Beyond Hangman

Test Reports

Test Plan (Fourth Iteration) 21/3/2019

Objective

The objective is to test the code that has been implemented for both the single-player and multiplayer game modes.

What to test and how

UC2 and UC3 will be tested by writing and running dynamic manual test-cases. The *UsernameDataHandler*, *LivesGame*, and *MultiplayerGame* classes will be tested by writing automated tests. A large portion of these classes' functionality will also be tested naturally through the manual test-cases.

Relevant Use-Cases

UC 2 Play Game – Single Player

Pre-condition: ---

Post-condition: Menu, quit, or play again

Main Scenario

1. The gamer has chosen to play a single-player game.
2. The system randomly selects a word. The system displays remaining lives, the hangman platform, remaining tries, words completed, how to return to the main menu, and the hidden version of the word. The system prompts the gamer to guess a letter.
3. The gamer guesses a letter.
4. The system confirms that this letter has not been previously guessed. The system adds this letter to a list of already guessed letters, which it displays to the user.
5. The letter is correct. The system reveals the spots in the word that this letter corresponds to.
6. The word is complete and the gamer has won the round. The system increments "words completed" value.

Return to step 2.

Alternative Scenarios

3.1 The gamer decides to quit.

1. The system returns to the main menu. (Use Case 1, step 2).

4.1 The system determines that the letter has previously been guessed.

1. Go to step 3.

5.1 The letter is incorrect.

1. The system decrements the value of “remaining tries” and updates the hangman platform.

    1a. “Remaining tries” value is above zero.

        1. Go to step 3.

    1b. “Remaining tries” is zero.

        1. The system decrements the value of “remaining lives”.
            a. Remaining lives value is above zero.
                i. Go to step 2.
            b. Remaining lives value is zero.
                i. The system saves and displays statistics, tells gamer that they have run out of lives and that the game is over. The system returns to the main menu.

UC 3 Play Game – Multiplayer

Pre-condition: ---

Post-condition: Menu, quit, or play again

Main Scenario

1. The gamer has chosen to play a multiplayer game.
2. The system asks gamer #1 to enter a word or quit.
3. Gamer #1 enters a word.
4. The system verifies that the word is valid. The system displays the hangman platform, remaining tries, round number, how to return to the main menu, and the hidden version of the word. The system Gamer #2 to guess a letter.
5. Gamer #2 guesses a letter.
6. The system confirms that this letter has not been previously guessed. The system adds this letter to a list of already guessed letters, which it displays to the user.
7. The system determines the letter is correct. The system reveals the spots in the word that this letter corresponds to.
8. The word is complete and Gamer #2 has won the round. The system increments Gamer #2’s score.

Return to step 4 and switch the roles of Gamer #1 and Gamer #2.

Alternate Scenarios

3.1, 5.1 The gamer decides to quit.

1. The system returns to the main menu. (Use Case 1, step 4).

4.1 The system determines that the word is invalid.

1. Go to step 3.

6.1 The system determines that the letter has previously been guessed.

1. Go to step 5.

7.1 The system determines the letter is incorrect.

1. The system decrements "remaining tries" value.
    a. "Remaining tries" value is above zero.
        i. Go to step 5.
    b. "Remaining tries" value is zero.
        i. The system increments "round number" value.
            1. "Round number" value is 11.
                a. The system determines which player has a higher score and declares them the winner. The gamers could also have the same score and tie. The system returns to the main menu, (Use Case 1, step 4), or the gamers may play again (go to step 2)
        ii. Go to step 4. Switch the roles of Gamer #1 and Gamer #2

8.1 Gamer #1 has a score value of 5.

1. The system proclaims Gamer #1 the winner. The system returns to the main menu (Use Case 1, step 4.), or the gamers may play again (go to step 2).

8.2 Both gamers have a score value of 5.

1. The system proclaims that the gamers have tied. The system returns to the main menu (Use Case 1, step 4.), or the gamers may play again (go to step 2).

8.3 Gamer #2 has a score of 5 and Gamer #1 has a score of less than 4.

1. The system proclaims Gamer #2 the winner. The system returns to the main menu (Use Case 1, step 4.), or the gamers may play again (go to step 2).

Manual Test-Cases

TC2.1 Single-Player Quit

Use Case: UC2 Play Game – Single Player

Scenario: Single-Player Quit

An alternate scenario of UC2 is tested where a user is in the game but decides to quit and return to the main menu rather than play.

