CAD TESTS SYSTEM DOCUMENTATION

CONTENTS

IMPORTANT NOTES, PLEASE READ

During development, our team discovered that the code needed to read the text file was not cross compatible between the vid ruby version 2.3 and one of our team members using MacOS ruby version 2.6. Hence, we have included both in our files and in this documentation, specifically for the tests regard the reading form files. For simplicity we refer to these difference with tags of vid and MacOS in the documentation.

HOW TO USE THE INSTRUCTION MANUAL/DOUCMENTATION

This documentation is has been written to guide the user on how to install and run the testsSystem specification and installation instructions. (i.e. software required to maintain and run the tests, etc.) The documentation is also written to provide a description of each building block implemented (i.e. what it does to pass the test). And to include screenshots illustrating the passing of tests.

The document includes a contents page to aid navigation

SYSTEM SPECIFICATION

To run the Ruby and TDD software, along with the tests, your computer must be running either Windows, MacOS, or Linux.

INSTALLING AND RUNNING THE TESTS

INSTALLING

Before you are able to run the files, you will need to have Ruby and TDD software installed. To install these, please follow the respective links and instructions.

Ruby

https://www.ruby-lang.org/en/documentation/installation/

TDD

Install the TDD gem

Command: `gem install rspec`

RUNNING

To run the test, please extract the ***yourfullname*-wadca2.zip** folder.

In the terminal, navigate to the location where the files have been extracted to.

To run the test, run the command: *rsepc wad_wof_spec_01.rb*

CODE BLOCKS

The following will be demonstrating each section of the code and explain how it passes the test.

TEST 1: START METHOD

This test is looking at the code and checking for a method called "*start*".

```
def start()
 output.puts("Welcome to WOF!")
 output.puts("Created by: "+ created_by() + " (" + student_id() + ")")
end
```

As seen in the code, a method called *start* has been implemented in ruby with the correct syntaxes. Hence the code passes the first test.

TEST 2: WELCOME MESSAGE

This test is looking at the output and checking if a welcome message has been displayed when the *start* method is called.

```
def start()
output.puts("Welcome to WOF!")
output.puts("Created by: "+ created_by() + " (" + student_id() + ")")
end
```

As seen in the code, an output with the value "*Welcome to WOF!*" has been implemented within the *start* method. Hence it will pass the test as when the *start* method is called, the method will then execute the output instruction, thus passing the test.

TEST 3: CREATED_BY METHOD

This test is checking for a method called "*created_by*" and is expecting a return with the values "*Noura El Khadri,Ruilin Wang,Adam Taylor,Alexander Bremner*", where this text is the student names.

```
def created_by()
myname = "Noura El Khadri,Ruilin Wang,Adam Taylor,Alexander Bremner"
end
```

(1) As seen in the code, a method called "*created_by*" has been created with the correct syntaxes.

(2) In addition, a text output with the value "*Noura El Khadri,Ruilin Wang,Adam Taylor,Alexander Bremner*" has been outputted.

Hence the code passes the test as it has executed a method called "*created_by*", and this method has outputed the correct values of "*Noura El Khadri,Ruilin Wang,Adam Taylor,Alexander Bremner*".

TEST 4: STUDENT_ID METHOD

This test is checking for a method called "*Student_ID*", where this method returns values which are the student ID numbers "*51986044,51986323,51988294,51987069*"

```
def student_id()
studentid = "51986044,51986323,51988294,51987069"
end
```

(1) As seen in the code, a method called "*Student_id*" has been created with the correct syntaxes.

(2) In addition, a text output with the value "*51986044,51986323,51988294,51987069*" has been outputted.

Hence the code passes the test as it has executed a method called "*created_by*", and this method has outputed the correct values of "*Noura El Khadri,Ruilin Wang,Adam Taylor,Alexander Bremner*".

