Homework 08 – Wordle!

Authors: Helen, Ruchi, Sooraj, Chelsea
Topics: JavaFX

Problem Description

As we near the end of the semester, you're looking for a way to help you and your friends review all the topics you've learned in CS 1331. You've noticed that Wordle has been all the craze these days, so you decide to create your own version focusing on Java and OOP terms—Jordle!

However, entering words into a console is no fun. Luckily, you've recently learned JavaFX! What a perfect opportunity to create a GUI application using various UI controls, layout management, and UI event handling.

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.

It may be helpful to learn a bit more about Wordle here: https://www.nytimes.com/games/wordle
Essentially, the user has 6 tries to guess a 5-letter word. For each guess the user makes, the letters will either appear green (correct letter in the correct place), yellow (correct letter in the wrong place), or grey (wrong letter in the wrong place). The goal is to guess the word correctly.

JavaFX Installation Directions

JavaFX installation instructions can be found on Canvas here.

- 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.

Solution Description

Provided Files

- `Words.java`

  This is the class that contains the list of words your program will use. All words in this list **do not have any duplicate characters**; your code is only responsible for accounting for words without duplicate characters. Access the list in your JavaFX class using `Words.list`
  - Feel free to modify this list to test your code; you will **not** be submitting this file. However, make sure your code still works with the words provided.

Requirements

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

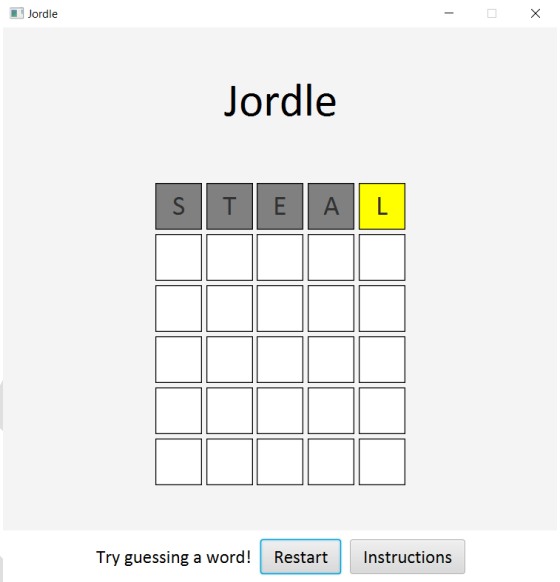
- Create a window with the title "Jordle." The window should be sized appropriately, such that contents within the window are not cut off.
- Use at least **one anonymous inner class** and **one lambda expression** in your implementation. (This is required for full points)
- Generate a **random word** when the game starts from the word list in `Words.java`. You can access the list using `Words.list`.
- Include an **instructions button** that, when clicked, opens a **new window** with a list of written instructions for how to play Jordle.
- 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
    - 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 (ex. green)
        - If the correct letter is in the wrong place, the cell should turn a different color (ex. yellow)
        - If the wrong letter is in the wrong place, the cell should turn a color different from the default background (ex. grey)
    - Pressing enter when a 5-letter word is not supplied should **alert** the user that their guess is invalid
    - The user can guess words that don't exist (any combination of characters).
- Include a **restart button** that, when clicked, resets the grid and generates a new random word.
- Include a **text label** that lets the user know about the status of their game:
    - Contains a default message (ex. "Try guessing a word!")
    - Notifies the user of the correct word if they lose after 6 guesses (ex. "Game over. The word was {word}.")
    - Congratulates the user when they win (ex. "Congratulations! You've guessed the word!")
    - Changes back to the default message when the user clicks restart

Screenshots

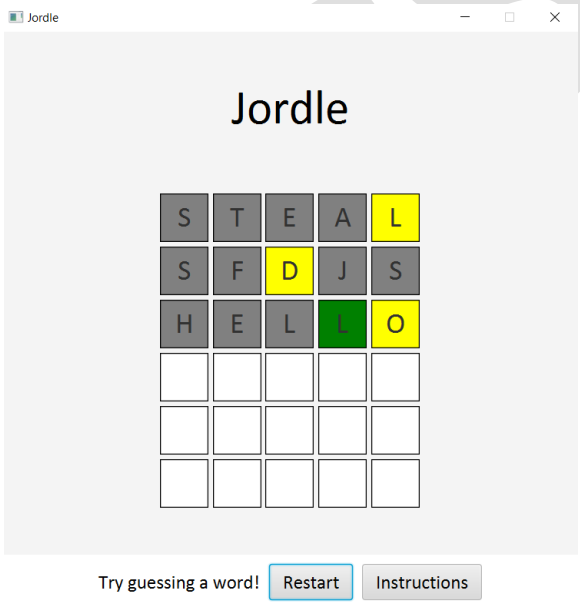
Here are images of a `Jordle` which correctly meets the requirements above. You may style your game just like this, or differently, but make sure to fulfill the requirements above.