Pre-condition: data.txt file is empty.

Test Steps

- Start the application
- The system shows “Register a new username(letters only) -> “
- Enter “Gilbert” and press enter
- The system displays the main menu
- Enter “1” and press enter to begin a normal single-player game.
- The system sets up the game and the round and requests the user to guess a letter.
- Enter “quit” and press enter.
- The system shows “Are you sure you want to quit? ‘yes’ or ‘no’ -> “
- Enter “yes” and press enter.
- The system shows “Would you like to play another game? ‘yes’ or ‘no’ -> “
- Enter “no” and press enter.

Expected

- The system returns to the main menu of the application.

TC 2.2 Single-Player Game

Use Case: UC2 Play Game – Single Player

Scenario: Single-Player Game

A simulation of the main scenario of UC2 is done.

Precondition: Replace the file that the NormalLivesGame class gets its data from (normalWords.txt), with another .txt file called fakeWord.txt. This file will contain only a single line of text with the characters: “aassd”. No spaces and no line breaks. After running all the tests in this document, return NormalLivesGame to its original state with the normalWords.txt file.

Test Steps

- Start the application
- The system shows list of registered users (only one, “Gilbert”, should be shown) and shows “Type the number of your username or type ‘new’ to register a new username -> “
- Enter “1” to select Gilbert and press enter
- The system displays the main menu
- Enter “1” and press enter to begin a normal single-player game.
- The system sets up the game and the round and requests the user to guess a letter.
- Enter “a” and press enter
- The system responds by revealing the letter and requesting another letter.
- Repeat the previous two steps twice, replacing “a” with “s” the first time, and replacing “a” with “d” the last time.
- The system determines that a word has been successfully guessed and increases word solved by one. The system presents a “new” word.
- Enter “q” and press enter. Repeat with “w”, “e”, “r”, “t”.
- The system determines that the word has been failed, decreases remaining lives by one, and presents a new word.
- Repeat the previous two steps two more times.
- The system determines that there are no lives remaining and the game is over. The system shows “Would you like to play another game? ‘yes’ or ‘no’ -> “
- Enter “no” and press enter.

Expected

- The system returns to the main menu of the application.

TC3.1 Multiplayer Quit

Use Case: UC3 Play Game – Multiplayer

Scenario: Multiplayer Quit

An alternate scenario of UC3 is tested where a user is in the game but decides to quit and return to the main menu rather than play.

Precondition: None

Test Steps

- Start the application
- The system shows list of registered users (only one, “Gilbert”, should be shown) and shows “Type the number of your username or type ‘new’ to register a new username -> “
- Enter “1” to select Gilbert and press enter
- The system displays the main menu
- Enter “3” and press enter to begin a multiplayer game
- The system requests that player #1 enter a word for player #2 to guess.
- Enter “aassd” and press enter.
- The system sets up the game and requests player #2 to enter a letter.
- Enter “quit” and press enter.
- The system shows “Are you sure you want to quit? ‘yes’ or ‘no’ -> “
- Enter “yes” and press enter.
- The system shows “Would you like to play another game? ‘yes’ or ‘no’ -> “
- Enter “no” and press enter.

Expected

- The system returns to the main menu of the application.

TC3.2 Multiplayer Player One Win

Use Case: UC3 Play Game – Multiplayer

Scenario: Multiplayer Player One Win

A scenario of UC3 is tested where a multiplayer game is played out, resulting in player #1 winning the game.

Precondition: None

Test Steps

- Start the application
- The system shows list of registered users (only one, “Gilbert”, should be shown) and shows “Type the number of your username or type ‘new’ to register a new username -> “
- Enter “1” to select Gilbert and press enter
- The system displays the main menu
- Enter “3” and press enter to begin a multiplayer game
- The system requests that player #1 enter a word for player #2 to guess.
- Enter “aassd” and press enter.
- The system sets up the round and requests player #2 to enter a letter.
- Enter “q” and press enter. Repeat with the letters “w”, “e”, “r”, “t”.
- The system determines that the round is lost. The system keeps player #2’s score at 0. The system requests player #2 to enter a word.
- Enter “aassd” and press enter.
- The system sets up the round and requests player #1 to enter a letter.
- Enter “a” and press enter. Repeat with “s” and “d”.
- The system determines that the round is won. The system increments player #1’s score to 1. The system requests player #1 enter a word.
- Repeat the process, regardless of which player’s turn it is, of entering “aassd” at the beginning of a round, and entering “a”, “s”, “d” each round in order for all the rounds to count as victories. Continue this until player #2 has 4 points and player #1 has 5 points.
- The system determines that player #1 is the winner. The system shows “Would you like to play another game? ‘yes’ or ‘no’ -> “
- Enter “no” and press enter.