TEST 5: CREATED BY

This test is checking for an output that displays who created the program, when the *start* method is executed.

```
def start()
output.puts("Welcome to WOF!")
output.puts("Created by: "+ created_by() + " (" + student_id() + ")")
end
```

As seen in the code, an output with the value "Created by:" along with the outputs from methods *created_by* and *student_id* has been implemented within the *start* method. Hence it will pass the test as when the *start* method is called, the method will then execute the output instruction. This execution results in the two methods being called and subsequently their values passed back into the instruction. Thus this will past the test as the method values will be outputted.

TEST 6: MENU OPTIONS

This test is checking for a return value containing all the menu options, when the *menuoptions* method is called.

```
def menuoptions()
moptions = "Menu: (1) Play | (2) New | (3) Analysis | (9) Exit\n"
end
```

As seen in the code, a return value of "*Menu: (1) Play | (2) New | (3) Analysis | (9) Exit\n*" has been implemented within the *menuoptions* method. Hence it will pass the test as when the *menuoptions* method is called, the method will then execute the output instruction, thus passing the test.

TEST 7: MENU INSTRUCTIONS

This test is checking for a return that is the instruction "Select option from menu.", when the *menuprompt* method is executed.

```
def menuprompt()
mprompt = "Select option from menu."
end
```

As seen in the code, an output with the value "*Menu: (1) Play | (2) New | (3) Analysis | (9) Exit\n*" has been implemented within the *menuoptions* method. Hence it will pass the test as when the *menuprompt* method is called, the method will then execute the output instruction, thus passing the test.

TEST 8: BEGIN GAME

This test is looking at the output and checking if the text output "Begin game." has been returned, when the method *displaybegingame* has been called.

```
def displaybegingame()
output.puts("Begin game.")
end
```

As seen in the code, an output with the value "*Begin game.*" has been implemented within the *displaybegingame* method. Hence it will pass the test as when the *displaybegingame* method is called, the method will then execute the output instruction, thus passing the test.

TEST 9: NEW GAME

This test is looking at the output and checking if the text output "*New game created*." has been returned, when the method *displaynewgamecreated* has been called.

```
def displaynewgamecreated()
output.puts("New game created.")
end
```

As seen in the code, an output with the value "*New game created*." has been implemented within the *displaynewgamecreated* method. Hence it will pass the test as when the *displaynewgamecreated* method is called, the method will then execute the output instruction, thus passing the test.

TEST 10: GAME ANALISIS

This test is looking at the output and checking if the text output "*Analysis of game.*" has been returned, when the method *displaygameanalysis* has been called.

```
def displaygameanalysis()
output.puts("Analysis of game.")
end
```

As seen in the code, an output with the value "*Analysis of game.*" has been implemented within the *displaygameanalysis* method. Hence it will pass the test as when the *displaygameanalysis* method is called, the method will then execute the output instruction, thus passing the test.

TEST 11: FINISH MESSAGE

This test is looking at the output and checking if the text output "*Game finished.*" has been returned, when the method *finish* has been called.

```
def finish()
output.puts("Game finished.")
end
```

As seen in the code, an output with the value "*Game finished.*" has been implemented within the *finish* method. Hence it will pass the test as when the *finish* method is called, the method will then execute the output instruction, thus passing the test.

TEST 12: ERROR INVALID INPUT

This test is looking at the output and checking if the text output "*Invalid input.*" has been returned, when the method *displayinvalidinputerror has* been called.

```
def displayinvalidinputerror()
output.puts("Invalid input.")
end
```

As seen in the code, an output with the value "*Invalid input.*" has been implemented within the *displayinvalidinputerror* method. Hence it will pass the test as when the *displayinvalidinputerror* method is called, the method will then execute the output instruction, thus passing the test.

TEST 13: RESTART GAME

This test is looking at the code and checking for a method called "*resetgame*".

```
def resetgame()
@wordtable = []
@secretword = ""
@turn = 0
@resulta = []
@resultb = []
@winner = 0
@guess = ""
@template = "[]"
end
```

As seen in the code, a method called *resetgame* has been implemented in ruby with the correct syntaxes. Hence the code passes the test.

