CSC 335 Project 4: Cryptogram GUI

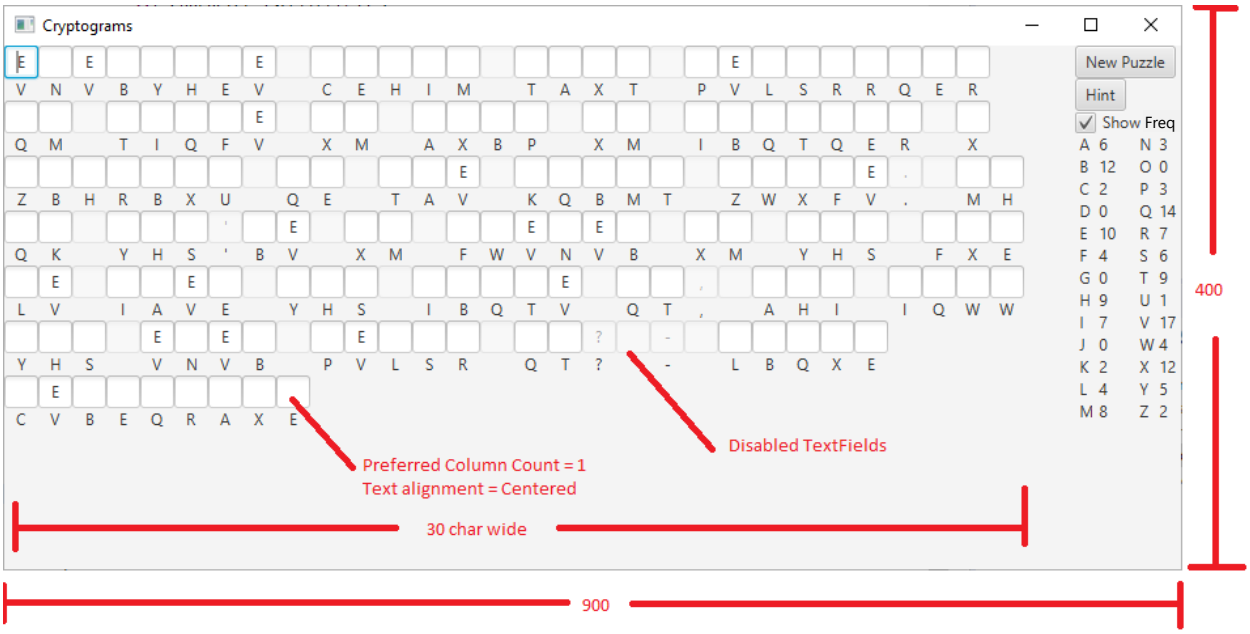
Due: Wednesday, November 3, 2021, by 11:59pm

GitHub Classroom Link: https://classroom.github.com/a/rImEmBO0

Project Description

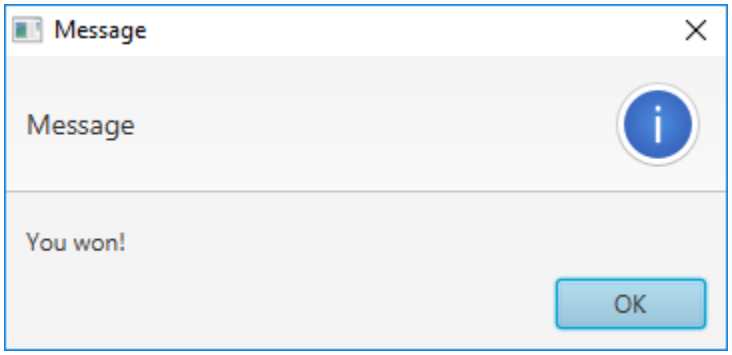
For this project, we will use JavaFX to create GUI for our Cryptogram game.

We will construct a new `CryptogramGUIView` that is a `javafx.application.Application` and use a `GridPane` to hold as many `VBox` objects as needed to represent our input spaces: (`TextFields` with a Preferred Column Count of 1) and the encrypted letter (`Label` centered in the `VBox`).



Lay out the main `Stage` as a `BorderPane`.

When you win, display a modal (`showAndWait()`) Alert:



After the game is over, do not allow any further editing until “New Puzzle” is clicked.

Implementation

Main Board

As mentioned above, your game board will be a `GridPane` with 30 columns and as many rows as needed to display the randomly-selected quote. For each cell of the `GridPane`, add a `VBox` that is centered and holds a 1-char `TextField` on top of a `Label` with the encrypted quote’s letter for that position. If the quote has a space or non-letter character, put the text in both the `TextField` and the `Label`, and disable the `TextField` so it cannot be edited.

Add this `GridPane` to a `BorderLayout`’s center position. In the right position, add buttons to create a new puzzle (with a newly selected random quote), get a hint (fill the appropriate boxes with the hint answer), and a checkbox that allows you to see the letter frequencies in the encrypted quote. If the box is checked, display a table of letter frequencies as shown above, otherwise hide it. Start with the table hidden and the checkbox unchecked.

Event Handling

You will add a `KeyTyped` handler to each `TextField`. When you type a letter in a `TextField`, all other `TextField`s that represent the same encrypted letter should be changed. This change should not be done in the event handler for the `TextField`, but rather via the Observer/Observable pattern as described below. Have your View change the Controller, which in turn will change the Model, that will request the view update itself.

Do not let multiple letters into a box. You may wish to “eat” the `KeyTyped` event that you handle so the `TextField` does not add the typed letter if you have already added it. You can do that by `consume()`ing the event in your event handler.

MVC

You are required to create this version of the project using the MVC architectural pattern. You must have the following 5 classes:

1. `Cryptogram` – This is the main class. When invoked with a command line argument of “-text”, you will launch the text-oriented UI. When invoked with a command line argument 0f “-window” you’ll launch the GUI view. The default will be the GUI view.
2. `CryptogramGUIView` – This is the JavaFX GUI as shown above
3. `CryptogramTextView` – This is the UI that we built in project 2
4. `CryptogramController` – This class contains all of the game logic, and must be shared by the textual and graphical UIs. You may not call into different controllers from the different UIs and all methods provided must be useful to **both** front ends.
5. `CryptogramModel` – This class contains all of the game state and must be also shared between the two front ends.

Observer/Observable

For the GUI front end, when the controller changes the model, we'd like to change the view. The model can do this for us, by notifying us when the model changes if our view is an `Observer` and the model is an `Observable`.

Have your model class extend `java.util.Observable` and have your view class implement `java.util.Observer`. As a requirement to the `Observer` interface, you will need to implement the `update(Observable o, Object arg)` method in the View. This method should change the `TextField`s of **ONLY** the letters that represents the most recent guess. You can learn which `TextField`s to update by passing the new guess via the `Object arg` parameter. This is set by the model calling `setChanged();` and `notifyObservers(Object arg)`. Whatever we pass to `notifyObservers()` will become the `arg` parameter of `update()`.

Requirements

- A main class, that launches your view using :

  ```
  Application.launch(CryptogramGUIView.class, args);
  ```

  for the GUI version or launches the text version, depending on the commandline argument given
- At least the five (5) classes described above, with more as you deem necessary (for instance, the code that breaks the quote into lines may be in its own object to more easily share between the two front-ends)
- You are **not** required to use `ArrayMap` any longer. If it didn't quite work, that's fine, you may use `HashMap` for your `Map`-ping needs.
- The GitHub repository is once again empty, so create a new Java Project in eclipse and import what files from the previous projects you'd like to start with

Submission

As always, the last pushed commit prior to the due date will be graded.