Manual for Wheel of Fortune

Group B – Atanas Komsiyski ID: 51985145, Aleksandra Nenkova ID:51984461, Reece Tait ID:51982218, Stanislav ID:51985020

System Specification

Software Needed

- Windows/Mac/Linux Operating System
- Latest version of Ruby
    - Gems:
        - RSPEC – Needed to run the tests
- Access to the terminal/command line

Running the Software

- To run the test, use the following command
    - *'rpsec wad_wof_spec_01.rb'*
    - There should be no red error messages and a row of green dots at the top (see screenshot below)
    - Ignore any warnings presented
    - Ensure the wordfile text file is **not** empty,
- To run the game, use the following command
    - *'ruby wad_wof_run_01.rb'*

The Building Blocks

Start

The start function does two things:

- It displays a welcome message by calling the put method from the *output* object
- It calls the *created_by* and *student_id* functions and outputs their results using the put method from the *output* object

Created_By

Returns a string containing the name of each of the developers

Student_Id

Returns a string containing the student ID of each of the developers

MenuOptions

Returns a string containing the following menu options:

- (1) Play [Begins the game] [Invokes *displaybegingame* function]
- (2) New [Creates a new game session] [Invokes *displaynewgamecreated* function]
- (3) Analysis [Returns a breakdown of the last game played] [Invokes *displaygameanalysis* function]
- (4) Exit [Exits from the game]

MenuPrompt

Returns a prompt telling the user to select an option from the menu

DisplayBeginGame
Displays a begin game message using the puts method from the output *object* when menu option 1 is selected

DisplayNewGameCreated
Displays a new game created message using the puts method from the *output* object when menu option 2 is selected

DisplayGameAnalysis
Displays a game analysis message using the puts method from the *output* object when menu option 3 is selected

Finish
Displays a game finished message using the puts method from the *output* object

DisplayInvalidInputError
Displays an error message if an invalid input is entered using the puts method from the *output* object

ResetGame
Resets the values of the game variables to their default values and prepares the game for a fresh start

ReadWordFile(filename)
Opens the specified file and loops through each line within the file. It then increments the amount of lines and stores the line in the *wordtable* object, removing the newline character at the end, before returning the value of amount

GenSecretword
Gets a random value from the *wordtable* object and sets the value of the *secretword* object to a capitalised version of it.

CheckWordUpcase?
Checks if the value of the *secretword* variable is in uppercase and returns true if it is, or false if it isn't

SetSecretword(word)
Sets the value of *secretword* to the incoming value from the word variable

GetSecretword
Returns the value of the *secretword* object

CreateTemplate
Creates a local variable called *template* which will store the current state of the template. Splits the value of the *secretword* object into an array of characters before looping over each character and adding an underscore ("_") into the *template* variable per character. It then sets the *template* object to the value of the local *template* variable before returning it

GetSecretTemplate

Returns the value of the *secretword* object and its corresponding *template* object in an array

IncrementTurns

Increases the amount of turns taken by 1

GetTurnsLeft

Sets the value of the *turnsleft* object to the value of the *turnsleft* variable. It then takes 1 away from amount of possible goes, using the constant variable *GOES*, and sets the *turnsleft* object to the value.

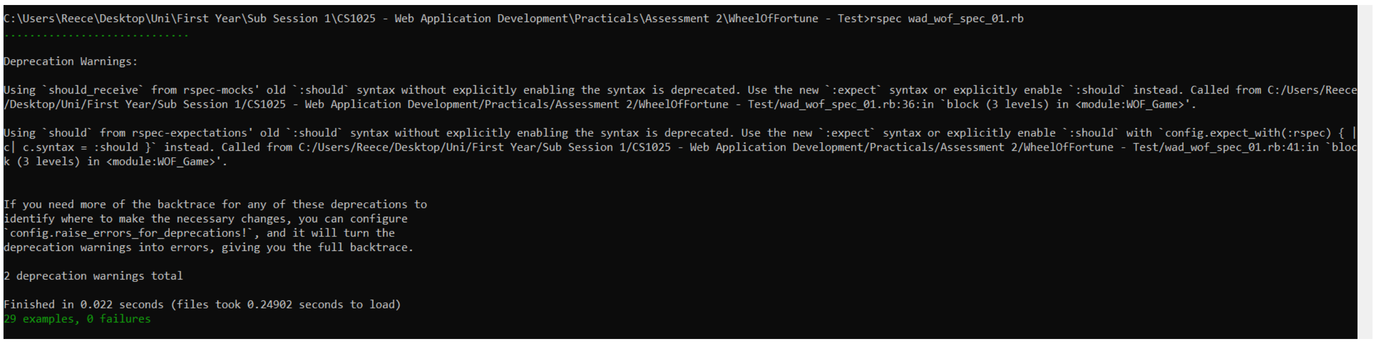
DisplayWinner(won)

Returns a win/lose message depending on the value of the *won* variable. If true, it'll return a win message. If false, it'll return a lose message.

DisplayCredits(i, names, ids)

Splits the *names* and *ids* arrays from the input into local variables before returning with the value of each array at the index *i* from the input

Screenshot of Tests Working



```
C:\Users\Reece\Desktop\Uni\First Year\Sub Session 1\CS1025 - Web Application Development\Practicals\Assessment 2\WheelOfFortune - Test>rspec wad_wof_spec_01.rb
.............................

Deprecation Warnings:

Using `should_receive` from rspec-mocks' old `:should` syntax without explicitly enabling the syntax is deprecated. Use the new `:expect` syntax or explicitly enable `:should` instead. Called from C:/Users/Reece
/Desktop/Uni/First Year/Sub Session 1/CS1025 - Web Application Development/Practicals/Assessment 2/WheelOfFortune - Test/wad_wof_spec_01.rb:36:in `block (3 levels) in <module:WOF_Game>'.

Using `should` from rspec-expectations' old `:should` syntax without explicitly enabling the syntax is deprecated. Use the new `:expect` syntax or explicitly enable `:should` with `config.expect_with(:rspec) { |
c| c.syntax = :should }` instead. Called from C:/Users/Reece/Desktop/Uni/First Year/Sub Session 1/CS1025 - Web Application Development/Practicals/Assessment 2/WheelOfFortune - Test/wad_wof_spec_01.rb:41:in `bloc
k (3 levels) in <module:WOF_Game>'.


If you need more of the backtrace for any of these deprecations to
identify where to make the necessary changes, you can configure
`config.raise_errors_for_deprecations!`, and it will turn the
deprecation warnings into errors, giving you the full backtrace.

2 deprecation warnings total

Finished in 0.022 seconds (files took 0.24902 seconds to load)
29 examples, 0 failures
```