Homework 09 – Jordle

Authors: Daniel. Adapted from Helen, Ruchi, Sooraj, and Chelsea's assignment!
Topics: JavaFX

Problem Description

In this assignment, you will be implementing your own version of Wordle using JavaFX!

For this homework, there will be a few basic requirements, but the UI and design of the game is largely up to you! Feel free to explore some fun aspects of JavaFX. This homework is intentionally more open-ended, and we will grade most of it manually.

Solution Description

You will need to complete and turn in 1 class: `Jordle.java`. This class will be the base for your GUI and you will need to meet the requirements listed below. Feel free to create additional classes to help make your code more readable and modular, but the base class for your GUI must be `Jordle.java`.

Provided Files

We have provided you with three files to help you implement Jordle: `Backend.java` `InvalidGuessException`, and `words.txt`.

- `Backend.java`

  This class represents the backend implementation of your Jordle game and contains the logic used in checking the correctness of the player's guesses. There are four public methods you should know:
  - Constructor
    - The Backend class has one no-arg constructor which initializes a random 5-letter target word from the Jordle word bank `words.txt`. The target word is a private instance field of Backend.
  - `reset()`
    - This method sets a new target word in the backend of the Jordle game by picking a random 5-letter word from `words.txt`. The constructor calls this method once to set an initial target word, but you should call this method whenever you need to set a new target word.
  - `check(String)`
    - This method takes in a String, which is a word that you want to guess. If the word is an invalid Jordle guess, this method throws an `InvalidGuessException`. If the word is valid, the method checks the correctness of the guess against the target word by returning a five-letter String only containing the characters 'g', 'y', or 'i'.
      - A 'g' means that the correct letter is in the correct place.

        - A 'y' means that this letter is in the target string, but in an incorrect position.
        - A 'i' means that this letter is not in the target string, and is an incorrect guess.
    - For example, if the target word is "trace", guessing "adieu" will return "yiiyi". Guessing "brace" will return "igggg". Guessing "trace" will return "ggggg".
    - This method is case-insensitive on its inputs. For example, if the target is "trace", then inputs "trace", "TRACE", "TraCe", and so on, will all return "ggggg".
  - `getTarget()`
    - This is a getter for the target word.

- `InvalidGuessException.java`

  This class is a **checked** exception which is thrown by `Backend`'s `check` if the input word is not valid. An invalid guess is a String that is null, empty, blank, or has a length not equal to five characters. For example, "adieu" is a valid guess, but "", " ", "jordle", or "hi" are not.

- `words.txt`

  This text file is the word bank for your Jordle game. This file contains 29 five-letter words, each one on a new line. Feel free to add your own five-letter words to the word bank! When you are testing your Jordle application, make sure that words.txt is in the same directory as Backend.java.

JavaFX Details

- To compile a JavaFX program, run this command:
  - `javac --module-path javafx-sdk-11.0.2/lib/ --add-modules=javafx.controls FileName.java`
- To run a JavaFX program, run this command:
  - `java --module-path javafx-sdk-11.0.2/lib/ --add-modules=javafx.controls FileName.java`
- Modify the module path if your JavaFX download is not located in the same folder.

Jordle.java

Write a JavaFX application class called `Jordle`. This class must meet the following minimum requirements:

- Create a window with the title "Jordle." The window should be sized appropriately, such that contents within the window are not cut off.
- The application will have two screens – the welcome screen and game screen.
    - **Welcome Screen**
        - On this screen, there should be a non-editable display of text that says "Jordle".
        - There should be an appropriate background image that relates to Jordle. Name this file `jordleImage.jpg`.
            - If you don't want to find your own image, we have provided one for you.
            - **Make sure to use relative paths for images in your application.**
        - There should be a button that says "Play", which when pressed, takes the user to the game screen.
    - **Game Screen**
        - This is where Jordle will be played.
        - You should create an instance of a **Backend**.
        - You should Include an instructions button that, when clicked, opens a **new window** with a list of written instructions for how to play Jordle.
            - Opening the new window should not close the game screen.
        - Include a **6-by-5 grid** where the user will have 6 opportunities to guess 5-letter words.
            - The grid should read **key input** from the user.
                - Typing keys should populate the cells in the current row of the grid one by one, left to right (like in a regular game of Wordle).
            - Pressing numbers or special characters should not do anything.
            - Pressing backspace should remove one character at a time.
            - Pressing enter should evaluate the user's 5-letter guess against the actual word.
                - If the correct letter is in the correct place, the cell should turn a certain color (for example, green).
                - If the letter exists in the target word, but in the incorrect place, the cell should turn a different color (for example, yellow).
                - If the letter does not exist in the word, the cell should turn a different color from the default background (for example, grey).
            - Pressing enter when a 5-letter word is not supplied should **alert** the user that their guess is invalid.
        - Include a restart button that, when clicked, resets the grid and resets the Backend, generating a new target word.
        - Include a text label that lets the user know about the status of their game:
            - Contains a default message (for example, "Try guessing a word!")
            - Notifies the user of the correct word if they lose after 6 guesses (for example: "Game over. The word was {word}.").

