CS5031 - Software Engineering Practice

University of
St Andrews

Practical 1

Wordle Report

Matriculation Id – 220032472

February 13th, 2023

Running the Application

The game can be run using the following commands.

The needs to be packaged using:

java clean package

This creates the executable JAR file which can be run using the following commands

java -cp target/wordle-1.0-SNAPSHOT.jar stacs.wordle.WordleApp

java -cp target/wordle-1.0-SNAPSHOT.jar stacs.wordle.WordleGui

The first command runs the game on **Command Line** whereas the second code runs the application on a **Graphical User Interface**

To run the test cases run the following command

mvn test -Dtest=WordleAppTest

Test Driven Design

The tests were written considering the working model of wordle, where the required input was passed and a random word is selected at random, acting as the word of the day. These were compared and the required output were asserted in the test result

The tests were developed using the **Laws of TDD** stated in Clean code by Robert C Martin

- **First Law** - The basic tests needed for comparing the words were developed by analysing the type of output that needs to be produced for each given guess. Failing test were created for first iteration before writing production code.

- **Second Law** - You may not write more of a unit test than is sufficient to fail, and not compiling is failing. Limited test needed to test the working of logic were developed before writing the implantation. This included only the main logic of the code (word character match) which was later expanded to other cases on need basis.

- **Third Law** You may not write more production code than is sufficient to pass the currently failing test. Additional code needed for logical working of the application, like in case of console app, colour coding was needed for displaying appropriate result. This needed a cleaner reusable class that serves this purpose, but testing this will not be related to the actual business logic needed in the case of Wordle.

Game Interface

The game can be played via console or GUI. The result produced by both the interface are similar, following the same rules and scoring strategy.

At the start of game, the rules needed for playing is displayed in both Command line and GUI, indicating how the results can be interpreted.

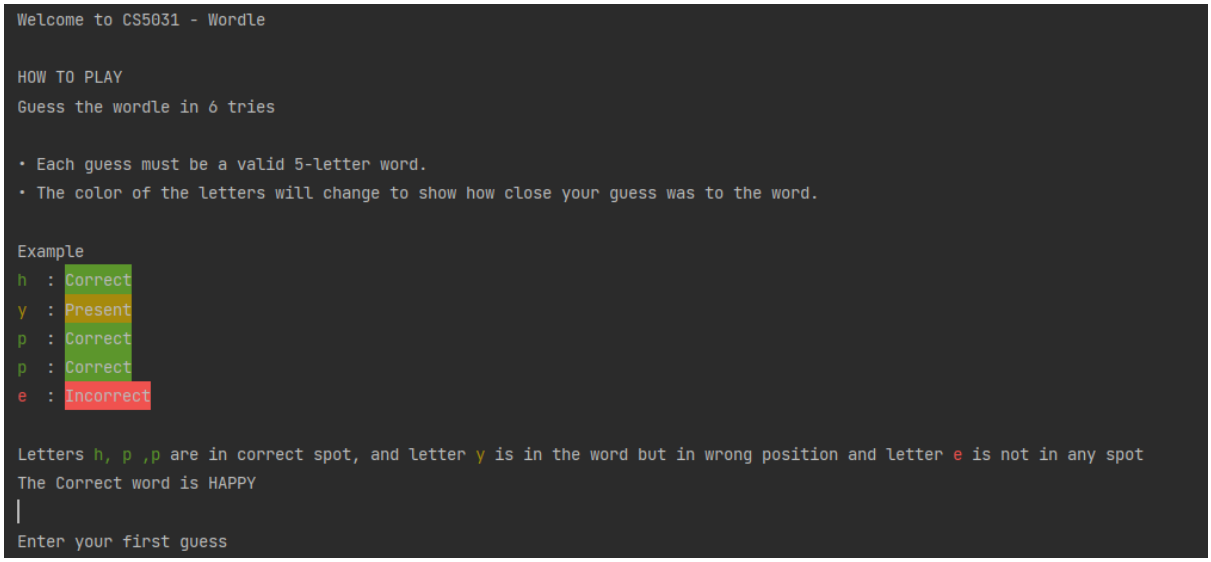


Figure showing the rules of the game in Command line

After successfully identifying the word, the statistics containing the word guess distribution along with score is displayed.

The word selected for the game is also displayed as part of Statistics.

Scoring strategy

Since every word is unique, the scoring is based on the number of correct letters guessed and the attempts taken to guess the correct word.

Score is calculated for each number of tries and correct letters guessed in each.

1. All green (10 points each) - Each try can fetch total of 50 points if all letters are correct

2. All yellow (5 points each) - Each try can fetch total of 25 points

3. All tries have a total of 20% weight overall multiplied by the descending number of tries that they take to guess.

Example - If a user gets all partial (yellow) in first try and guesses all in second then total score:

1st try - 25 points -> 10%

2nd try - 50 points -> 80%

Total score percentage (score) - 90%

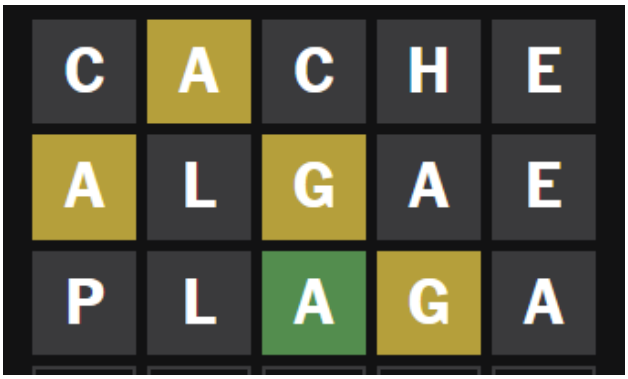
Additional Feature

System does not allow the user to enter a guess word twice. This is currently allowed in wordle, but in the implementation for this coursework it is prevented, as removing this would allow user to make a better guess, given there are only six tries.

Edge case

1. If the character is already in the other position and we entered it twice, what must happen

Example: In the following case when the letter "**A"** was used in multiple places, but was considered to be partial only in one position and not in both as the final word has only one "**A**"



This has been tackled after successive iteration of the logic refactoring.

Refactoring

1. The initial set of tests were written just to compare words and the characters in its right position. This did not capture the edge case stated in the above section. The code had to be refactored to accommodate the presence of characters in multiple positions.

   The GUI and command line code were written parallel to each other and were initially built by writing repeated logic in both the classes. This was later refactored in the following commit

   Git Commit Id- 5c542bed81eeb349f5de98fa2c03ec03e10c5715

   A common service (WordleService.java) class was built that catered to the needs of both the UIs without have code repetition.

2. Another refactoring was done with respect to the variables. Initially two separate set of variables were used to store the user input and the matched output result. This was later combined into a single dictionary value. The Refactoring can be found under the commit

   Git Commit Id - 5c8277c4d7ffb2da6407c2f3d0c6fd27b8314335