables are reversed. In the CSV file, the order is longitude then latitude, but in Google's map API, they are in the opposite order.

Layout Resizing

Your application's default window size should be large enough for all of the text fields to be displayed with at least a reasonable-sized map view. When the user increases the size of your application window, only the map view should increase in size. Can you figure out how to specify a minimum size? It wasn't covered in lecture, but search engines should be able to help you.

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.