  - Congratulates the user when they win (for example, "Congratulations! You've guessed the word!").
  - Changes back to the default message when the user clicks restart.
- When creating your implementation, use at least **one anonymous inner class** and **one lambda expression** in your implementation. (This is required for full points).

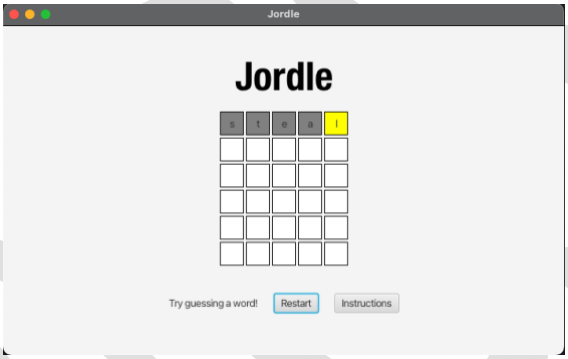
Screenshots

Here are images of a Jordle which correctly meets the requirements above. You may style your game just like this, or differently, but make sure to fulfill the requirements above.

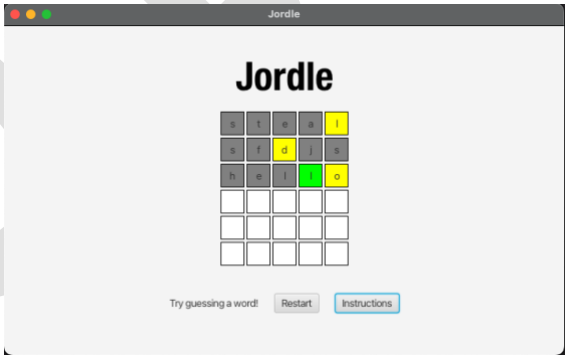
Welcome screen

Separate instructions window

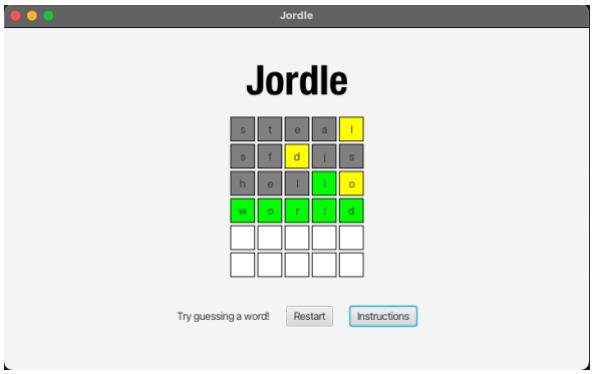
First guess



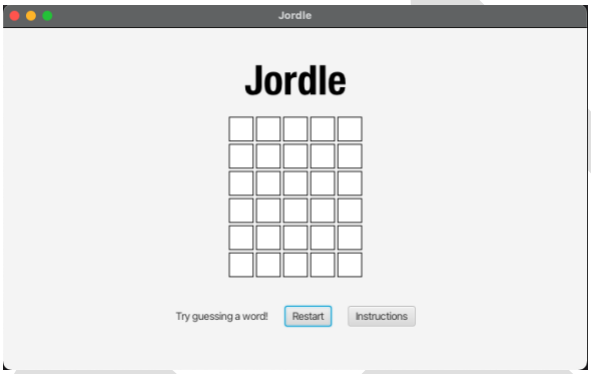
Third guess



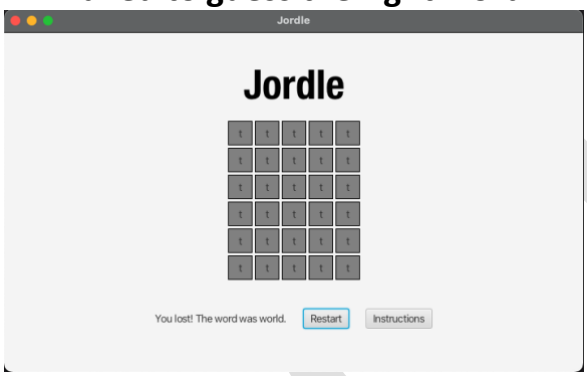
Guessed correct word



Clicked restart



Failed to guess the right word



Alert not 5 letters

JavaFX Tips and Tricks

- The Java API is your friend. Use it to your advantage.
- Make sure you are able to properly run your JavaFX programs before starting this assignment. Do **NOT** wait until the last minute to configure your JavaFX!
- Work incrementally. Get a basic UI up and running. Add buttons, cells for display, etc., piece by piece. Think of what layout manager(s) you want to use.