TEST 14: RESET GAME OBJECT VARIBLES

This test is checking for a return value containing all the correct reseted object values, when the *restgame* method is called.

```
def resetgame()
@wordtable = []
@secretword = ""
@turn = 0
@resulta = []
@resultb = []
@winner = 0
@guess = ""
@template = "[]"
end
```

As seen in the code, a the values of "*@wordtable = [], @secretword = "", @turn = 0, @resulta = [], @resultb = [], @winner = 0, @guess = "", @template = "[]""* objects has been set to the correct values within the *restgame* method. Hence it will pass the test as when the *restgame* method is called, the method will then execute the output instruction, resetting all the values, thus passing the test.

TEST 15: READ WORD FILE (VDI)

This test is checking for a method called "*readwordfile*" and passing the value of the file to read, where the tested file name is "*wordfile.txt*".

```ruby
def readwordfile(fileToRead)
@wordtable = File.read(fileToRead).split"\n"
words = @wordtable.length

#The following is the reqired code to read the file on MacOS
# wordtable = []

# count = 0
# file = File.open(fileToRead)

# file.each_line do |x|
    # count += 1
    # wordtable.append x.strip
# end

# file.close

# @wordtable = wordtable

# return count

end
```

(1) As seen in the code, a method called "*readwordfile*" has been created with the correct syntaxes.

(2) In addition, this line takes the file name passed into the method and runs a ruby pre defined method on the file name. This extracts the text from the file and stores it in the object "wordtable"

Hence the code passes the test as it has executed a method called "*readwordfile*", and this method reads and stores the value of the file from the file name passed into the method.

TEST 15: READ WORD FILE (MACOS)

This test is checking for a method called "*readwordfile*" and passing the value of the file to read, where the tested file name is "*wordfile.txt*".

```ruby
def readwordfile(fileToRead)
@wordtable = File.read(fileToRead).split"\n"
words = @wordtable.length

#The following is the reqired code to read the file on MacOS
# wordtable = []

# count = 0
# file = File.open(fileToRead)

# file.each_line do |x|
    # count += 1
    # wordtable.append x.strip
# end

# file.close

# @wordtable = wordtable

# return count

end
```

As seen in the code, a method called "*readwordfile*" has been created with the correct syntaxes. In addition, the method takes the name passed into it and uses it. As highlighted the code uses a predefined ruby method to open the file using the name passed into it. This is then stored in the variable flie. Then another predefined ruby method is used to close the file. Hence the code passes the test as the correct method has been created and it opens the file of the name passed into the method.

TEST 16: THE NUMBER OF WORDS OR PHRASES (VDI)

This test tests what a prevevious test has tested by checking for the method "*readwordfile",* and it test that this method passes the value of the number of words or phrases in the wordfile.txt which is 4.

```ruby
def readwordfile(fileToRead)
@wordtable = File.read(fileToRead).split"\n"
words = @wordtable.length

#The following is the reqired code to read the file on MacOS
# wordtable = []

# count = 0
# file = File.open(fileToRead)

# file.each_line do |x|
    # count += 1
    # wordtable.append x.strip
# end

# file.close

# @wordtable = wordtable

# return count

end
```

As seen in the code, the ruby code takes the values within the file and runs it though a ruby predefined method called “length”, this returns the value of the number of words or prases in the text, and the code then stores it in a variable. Hence it passes the test as the method returns the number of words or phrases in the text document.