Expected

- The system returns to the main menu of the application.

TC3.3 Multiplayer Player Two Win

Use Case: UC3 Play Game – Multiplayer

Scenario: Multiplayer Player Two Win

A scenario of UC3 is tested where a multiplayer game is played out, resulting in player #2 winning the game.

Precondition: None

Test Steps

- Start the application
- The system shows list of registered users (only one, “Gilbert”, should be shown) and shows “Type the number of your username or type ‘new’ to register a new username -> “
- Enter “1” to select Gilbert and press enter
- The system displays the main menu
- Enter “3” and press enter to begin a multiplayer game
- The system requests that player #1 enter a word for player #2 to guess.
- Enter “aassd” and press enter.
- The system sets up the round and requests player #2 to enter a letter.
- Enter “a” and press enter. Repeat with “s” and “d”.
- The system determines that the round is won. The system increments player #2’s score to 1. The system requests player #2 to enter a word.
- Enter “aassd” and press enter.
- The system sets up the round and requests player #1 to enter a letter.
- Enter “q” and press enter. Repeat with the letters “w”, “e”, “r”, “t”.
- The system determines that the round is lost. The system keeps player #1’s score at 0. The system requests player #1 enter a word.
- Repeat the process, regardless of which player’s turn it is, of entering “aassd” at the beginning of a round, and entering “a”, “s”, “d” each round in order for all the rounds to count as victories. Continue this until player #2 has 5 points.
- The system determines that player #2 is the winner. The system shows “Would you like to play another game? ‘yes’ or ‘no’ -> “
- Enter “no” and press enter.

Expected

- The system returns to the main menu of the application.

TC3.4 Multiplayer Game Tie

Use Case: UC3 Play Game – Multiplayer

Scenario: Multiplayer Game Tie

A scenario of UC3 is tested where a multiplayer game is played out, resulting in a tie between the two players

Precondition: None

Test Steps

- Start the application
- The system shows list of registered users (only one, “Gilbert”, should be shown) and shows “Type the number of your username or type ‘new’ to register a new username -> “
- Enter “1” to select Gilbert and press enter
- The system displays the main menu
- Enter “3” and press enter to begin a multiplayer game
- The system requests that player #1 enter a word for player #2 to guess.
- Enter “aassd” and press enter.
- The system sets up the round and requests player #2 to enter a letter.
- Enter “a” and press enter. Repeat with “s” and “d”.
- The system determines that the round is won. The system increments player #2’s score to 1. The system requests player #2 to enter a word.
- Repeat the process, regardless of which player’s turn it is, of entering “aassd” at the beginning of a round, and entering “a”, “s”, “d” each round in order for all the rounds to count as victories. Continue this until player #2 and player #1 both have 5 points.
- The system determines that the result of the game is a tie. The system shows “Would you like to play another game? ‘yes’ or ‘no’ -> “
- Enter “no” and press enter.

Expected

- The system returns to the main menu of the application.

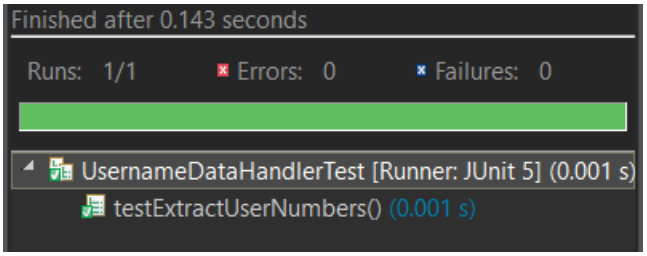
Manual Test Case Report

Test Traceability and Success Matrix

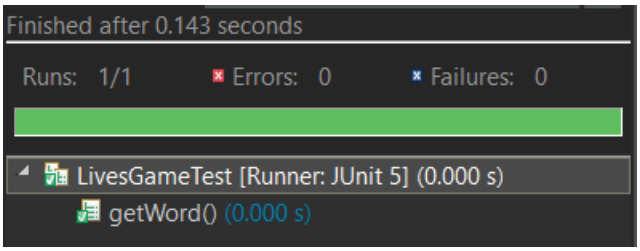
Test	UC2	UC3
TC2.1	1/OK	0
TC2.2	1/OK	0
TC3.1	0	1/OK
TC3.2	0	1/OK
TC3.3	0	1/OK
TC3.4	0	1/OK
Coverage and Success	2/OK	4/OK

