Wordle

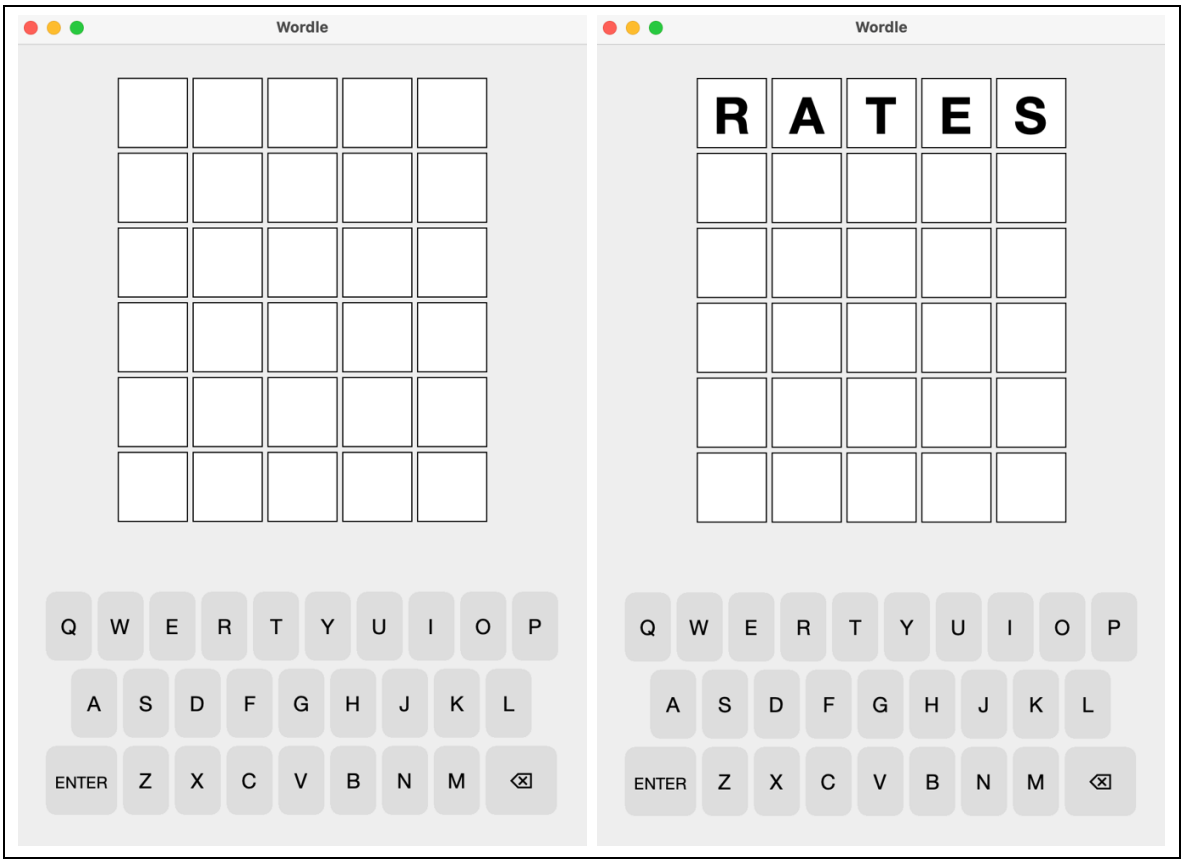
Acknowledgements to Eric Roberts of Willamette University

This project is designed to help you practice working with strings in the context of an engaging application: the Wordle game initially developed by Josh Wardle, now available on the *New York Times* website. Given Wordle's enormous popularity, we thought having you implement the game would be fun.

To get started on this project, download, extract, and save the Wordle project to your computer. The Graphical User Interface (GUI) has been completed for you, so you can run the `main` method in the `Wordle` class to see what it looks like. It creates a window, draws the letter boxes, and creates the keyboard at the bottom of the window. You can even type in letters by typing on the keyboard or clicking the keys on the screen, just as you can when playing the online version. Figure 2, for example, shows both the initial screen and the screen you get after typing in the five letters in the helpful starting word RATES, which includes five of the most common letters.

Unfortunately, that's all the program does at this point. It doesn't let you play the Wordle game (yet). Your job is to make the game playable. But first, it is worth reviewing the rules for Wordle in case you've somehow managed to miss this craze.

Figure 2. Running the starter program



Playing Wordle

The object of the Wordle puzzle is to figure out the secret word for the day using no more than six guesses. When you type in a word and then hit the RETURN or ENTER key, the website gives you information about how close your guess is by coloring the background of the letters. For every letter in your guess that is in its correct position, Wordle colors the background a light shade of green. For every letter that appears in the word but is not in the correct position, Wordle colors the background a brownish-yellow. All letters in the guess that don't appear in the word are colored a medium gray.