1. Separate instructions window

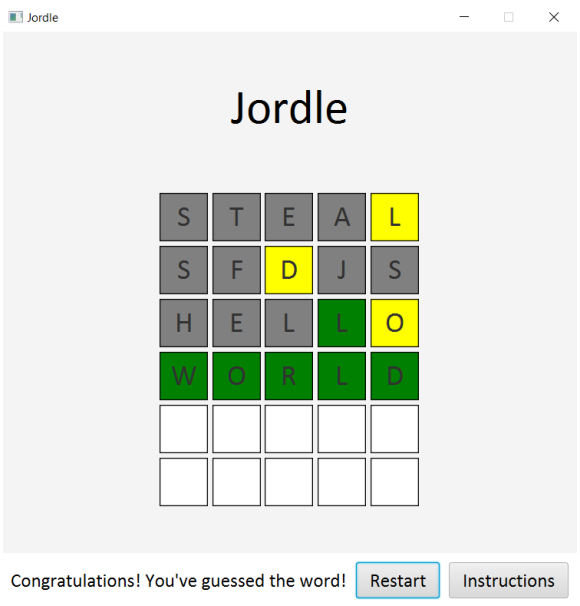
2. First guess



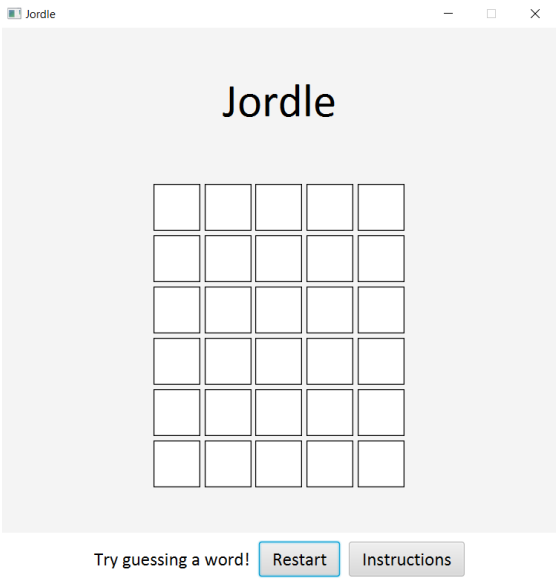
3. Third guess



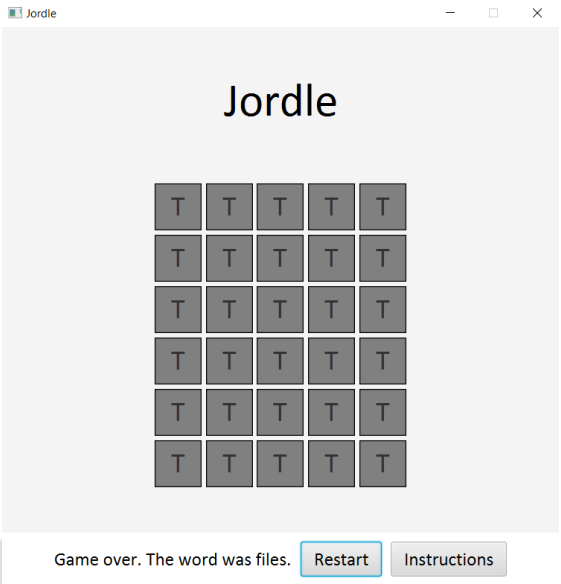
4. Guessed correct word



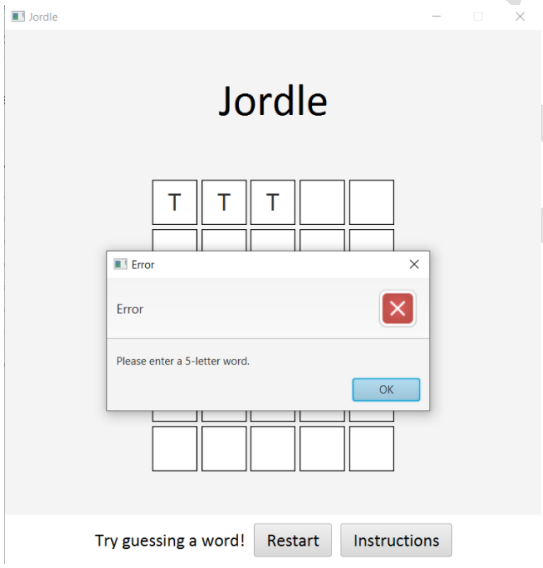
5. Clicked restart



6. Failed to guess the right word



7. Alert not 5 letters



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 Opportunities

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 **`ExtraCredit.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 ExtraCredit.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:

- 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 account for duplicate characters (use the second array in `Words.java` to test your code)
- Ability to toggle between light/dark mode, different colors for blocks
- Ability to swap between variations of wordle, like dordle, quordle, octordle, sedecordle, etc.
- A leaderboard window that shows past words and associated scores
- A file that contains words the user has already played, so the user cannot play the same word even between game runs
- 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!

Allowed Imports

There are no disallowed imports for this assignment. *Feel the full power* of the Java API. (It's over 9000)

Feature Restrictions

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

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

Checkstyle

You must run checkstyle on your submission (To learn more about Checkstyle, check out cs1331-style-guide.pdf under CheckStyle Resources in the Modules section of Canvas.) **The Checkstyle cap for this assignment is 45 points.** This means there is a maximum point deduction of 45. 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 points we would take off (limited by the checkstyle cap mentioned in the Rubric section). In future homeworks 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`
- **If you implemented extra credit, also submit `ExtraCredit.txt`**

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 test 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 Piazza 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 homework. We will only grade your latest submission. **Be sure to submit every file each time you resubmit**.

Gradescope Autograder

Due to the open-ended nature of this homework, Gradescope will only check for compilation and Checkstyle. The remainder of your assignment will be graded by a TA once the submission deadline has passed, so the output on Gradescope will not reflect your grade for the assignment.

Collaboration

Only discussion of the Homework (HW) at a conceptual high level is allowed. You can discuss course concepts and HW 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.

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.