Automated Test Cases

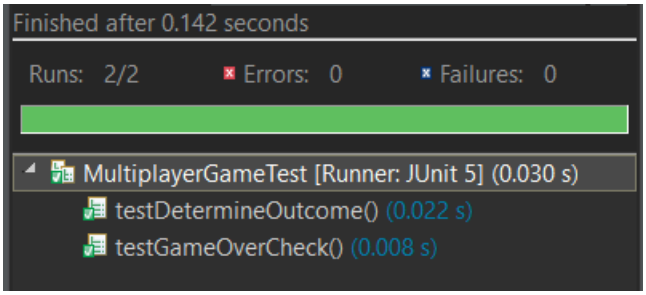
UsernameDataHandler Class Test Results



LivesGame Class Test Results



Multiplayer Class Test Results



Test Plan (Third Iteration) 8/3/2019

Objective

The objective is to test the code that has been implemented thus far.

What to test and how

UC1 will be tested by writing and running dynamic manual test-cases. Note that only one game mode is partially implemented, and that the leaderboard feature has also not been implemented. That means UC2 and UC3 are currently not implemented, and all will be tested at a later time after they have been implemented. UC4 and UC5 will also be tested by writing and running dynamic manual test-cases. The *Game*, *Round*, and *User* classes will be tested by writing automated tests.

Time Plan

Task	Estimated	Actual
Manual TC	2h	2.5h
Unit Tests	3h	5h
Running Manual Tests	30m	15m
Test Report	1h	3h

Relevant Use-Cases

UC 1 Start Game

Pre-condition: none

Post-condition: the game menu is shown.

Main Scenario

1. Starts when the gamer wants to begin a session of the hangman game.
2. The system asks the gamer to select their name from a list of names already registered to the system, or to enter a new name if this is the gamer's first time playing.
3. The gamer selects their name.
4. The system presents a main menu with the option to play and the option to quit the game.
5. The gamer chooses to start a game.
6. The system starts the game. (See Use Case 2)

Alternative Scenario

3.1 Gamer's first time playing.

1. Gamer registers. (See Use Case 4)

4.1 The gamer chooses to quit the game.

1. The system quits the game. (See Use Case 5)

UC 4 Register Player

Pre-condition: First time player

Post-condition: New username registered. Main menu displayed.

Main Scenario

1. Gamer is playing for the first time.
2. System requests the gamer to give a username
3. Gamer gives a username
4. System finds the username to be valid. New gamer added to existing list of gamers. System moves Gamer to main menu.

Alternate Scenarios

4.1 System finds that the username has invalid characters.
4.1.1 System notifies gamer that the username has invalid characters. Return to step 2.

UC 5 Quit Game

Precondition: The game is running.

Postcondition: The game is terminated.

Main scenario

1. Starts when the user wants to quit the game.
2. The system prompts for confirmation.
3. The user confirms.
4. The system terminates.

Alternative scenarios

3.1. The user does not confirm

1. The system returns to its previous state

Manual Test-Cases

TC1.1 Start Game Successful (pre-registered User)

Use Case: UC1 Start Game

Scenario: Start Game Successful

The main scenario of UC1 is tested where a user selects their username and begins a round of Hangman.

Precondition: There must be a single pre-existing user, “Robby”, in the “data.txt” file. This can be achieved by ensuring the txt file has only one line that reads exactly (without quotes):

“Robby | Words Solved All Time: 0”

Test Steps

- Start the application
- The system shows list of registered users (only one, “Robby”, should be shown) and shows “Type the number of your username or type 'new' to register a new username ->”
- Enter “1” to select Robby and press enter.
- The system shows “Welcome to Hangman, Robby!” as well as the options

  “[1] Play
  [2] Quit”
- Enter “1” to Play and press enter.

Expected

- The system begins a round of hangman

TC1.2 Start Game Quit

Use Case: UC1 Start Game

Scenario: Start Game Quit

An alternate scenario of UC1 is tested where a user selects their username but decides to quit the application, rather than play.

Precondition: Same as TC1.1

Test Steps

- Start the application
- The system shows list of registered users (only one, “Robby”, should be shown) and shows “Type the number of your username or type 'new' to register a new username ->”
- Enter “1” to select Robby and press enter.
- The system shows “Welcome to Hangman, Robby!” as well as the options
  “[1] Play
  [2] Quit”
- Enter “2” to Quit and press enter.

