CSC120 Project: Wordle

Instructor: Mitsu Ogihara

The submission deadline: May 5th, 4:30 PM

1 The Game of Wordle

The goal of the project is to write an application for playing the game of Wordle, a popular internet game. The objective of the game is finding a secret word. Like Mastermind, the player makes guesses and for each guess, receives some feedback from the program. The feedback is letter-wise, where the program asserts each letter of a guess to be a hit, a miss, or neither. We will use `H` for a hit, `m` for a miss, and `-` for neither. Thus, each feedback is a String consisting only of these characters having the same length as the guess. For example, if the secret is `comedy` and the guess is `clouds`, the feedback is `H-m-H-`, as shown below:

```
comedy
clouds
H-m-H-
```

The hits and the misses are defined as follows.

Hit

Like in Mastermind, a letter in a guess is a hit if and only if the secret has the same letter at the same position.

Miss

A letter in a guess is a miss if the letter is not a hit but appears somewhere in the secret. Unlike Mastermind, a word may contain multiple occurrences of the same letters, e.g., "fusses", and so which occurrence of a matching letter in the secret can contribute to a miss requires some definition. We have three possible definitions.

(c1) A letter in the secret can contribute to one hit and any number of misses.

(c2) Only a letter in the secret not contributing to a hit can contribute to a miss, and such a letter can contribute to any number of misses.

(c3) Only a letter in the secret not contributing to a hit can contribute to at most one miss. A contention for a matching letter may occur among some letters appearing the guess. The contention can be resolved arbitrarily.

The following examples show how the feedback is different depending on the definition.

Secret	Guess	c1	c2	c3
`sender`	`feeble`	`-Hm--m`	`-Hm--m`	`-Hm---` or `-H---m`
`beasts`	`feeble`	`-Hmm-m`	`-H-m--`	`-H-m--`
`ending`	`feeble`	`-mm--m`	`-mm--m`	`-m----`, `--m---`, or `-----m`

Design choice

Make it clear in your comments in the program, clarify which one you have chosen.

- Implementing (c2) requires checking for each letter in the secret, if it is already contributing to a match.
- Implementing (c3) requires checking for each letter in the secret, if it is already contributing to a match, and if not, it is already contributing to a miss.
- Implementing (c1) does not require such complicated checks, and so will receive 3 fewer points.

2 Selecting a Secret

The program shall select a word from its set of words. It is recommended you use the file `worddata.txt`. The format of this data file is as follows:

- The first token is the length of the shortest word in the collection (which is 3).
- The second token is the length of the longest word in the collection (which is 10).
- After these two numbers, the words appear length-wise going from the short to the longest. For each length, the first token is the number of words having that length, and then all the words having the length appear.

A smaller data set containing only length-6 words

The length-6 part of the word data is extracted into `words6.txt`. If incorporating words having variable lengths is difficult, you may use this alternate data file and treat 6 as the only possible length.

The coding may be easier if you write a class that is responsible for reading word data, making a random selection, and possibly checking the existence of a guess in the collection.

3 Interaction

Since the length of the word is not uniform in the word file, the program must allow the user to select the word length before choosing the secret. Once receiving the length from the user, if the words data in use contains at least one word of that length, the program should make a random selection from the words having the chosen length and use it as the secret.

When the user enters a guess, the program must feedback according to the rules described in an earlier section.

In addition to providing feedback, the program should accept the following commands:

- `s`: the program reveals the secret;
- `h`: the program presents the history of guesses for the present puzzle (like the Mastermind program, the program should maintain the history of guess-feedback pairs);
- `g`: the program terminates the present puzzle, but prior to that, shows the secret and the history.

The program should decide, upon receiving an input string from the user, if it is one of the single-letter commands, it is a valid guess, or none of these. The program should decide whether or not to accept an input string as a valid guess using one of the following two conditions:

(i1) The input string is equal in length to the length of the secret, and contains only lowercase letters.

(i2) The input string is equal in length to the length of the secret, and appears in the word data.

4 Project Submission and Evaluation

You must submit one plain text file.

- When you submit your work, copy and paste the input and output appearing in the terminal area of IntelliJ in a text file.
- For each java file of your code, provide detailed comments for each method and for each variable. For the hit-and-miss calculation, describe the algorithm you used.
- In the main program file (that is, the one where your main method resides), using multiple-line comments, describe the strategy you employed in developing the code.
- The attached program `AssembleFiles` allows its user to assemble files (that are in the plain text format) into a single file. Create an assembly consisting of your java files and the execution sample file.