For example, suppose that the secret word for the day was RELIC, and your first guess was RATES, as in the Figure 2 example. The R is in the correct position, and the word contains an E, but not in the position you guessed. The secret word

does not have any of the letters A, T, and S. Wordle reports that information by changing the background colors of the squares like this:

Even though you know the position of the R, it doesn't make sense to guess more words beginning with R at this point because doing so gives you no new information. Suppose you tried guessing the word LINGO, which contains five new letters, two of which appear in the word but none in the correct positions. Wordle responds by coloring the letter squares in your second guess as follows:

Putting these two clues together means that you know that the word begins with an R, contains the letters E, L, and I in some order other than the one you guessed, and that the letters A, T, S, N, G, and O do not appear anywhere in the word. These answers give you an enormous amount of information. If you think carefully about it, you might find the word RELIC, which is, in fact, the only English word that meets these conditions:

Done in three!

It is worth noting a few other rules and special cases. The secret word and each of your guesses must be an actual five-letter English word. Each guess will be checked against a list of more than 12,000 valid five-letter words. If you guess a word that is not in the word list, Wordle displays a message to that effect, at which point you can delete the letters you've entered and try again. Another rule is that you only get six guesses. If all the letters don't match by then, Wordle gives up on you and reveals the secret word.

The most interesting special cases arise when the secret word and the guesses contain duplicates of the same letter. Suppose, for example, that the secret word is GLASS, and you, for some reason, guess SASSY. Wordle responds with the following colors:

The green S shows that there is an S in the fourth position, and the yellow S shows that a second S appears somewhere else in the secret word. The S in the middle of SASSY, however, remains gray because the secret word does not contain three instances of the letter S. Properly handling special cases like this is the most challenging aspect of this project.

The `WordleGUI` class

When you run the `main` method in `Wordle`, the `WordleGUI` class is instantiated as the variable `wordle`. Even though you don't have to make any changes to `WordleGUI` or understand the details of its operation, you need to know what capabilities it offers so you can use those facilities in your code. The methods of the `WordleGUI` class are outlined below in Figure 3. The right column of the table gives only a brief description of what these methods do. More complete descriptions appear later in this handout in the description of the step that requires them.

Figure 3. Useful methods from the `WordleGUI` class

`wordle.getCurrentRow()`	Returns the current row.
`wordle.setCurrentRow(`*row*`)`	Sets the row in which typed characters appear.
`wordle.getSquareLetter(`*row*`, `*col*`)`	Returns the letter in the specified row and column.
`wordle.setSquareColor(`*row*`, `*col*`, `*color*`)`	Sets the color of the specified square.
`wordle.getSquareColor(`*row*`, `*col*`)`	Returns the color of the specified square.
`wordle.setKeyColor(`*letter*`, `*color*`)`	Sets the color of the specified key letter.
`wordle.getKeyColor(`*letter*`)`	Returns the color of the specified key letter.
`wordle.showMessage(`*message*`)`	Shows a message below the squares.

Here are some predefined constant values that will be helpful in completing the required methods.

Figure 4. Useful constants from the `WordleGUI` class

`int wordle.N_ROWS`	Total number of display rows (normally 6)
`int wordle.N_COLS`	Total number of display columns (normally 5)
`Color wordle.MISSING_COLOR`	Light gray color
`Color wordle.CORRECT_COLOR`	Green color
`Color wordle.PRESENT_COLOR`	Brownish-yellow color

(Note: In Java, a `Color` object represents specific numeric values of red, blue, and green colors. The three `Color` constants above are already defined for you.)

Now, it's time to color the individual letters. While it is easy to pick a color (green, yellow, or gray), the logic for determining the right color for each letter is more complicated than it seems. We will tackle it in this fashion, coloring both the guessed word letters and the corresponding keyboard letters at the bottom of the display simultaneously. Here is the general strategy for how we will proceed.

1. First, we'll color all the guessed word letters gray, as if they were missing. Then, we won't have to return later to deal with missing letters. This color is the predefined `Color` constant `wordle.MISSING_COLOR`.
2. Second, we'll check for correctly placed letters, one at a time, and color those green. This color is the predefined `Color` constant `wordle.CORRECT_COLOR`.
3. Third, we'll deal with misplaced letters (present in the secret word but in the wrong spot). This part is the most challenging task, as we have to know how many repeated letters to color yellow (not necessarily all of them). (Go back and look at the SASSY example above if this does not make sense to you yet). This color is the predefined `Color` constant `wordle.PRESENT_COLOR`.

All three steps are executed so quickly that you will not see them change from gray to green or yellow.