Extra Credit

As this homework is relatively open-ended, there will be up to **30 bonus points available** for expressing creativity through your implementation.

NOTE: If you decide to do extra credit, also submit an **`extra_credit.txt`** file detailing what extra credit you decided to implement (a few sentences or so). We will **NOT accept regrades or grade extra credit** if a .txt file is not provided!!!!!!

If your code requires a different command to compile and run (i.e., using different javafx modules), then put the command in the extra_credit.txt.

All these items are subjectively graded. Additions will be considered under the following categories. You may earn points under a certain category multiple times:

THIS GRADED ASSESSMENT IS NOT FOR DISTRIBUTION.
ANY DUPLICATION OUTSIDE OF GEORGIA TECH'S LMS IS UNAUTHORIZED

- 5 – Non-trivial enhancement to graphics of the program
- 5 to 15 – Non-trivial enhancement to the controls of the program
- 5 to 15 – Non-trivial new feature added

Some examples include, but are certainly not limited to:

- Ability to toggle between light/dark mode, different colors for blocks
- A "statistics" window that shows the distribution of number of guesses required to guess words or win percentage
- Audio feedback for correct and incorrect letter placements
- Adding appropriate images and/or graphics for when the user guesses a word correctly

Feel free to be creative; these are just a few ideas!

Checkstyle

You must run Checkstyle on your submission (to learn more about Checkstyle, check out cs1331-style-guide.pdf under the Checkstyle Resources module on Canvas). **The Checkstyle cap for this assignment is 50 points.** This means there is a maximum point deduction of 50. If you don't have Checkstyle yet, download it from Canvas → Modules → Checkstyle Resources → checkstyle-8.28.jar. Place it in the same folder as the files you want to run Checkstyle on. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar YourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the number of points we would take off (limited by the Checkstyle cap). In future assignments we will be increasing this cap, so get into the habit of fixing these style errors early!

Additionally, you must Javadoc your code.

Run the following to only check your Javadocs:

```
$ java -jar checkstyle-8.28.jar -j yourFileName.java
```

Run the following to check both Javadocs and Checkstyle:

```
$ java -jar checkstyle-8.28.jar -a yourFileName.java
```

For additional help with Checkstyle see the CS 1331 Style Guide.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Jordle.java`
- `jordleImage.jpg`
- `extra_credit.txt` (optional)
- **Any other files needed for your code to compile and run**

Make sure you see the message stating the assignment was submitted successfully. From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder tests are provided as a courtesy to help "sanity check" your work and you may not see all the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or Professor via the class forum for clarification.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the assignment. We will only grade your latest submission. **Be sure to submit every file each time you resubmit**.

Gradescope Autograder

If an autograder is enabled for this assignment, you may be able to see the results of a few basic test cases on your code. Typically, tests will correspond to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue. **We reserve the right to hide any or all test cases, so you should make sure to test your code thoroughly against the assignment's requirements.**

The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g., non-compiling code)
- Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or Checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Burden of Testing

You are responsible for thoroughly testing your submission against the written requirements to ensure you have fulfilled the requirements of this assignment.

Be **very careful** to note the way in which text output is formatted and spelled. Minor discrepancies could result in failed autograder cases.

If you have questions about the requirements given, reach out to a TA or Professor via the class forum for clarification.

Allowed Imports

- Any imports from any package that starts with `java` or `javafx` that do not belong to the AWT or Swing APIs.
    - ***IMPORTANT: You are NOT allowed to use FXML or any of the associated classes or packages.***
- Note that JavaFX is different than AWT or Swing. If you are trying to import something from `javax.swing` or `java.awt`, you are probably importing the wrong class.
- The only modules you should use are javafx.controls and javafx.media.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our autograder. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`
- `System.arraycopy`

Collaboration

Only discussion of the assignment at a conceptual high level is allowed. You can discuss course concepts and assignments broadly; that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going too far. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs.

The only code you may share are test cases written in a Driver class. If you choose to share your Driver class, they should be posted to the assignment discussion thread on the course discussion forum. We encourage you to write test cases and share them with your classmates, but we will not verify their correctness (i.e., use them at your own risk).

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items.
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope.
- Submit every file each time you resubmit.
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points.
- **Check on Ed Discussion for a note containing all official clarifications and sample outputs.**

It is expected that everyone will follow the Student-Faculty Expectations document and the Student Code of Conduct. The professor expects a **positive, respectful, and engaged academic environment** inside the classroom, outside the classroom, in all electronic communications, on all file submissions, and on any document submitted throughout the duration of the course. **No inappropriate language is to be used, and any assignment deemed by the professor to contain inappropriate offensive language or threats will get a zero.** You are to use professionalism in your work. Violations of this conduct policy will be turned over to the Office of Student Integrity for misconduct.