5 Coding Tips

- Since the project is to substitute the final, you must not collaborate/consult with others.
- The programming task can be accomplished using only those classes and methods covered in the course. Using classes not covered in the course may result in reductions of points. The following two methods are exceptions. You may use them without penalty.
  - The method `toCharArray()`: when applied to a `String` object, the method return an array of `char` representing the String. For example, if `w` is a `String` equal to `"csc120"`. The call `w.toCharArray()` returns a 6-element `char` array whose elements are `'c'`, `'s'`, `'c'`, `'1'`, `'2'`, `'0'` in this order.

    The secret and the guesses can be converted to arrays of `char`. Writing the code for comparison may be easier with the use of array representations than with the use of `charAt()`.
  - One of the constructor for `String` takes a `char` array as the parameter. For example, if `x` is a `char` array whose elements are `'a'`, `'x'`, `'i'`, `'s'` in this order, then `new String(x)` produces `"axis"`.

    You can build the feedback as an array of `char` and then turn the array into a `String`.
- For reading data from the data file, you may need to use a multi-dimensional array in the case where your source file is `worddata.txt`.
- For implementing (i2), you can use the search algorithm you used in the write-in election program.

6 Example

The example below uses (c2) and (i2) in the above.

```
##########################################################
# Let's play Wordle.                                     #
# Your goal is to guess a secret word.                   #
# The word may have duplicated letters.                  #
# For each guess, you receive a feedback.                #
# 'H' for hits, 'm' for misss, and '-' for others.       #
# Your commands are as follows                           #
#   s for showing the secret,                            #
#   h for show the history, and                          #
#   g for giving up and terminating the present puzzle. #
```

```
############################################################
Choose the length for the secret: 4
> cane
-H--
> sail
mH--
> pays
-HHH
> rays
-HHH
> days
-HHH
> ways
-HHH
> bays
-HHH
> gays
-HHH
> jays
HHHH
You've got it.
History
 1:cane:-H--
 2:sail:mH--
 3:pays:-HHH
 4:rays:-HHH
 5:days:-HHH
 6:ways:-HHH
 7:bays:-HHH
 8:gays:-HHH
 9:jays:HHHH
Another game? y
############################################################
# Let's play Wordle.                                       #
# Your goal is to guess a secret word.                     #
# The word may have duplicated letters.                    #
# For each guess, you receive a feedback.                  #
# 'H' for hits, 'm' for misss, and '-' for others.         #
# Your commands are as follows                             #
#   s for showing the secret,                              #
#   h for show the history, and                            #
#   g for giving up and terminating the present puzzle.    #
```

```
############################################################
Choose the length for the secret: 5
> doing
-H-H-
> loans
-H-HH
> towns
-H-HH
> h
History
 1:doing:-H-H-
 2:loans:-H-HH
 3:towns:-H-HH
> g
History
 1:doing:-H-H-
 2:loans:-H-HH
 3:towns:-H-HH
Secret = moons
Another game? y
############################################################
# Let's play Wordle.                                       #
# Your goal is to guess a secret word.                     #
# The word may have duplicated letters.                    #
# For each guess, you receive a feedback.                  #
# 'H' for hits, 'm' for misss, and '-' for others.         #
# Your commands are as follows                             #
#   s for showing the secret,                              #
#   h for show the history, and                            #
#   g for giving up and terminating the present puzzle.    #
############################################################
Choose the length for the secret: 6
> double
-----m
> makers
---H-m
> queens
--HH-m
> sheeps
Invalid input.
> sleeps
m-HH-m
```

```
> gheese
Invalid input.
> creeds
H-HH-m
> cheese
HHHHH-
> cheesy
HHHHHH
You've got it.
History
 1:double:-----m
 2:makers:---H-m
 3:queens:--HH-m
 4:sleeps:m-HH-m
 5:creeds:H-HH-m
 6:cheese:HHHHH-
 7:cheesy:HHHHHH
Another game? n
Bye!
```

7 Rubric for Evaluation

The following rubric will be used for evaluating the submitted work.

1. (10 points) The code is syntactically correct and uses nothing beyond what has been taught in the course.

2. (15 points) The code has detailed comments about variables, logic, and methods and about your development strategy.

3. (10 points) The execution file shows at least three different puzzles, demonstrating all possible interactions.

4. (20 points) The program computes hits and misses correctly (no more than 17 points if (c1) is implemented instead of (c2)).

5. (15 points) The program uses one of the two data files as its source.

6. (30 points) Operational accuracies.

   (a) (5 points) The program allows an indefinite number of games and allows termination of the program.

   (b) (5 points) The program allows the user to specify the word length of the secret, even if 6 is the only available length.

(c) (5 points) The program reveals its secret upon receiving `s`.

(d) (5 points) The program terminates the puzzle upon receiving `g` as described in the above.

(e) (5 points) The program prints the history upon receiving `h`.

(f) (5 points) The program checks the validity of each guess.