Expected

- Shows output from UC5

TC1.3 Start Game to Register (no registered users/first user)

Use Case: UC1 Start Game

Scenario: No registered users. Must register before starting

An alternate scenario of UC1 where the “data.txt” file is empty and a player is forced to register a new username before playing

Precondition: The file “data.txt” must be completely empty.

Test Steps

- Start the application.

Expected

- The system shows that the there are no registered usernames.
- The system enters UC4.

TC1.4 Start Game and Register (there is a pre-existing user)

Use Case: UC1 Start Game

Scenario: There is a pre-existing user, but the current user would like to register as a new user.

An alternate scenario of UC1 where the “data.txt” file contains one existing user, “Robby”, and the user would like to register a new username rather than continuing as “Robby”

Precondition: Same as TC1.1

Test Steps

- Start the application.
- The system shows list of registered users (only one, “Robby”, should be shown) and shows “Type the number of your username or type 'new' to register a new username ->”
- Enter “new” and press enter.

Expected

- The system enters UC4.

TC4.1 Register Player (Correct entry)

Use Case: UC4 Register Player

Scenario: Player successfully registers

The main scenario of UC5 where a player has chosen to register a new username and enters a valid username.

Precondition: The “data.txt” file must be completely empty. This scenario entered by first following the steps of TC1.3.

Test Steps

- The system shows “Register a new username (letters only) ->“
- Enter “Tom” and press enter.

Expected

- The system registers new user “Tom” to the “data.txt” file.
- The system shows “Welcome to Hangman, Tom!”
- The system shows the main menu, “[1] Play [2] Quit”

TC4.2 Register Player (Failed initial entry)

Use Case: UC4 Register Player

Scenario: Player fails to register on first attempt, succeeds on second.

An alternate scenario of UC5 where a player has chosen to register a new username and enters an invalid username on the first attempt, and a valid username on the second.

Precondition: The same as TC4.1.

Test Steps

- The system shows “Register a new username (letters only) ->“
- Enter “Tom1” and press enter.
- The system shows “*Only letters are valid in a username*\n Try again ->”
- Enter “Tom” and press enter.

Expected

- The system registers new user “Tom” to the “data.txt” file.
- The system shows “Welcome to Hangman, Tom!”
- The system shows the main menu, “[1] Play [2] Quit”

TC5.1 Quit Game

Use Case: UC5 Quit Game

Scenario: User quits from main menu.

In the main scenario of UC5 the player chooses to quit the game, confirms quitting, and the program terminates.

Precondition: This scenario is entered by first following the steps of TC1.2.

Test Steps

- The system shows “Are you sure you want to quit? ‘yes’ or ‘no’ -> “
- Enter ‘yes’ and press enter.

Expected

- The system shows “Thanks for playing, Robby!”
- The program terminates.

TC5.2 Quit Game back to Main Menu

Use Case: UC5 Quit Game

Scenario: User quits from menu, but returns to the main menu.

An alternate scenario of UC6 where the player chooses to quit the game, but does not confirm, and is returned to the main menu.

Precondition: This scenario is entered by first following the steps of TC1.2.

Test Steps

- The system shows “Are you sure you want to quit? ‘yes’ or ‘no’ -> “
- Enter ‘no’ and press enter.

Expected

- The system returns to showing “[1] Play [2] Quit”.

Automated Test-Cases

Round Class Tests

computeLengthMinusWhiteSpace() Method

```
int computeLengthMinusWhitespace() {
    int lengthMinusWhite = word.length();

    for (int i = 0; i < word.length(); i++) {
        if (word.charAt(i) == ' ') {
            lengthMinusWhite--;
        }
    }

    return lengthMinusWhite;
```

computeLengthMinusWhiteSpace Test

```
@Test
void testComputeLengthMinusWhiteSpaceShortAndLongWords() {

    /*
     * Tests strings of length 3
     */

    Random rand = new Random();

    for (int i = 0; i < 10000; i++) { // Run test 10000 times

        StringBuilder word = new StringBuilder("");
        int whiteChars = 0;

        for (int k = 0; k < 3; k++) { // Calculate whitespace in strings of length 3

            int charIntValue = rand.nextInt(40) + 95;
            char insert = ' ';

            // Determine what is inserted
            if (charIntValue >= 97 && charIntValue <= 122) // If the value is in this range, insert corresponding
                                                          // letter
                insert = (char) charIntValue; // Else, insert space
            else
                whiteChars++;

            word.append(insert);

        }

        round = new Round(word.toString(), fakeDataScanSetup());

        // Compare Method to the the length of the 'word' minus the amount of whitespace
        // characters
        assertTrue(word.toString().length() - whiteChars == round.computeLengthMinusWhitespace());

    }
```