TEST 16: THE NUMBER OF WORDS OR PHRASES (MACOS)

This test tests what a prevevious test has tested by checking for the method “*readwordfile”,* and it test that this method passes the value of the number of words or phrases in the wordfile.txt which is 4.

```
def readwordfile(fileToRead)
@wordtable = File.read(fileToRead).split"\n"
words = @wordtable.length

#The following is the reqired code to read the file on MacOS
# wordtable = []

# count = 0
# file = File.open(fileToRead)

# file.each_line do |x|
    # count += 1
    # wordtable.append x.strip
# end

# file.close

# @wordtable = wordtable

# return count

end
```

As seen in the code. The variable count is used to count the number of words or phrases. On each line of the text document is a new word or phrase. So the highlighted method adds 1 to the variable count for each line there is in the text document. This value is then passed back out of the method. Hence the code passes the test as it returns the number of words or phrases

TEST 17: STORING WORDS AND PHRASES IN AN OJBECT (VDI)

This test is tesing that the words and phrases are stored in an object when the *readwordfile* method is executed, where the stored phrases are *“"duck","eider","mallard","punk duck"”.*

```ruby
def readwordfile(fileToRead)
@wordtable = File.read(fileToRead).split"\n"
words = @wordtable.length

#The following is the reqired code to read the file on MacOS
# wordtable = []

# count = 0
# file = File.open(fileToRead)

# file.each_line do |x|
    # count += 1
    # wordtable.append x.strip
# end

# file.close

# @wordtable = wordtable

# return count

end
```

As see in the code, when the text file is read, the contents of the file is stored in an object called "*wordtable*". Hence this passes the test as when the *readwordfile* method is executed, the contents of the text file are stored in the correct object.

TEST 17: STORING WORDS AND PHRASES IN AN OJBECT (MACOS)

This test is tesing that the words and phrases are stored in an object when the *readwordfile* method is executed, where the stored phrases are *""duck","eider","mallard","punk duck"".*

```ruby
def readwordfile(fileToRead)
@wordtable = File.read(fileToRead).split"\n"
words = @wordtable.length

#The following is the reqired code to read the file on MacOS
# wordtable = []

# count = 0
# file = File.open(fileToRead)

# file.each_line do |x|
    # count += 1
    # wordtable.append x.strip
# end

# file.close

# @wordtable = wordtable

# return count

end
```

As seen in the code, the words or phrases is stored in the object *wordtable*, this is done by seting up a array also called *wordtable*, then each line of the text document is stored in the array. And finaly the array is then set to the object *wordtable*. Hence the code passes the test as when the method *readwordfile* is called, the text in the document is stored in the correct object

TEST 18: SECRET WORD

This test is looking at the code and checking for a method called "*gensecretword*".

```ruby
def gensecretword()
word = @wordtable.sample.upcase
end
```

As seen in the code, a method called *gensecretword* has been implemented in ruby with the correct syntaxes. Hence the code passes the test.

TEST 19: RETURNING A SECRET WORD

This test is testing to see if a word from the object *wordtable* has been returned when the method *gensecretword* has been called.

```
def gensecretword()
word = @wordtable.sample.upcase
end
```

As seen in the code, when the *gensecretword* method is invoked, a ruby predefined function is executed which samples one of the variables, containing the list of words, from the *wordtable* object. Hence this code passes the test as it returns a word from the text file when the *gensecretword* method is invoked.

TEST 20: UPPER CASE AND LOWER CASE

This testing is testing to see if a true or false value is returned from the *gensecretword* method, depending on if the word is uppercase or lowercase, when the *checkwordupcase* method is called.

```
def checkwordupcase?()
word = gensecretword()
    if (word == word.upcase)
    isup = true
    else
    isup = false
    end
end
```

As seen in the code, the variable *word* stores a radom word from the text file by invoking the *gensecretword* method. Then the code checks the uppercase value of the *word* variable by using a ruby pre-defined function, and if it is true then it stores the ture Boolean value in a variable. Hence the code passes the test as it will return a true or false value depending on if the word is upper case or lower case.

TEST 21: SET THE SECRET WORD

This test is testing that a new method has been created called *setsecretword*, and that this method stores a value passed into it in an object called *secretword*

```
def setsecretword(setword)
@secretword = setword
end
```

As seen in the code, a method called *setsecretword* has been defiened. And on the second line the code takes the parameter passed in, and stores it in the object *secretword*. Hence this code passes the test as the correct method has been defined and the pramiter passed into the method is stored in the correct object.