Step 0: Set all the guessed letters to gray

Complete the `setAllLettersToMissing` method. You'll need to keep in mind the `WordleGUI` methods and constants listed above. First, store the game's current row, because we'll need to know which row of letters to color in the display. Second, loop through each of the display columns. Then, get an individual letter and color the entry letter and the keyboard letter. Complete the method as follows.

```
/** Set all letters to missing (gray) color */
private void setAllLettersToMissing()
{
    int row = wordle.getCurrentRow();                          // get the current row
    for(int col = 0; col < wordle.N_COLS; col++)               // loop through each column
    {
        String letter = wordle.getSquareLetter(row, col);      // get a letter
        wordle.setSquareColor(row, col, wordle.MISSING_COLOR);  // color entry letter
        wordle.setKeyColor(letter, wordle.MISSING_COLOR);       // color keyboard letter
    }
}
```

Notice that each guessed letter (and the corresponding keyboard letter below) is now colored gray.

Step 1: Find the correct letters (and color them green)

Complete the `findCorrectLetters` method. You want to check each letter in the method parameter `guess` against the instance variable `secretWord`. Get the current row, and then loop through each of the columns (as you did in Step 0), and use the loop variable to get a corresponding single letter in both `guess` and `secretWord` using the `String` method `substring`. For example, the following statement gets a single letter from position `col` in `guess` (do something similar to also get a single corresponding letter in `secretWord`.)

```
String g = guess.substring(col, col + 1);
```

Then, compare the corresponding single letters in `guess` and `secretWord` to see if they are identical (using the appropriate way to compare two `String` objects for equivalence). If so, color both the entry letter and keyboard letter green (`wordle.CORRECT_COLOR`).

Finally, this method determines if the user guessed the secret word (or not) by returning `true` or `false`. Do this by returning the `boolean` result of the comparison of `guess` and `secretWord` (again, using the appropriate way to compare two `String` objects for equivalence).

Check your work by running `WordleTest` to see that this step has been completed successfully. The tester will display the results of testing your method.

Notice that each correctly guessed letter (and the corresponding keyboard letter below) is now colored green.

Step 2: Count the present letters

"Present" letters appear in the secret word but are not in the correct position. The challenge here is for duplicate (or even triplicate) letters. Consider another example (similar to the earlier example). Say that the secret word is FERRY, and `guess` is as follows.

Notice that ERROR contains three Rs, while FERRY (the secret word) only has two Rs. The guess above is colored correctly in this case. First, the middle R is in the correct position, so it is colored green. FERRY has one other R, so the first R is colored yellow, but the last R is left as is (gray). In other words, before we color any letter yellow, we need to count the

number of times that letter appears in the hidden word and then color that many letters yellow (and no more). Oh, and one more thing: don't include the green letters in your count.

So, the first thing to do is complete the method `presentLetterCount`. This method has one parameter `letter`. Here, we want to count the number of times `letter` appears in `secretWord`, but also check the display to ignore any correctly placed (green) letters. Start by setting up a counter and then return the counter at the end of the method. Right after the counter, set up an integer `col` variable and set it equal to `secretWord.indexOf(letter)`. This method call will return the column (integer position) where `letter` appears in `secretWord`, or -1 if it does not appear. Therefore, if `col >= 0` means that `letter` indeed appears in `secretWord`, so we'll loop around again, looking for another occurrence of `letter` in `secretWord`.

Note: All color constants (such as `wordle.CORRECT_COLOR`) represent `Color` objects. Like `String` objects, `Color` objects have an `equals` method to determine if they are equivalent to another `Color` object or not.

Here is the pseudocode of what needs to be done in this method.

```
get the current row
initialize a counter to zero
initialize a column variable to the position of letter in secret word *
while the column is not negative
        if the display square color is not green    // ignore correct (green) letters
                increment the counter
        set column to the position of letter in secret word, starting at position column + 1 **
return the counter
```

* using `secretWord.indexOf(letter)` ** using `secretWord.indexOf(letter, col + 1)`

Check your work by running `WordleTest` to see that this step has been completed successfully.

Step 3: Color the present letters

Complete the `findPresentLetters` method according to the following pseudocode. Be sure to pay close attention to the code indentations, as they indicate precisely what is included in each control structure (e.g., for-loop, if statement, etc.).

(Recall that all color constants represent `Color` objects, which, like `String` objects, must use `equals` to determine equivalence. To determine if two objects are *not* equal, use the negation operator '!' before the beginning of the `equals` statement.)

```
get the current row
loop through each letter in guess
        get an individual letter in guess
        initialize count variable to the presentLetterCount(letter)
        initialize a col variable to zero
        while count is greater than zero and col is less than the length of guess
                if letter equals guess character at position col
                        if square color (row, col) equals missing color
                                set square color (row, col, present color)
                                if letter key color equals missing color
                                        set key color (letter, present color)
                        if square color (row, col) does not equal correct color
                                decrement count
                increment col
```

Check your work by running `WordleTest` to see that this step has been completed successfully.

Congratulations. You have completed the Wordle project and can play the game properly. Feel free to play it as much as you wish. Unlike the New York Times version, you can play this Wordle repeatedly.

Please submit a screenshot of the tester success message when you have completed this assignment.