Assignment 3

Fredrik Langå

fl222pw@student.lnu.se

https://github.com/Trasis33/fl222pw_1dv600

Use Cases

UC 1 Start Game

Precondition: none.

Postcondition: the game menu is shown.

Main scenario

1. Starts when the user wants to begin a session of the hangman game.
2. The system presents the main menu with a title, the option to play and quit the game.
3. The Gamer makes the choice to start the game.
4. The system starts the game (see Use Case 2).

Alternative scenarios

3.1 The Gamer makes the choice to quit the game.

The system quits the game (see Use Case 2)

UC 2 Play game

Precondition: The user has pressed the key corresponding to “Start Game”

Postcondition: the game is finished, asks to play again.

Main scenario

1. Starts when the user has started the game from the menu.
2. The system has chosen a word and prompts the user for input.
3. The user inputs a letter.
4. The system checks if the word contains the letter.
5. The system displays the result and number of guesses left.
6. The user is prompted for input.

Repeat from step 3

Correct guess scenario

6.1 The word is displayed, and the user is prompted to play again or quit.

Test Plan

Objective

The objective of this test plan is to test the code which was implemented in the last iteration during assignment 2.

What to Test

For unit tests I decided to test the function that generates a random word and the word class since without a word you don't have a game. Both originate from Use Case 2 – Play game, and I will be both running manual dynamic test cases and also write automated unit tests.

Time Plan

Task	Estimated	Actual
Manual Test Cases	2h	1h
Unit Tests	1h	3h
Running Manual Tests	10m	2m
Inspect Code	30m	25m
Test Report	1h	10m

Manual Test-Cases

TC1.1 From UC1 Start Game

Use case: UC1 Start Game

Scenario: The player successfully starts the game.

The main scenario of use case 1 is tested where a player successfully starts a game of hangman from a presented menu.

Pre-condition: none

Test steps:

- Start the application
- System displays "Want to play a game? (Use arrow keys) followed by answers "Yes" and "No" on separate lines
- Press Enter to select "Yes" which is the default option

Expected:

- System displays "guess a letter" and awaits input

TC2.1 From UC2 Play Game

Use Case: UC2 Play Game

Scenario: The player completes a round of the game

The main scenario of use case 2 is tested where a complete round of the game is played and the correct word is known. For the purpose of this test the words able to be chosen by the game has been limited to the word “test”.

Pre-condition: Use case 1 has been followed through, i.e. the player has started the game.

Test steps:

- System displays “guess a letter” and awaits input
- Press “t” and then press enter
- System displays “t _ _ t” and “remaining guesses: 9” on a new line
- Press “e” and then press enter
- System displays “t e _ t” and “remaining guesses: 9” on a new line
- Press “s” and then press enter

Expected:

- System displays “t e s t” and “remaining guesses: 9” on a new line, as well as “you win!” on a new line
- System displays main menu
- Press down arrow key and press enter
- System displays “Thank you, come again”

Comments:

Test Report

Test	UC1	UC2
TC1.1	1/OK	0
TC2.1	0	1/OK
COVERAGE&SUCCESS	1/OK	1/OK

Test	hangman	getWord	Word	letter
COVERAGE&SUCCESS	0%/FAIL	100%/OK	100%/OK	0

Comments:

The automated tests for “getWord” and “Word” functions as specified, the failing test on “hangman” is because the function is not yet implemented.

Automated Unit Tests

Test Code:

```
const sut = require('../src/hangman')
const getWord = require('../src/getWord')
const Word = require('../src/word')
const Letter = require('../src/letter')
const assert = require('chai').assert
const expect = require('chai').expect

describe('getWord', () => {
  it('should return a string', () => {
    expect(getWord.getWord()).to.be.a('string')
  })

  it('Should be a random word from an array', () => {
    let word1 = getWord.getWord()
    let word2 = getWord.getWord()

    assert.notDeepEqual(word1, word2)
  })
})

describe('word', () => {
  it('should return underscores if no letters has been guessed', () => {
    let word = new Word('test')
    word.setUp()

    expect(word.updateWord()).to.contain('_ _ _ _')
  })
```

```
20
21  describe('word', () => {
22    it('should return underscores if no letters has been guessed', () => {
23      let word = new Word('test')
24      word.setUp()
25
26      expect(word.updateWord()).to.contain('_ _ _ _')
27    })
28
29    it('should be a word object', () => {
30      let word = new Word('test')
31
32      word.setUp()
33      word.updateLetters()
34
35      let firstLetter = word.letterObjArray[0].value
36
37      expect(firstLetter).to.contain('t')
38
39      let letter = new Letter(word, false)
40      letter.showLetter()
41
42      expect(word).to.be.an('object')
43    })
44  })
45
46  describe('graphics', () => {
47    it('should print a string', () => {
48      expect(sut.graphics()).to.be.a('string')
49    })
50  })
51
```

Manual Test Results:

```
fredr@LAPTOP-1R09OK7N MINGW64 ~/Documents/1dv600/fl222pw_1dv600 (master)
$ npm start

> fl222pw_1dv600@1.0.0 start C:\Users\fredr\Documents\1dv600\fl222pw_1dv600
> node app.js

? Want to play a game? Yes
? guess a letter t
t _ _ t
remaining guesses: 9
? guess a letter e
t e _ t
remaining guesses: 9
? guess a letter s
t e s t
remaining guesses: 9
you win!
? Want to play a game? No
Thank you, come again

fredr@LAPTOP-1R09OK7N MINGW64 ~/Documents/1dv600/fl222pw_1dv600 (master)
$
```

Automatic Test Results:

```
fredr@LAPTOP-1R09OK7N MINGW64 ~/Documents/1dv600/fl222pw_1dv600 (master)
$ npm test

> fl222pw_1dv600@1.0.0 test C:\Users\fredr\Documents\1dv600\fl222pw_1dv600
> mocha tests/test.js



  getWord
    √ should return a string
    √ Should be a random word from an array

  word
    √ should return underscores if no letters has been guessed
    √ should be a word object

  graphics
    1) should print a string


  4 passing (24ms)
  1 failing

  1) graphics
       should print a string:
     AssertionError: expected undefined to be a string
      at Context.it (tests\test.js:52:34)



npm ERR! Test failed.  See above for more details.
```

Reflections

Writing unit tests in JavaScript I thought was quite difficult, especially since I'm using the Inquirer module to use a menu controlled by arrow keys. This meant that it was hard to run a "system under test" since anytime I instantiated a new hangman game from unit tests, the menu would load forcing me to press down arrow key to select "no" and quit the game. I did not want to modify my code just to implement the tests. I look forward to delve deeper in testing for Iteration 4.