TEST 22: GET THE SECRET WORD

This test is testing to see if a method called *getsecretword* has been created, and it is testing that the method return the value of the object *secretword.*

```
def getsecretword()
@secretword
end
```

As seen in the code, a new method has been defined called *getsecretword.* And followingly the method on line 2 is the object *secretword.* Hence this code passes the test as the correct method is created. And when the method is executed, the object on line two passes its value to the method which then passes the value back to the caller. Hence passing the test.

TEST 23: WORD TEMPLATE

This test is testing that a template has been created for the word, where the template represents how long the word is.

```
def createtemplate()
 @template= "[" + "_"*@secretword.length+"]"
end
```

As seen in the code, a template is created by firstly having square brackets surround the word that will be templated. Then to represent the length of the word, a predefined method is called to find the length of the word. Then the code mutiplys the resulting number of characters by the underscore sting, so as to have the number of underscore characters the same as the number of the word chracters. This is all combined together in one string and stored in the object *template.* Hence this code passes the test as a template of the word will have been crated where the template represents how long the word is.

TSET 24: THE WORD AND IT'S TEMPLATE IN THE ARRAY

This test is testing that a word and it counter part template is returned within an array when the method *getsecrettemplate* is called.

```
def getsecrettemplate()
return [getsecretword, @template]
end
```

As seen in the code, when the method *getsecrettemplate* is called, it executes the second line which passes out the array. Where the first index of the array is the *gestsecretword* method, which inturn invokes the method and passes back the object value *secretword* into the first index of the array. The second index of the array is the *template* object which is the template representing how long the *secretword* is. Hence this code passes the test as an array of the word and its template is returned when the method *getsecrettemplate* is called.

TEST 25: YOUR TURN!

This test is testing that a method called *incrementturn* has been created, where this method increases the value of the object *turn* by 1 when it is called.

```
def incrementturn()
@turn = turn + 1
end
```

As seen in the code, a method called *incrementturn* has been defined. And when the method is called it executes the code so that the value of the object *turn* increase by 1. This is done by setting the object to the itself plus 1. Hence the code passes the test has the correct method has been defined, and the object *turn* increase by 1 when the method is called.

TEST 26: NO TURNS LEFT FOR YOU

This test is testing that a method called *getturnsleft* has been created, where this method decreases the value of the object *turnsleft* by 1 when it is called.

```
def getturnsleft()
@turnsleft = GOES - @turn
end
```

As seen in the code, a method called *getturnsleft* has been defined. And when the method is called it executes the code so that the value of the object *turnsleft* decreases by 1. This is done by setting the object to the variable GOES minus the object *turn*. Thus when the object value of *turn* increases by 1, this value is subtracted to the object *turnsleft,* hence reducing this value by 1. Hence the code passes the test has the correct method has been defined, and the object *turn* decreases by 1 when the method is called.

TEST 27: YOU WIN

This test is testing that a value of “Well done. You win.” is returned when the Boolean value of true is passed into the method *displaywinner.*

```
def displaywinner(value)

if value == true
    won = "Well done. You win."
elsif value == false
    won = "Sorry, computer wins."
end
```

As seen in the code, the valule passed into the methdod *displaywinner* is checked with an if statement to see if the value passed it is the Boolean value of true. If this is the case then the code executes the next line where it stores the value of “Well done. You win.” In the *won* variable. Hence the code passes the test as when a the true value is passed into the method, it returns the correct value.

TEST 28: YOU LOSE

This test is testing that a value of "*Sorry, computer wins*." Is returned when the Boolean value of false is passed into the method *displaywinner.*

```
def displaywinner(value)

if value == true
    won = "Well done. You win."
elsif value == false
    won = "Sorry, computer wins."
end
```

As seen in the code, when a value that is not true is passed in, it doesn't meet the requirement of the second if statement. Hence when a value that is false is passed into the method, it checks the next conditional statement. The value of false is equal to false, so it runs the code inline with the condition. This then sets the variable *won* to the value *"Sorry, computer wins*.". Hence the code passes the test as when a the false value is passed into the method, it returns the correct value.

TEST 29: ROLL THE CREDITS

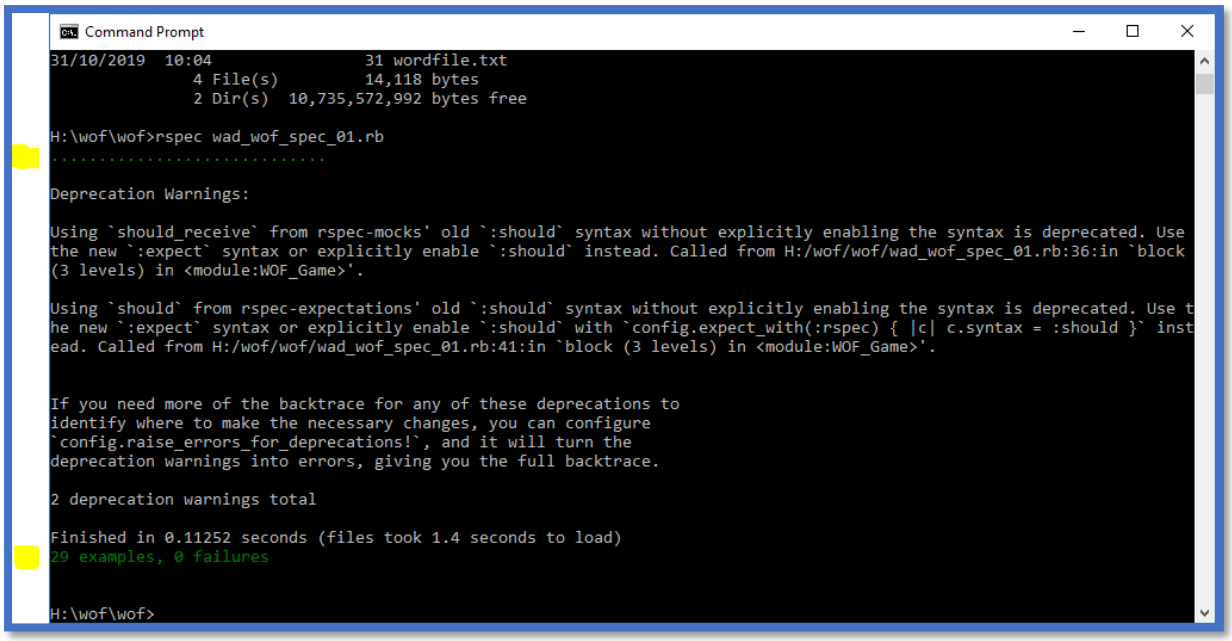
This test is testing that the names and student id's of all who have worked on the project are returned by index when the method *displaycredits* is called

```
def displaycredits(i, names, ids)
namescan = names.split(/,/)
idscan = ids.split(/,/)
namescan[i] + " ("+idscan [i] + ")"
end
```

As seen in the code, when the method *displaycredits* is called, the method splits all the names passed into it by comma and sets it to the variable *namescan*. The method also spits all the students id passed into it by comma and stores it in the variable *idscan.* Then the code returns both of these values in the form of an array. Hence the code passes the test, as the names of all the students and their ids who worked on the projects are returned by index when the method is called.

EVIDENCE OF CODE PASSING TESTS

As highlighted below, our code passes all tests. This is represented by the 29 green dots, where each dot is a passed test. Also noted towards the bottom of the command prompt, it is highlighted that the code as 0 failures. Thus further showing that the code passes all the tests.


Command Prompt
31/10/2019 10:04 31 wordfile.txt
4 File(s) 14,118 bytes
2 Dir(s) 10,735,572,992 bytes free
H:\wof\wof>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 H:/wof/wof/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 H:/wof/wof/wad_wof_spec_01.rb:41:in `block (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 0.11252 seconds (files took 1.4 seconds to load)
29 examples, 0 failures
H:\wof\wof>