**Shows first half of test method. Second half is identical, except that it takes very large strings as input rather than very short strings. There is only one test for this method.*

validateGuessedLetter() Method

```
boolean validateGuessedLetter() {       // Changed access modifier for JUnit test

    while (letterGuess.length() != 1 || !(Character.isLetter(letterGuess.charAt(0))) || guessed.toString().contains(letterGuess)) {

        if (letterGuess.toLowerCase().equals("quit")) {
            return false;

        } else if (letterGuess.length() != 1 || !(Character.isLetter(letterGuess.charAt(0)))) {
            System.out.print("You didn't enter a valid option. Try again -> ");
            letterGuess = scan.nextLine();

        } else if (guessed.toString().contains(letterGuess)) {
            System.out.print("You've already guessed this letter. Try again -> ");
            letterGuess = scan.nextLine();

        }

    }

    return true;

}
```

validateGuessedLetter Tests

```
@Test
void testValidateGuessedLetterCorrectEntries() {

    round = new Round("a", fakeDataScanSetup());

    assertTrue(round.validateGuessedLetter());


}

@Test
void testValidateGuessedLetterQuit() {

    round = new Round("a", quitScanSetup());

    assertFalse(round.validateGuessedLetter());

}
```

**Tests take fake data from files acting as a user would from a System.in Scanner. The first test verifies that the while loop continues and exits and returns true when a valid letter is finally guessed. The second test verifies that the user has entered "quit" and returns false.*

User Class Tests

isValidUsername(String name) Method

```
boolean isValidUsername(String name) {

    if (name.length() == 0)
        return false;
    else if (name.length() > 20)
        return false;

    char[] usernameChars = name.toCharArray();

    for (char c : usernameChars) {

        if (!Character.isLetter(c))
            return false;

    }

    return true;

}
```

isValidUsername Tests

```
@Test
void testLengthZeroUsername() { // Test #1

    for (int i = 0; i < 10000; i++)
        assertFalse(user.isValidUsername(randomLetterString(0)));

}

@Test
void testLengthOver20Username() { // Test #2

    for (int i = 0; i < 10000; i++)
        assertFalse(user.isValidUsername(randomLetterString(rand.nextInt(20) + 21))); // Random length between 21-40

}

@Test
void testInvalidCharsUsername() {   // Test #3

    for (int i = 0; i < 10000; i++)
        assertFalse(user.isValidUsername(randomNonLetterString(rand.nextInt(20) + 1))); // Random non letter string between 1-20 length

}

@Test
void testValidUsername() {  // Test #4

    for (int i = 0; i < 10000; i++)
        assertTrue(user.isValidUsername(randomLetterString(rand.nextInt(20) + 1)));

}
```

**Four tests of the same method. The first three test the three different scenarios in which the method returns false. The last tests the scenario in which the method returns true.*

Game Class Tests

verifyWord(String word) Method

```
public static boolean verifyWord(String word) {

    /*
     * A word/phrase can consist only of letters and spaces.
     *  - No more than two of the same letters consecutively
     *  - No more than one consecutive space
     *  - Word/Phrase must contain at least 5 letters
     *  - Word/Phrase must contain no more than 40 characters.
     *
     */

    if (computeLengthMinusWhitespace(word) < 5) {
        return false;
    } //else if (word.length() > 40) {      // Commented out for
        return false;                        assignment
    }

    int sameLetterInARow = 1;
    char currentChar = '\u0000';
    char previousChar = word.charAt(0);
    final char SPACE_CHAR = (char) 32;

    // Same three letters consecutive or two spaces consectutive, or non letter

    char[] wordArray = word.toCharArray();

    for (int i = 1; i < wordArray.length; i++) {

        currentChar = wordArray[i];

        if (!Character.isLetter(currentChar) && currentChar != SPACE_CHAR)  // Invalid character
            return false;


        if (currentChar == previousChar && currentChar != SPACE_CHAR)
            sameLetterInARow++;
        else
            sameLetterInARow = 1;


        if (sameLetterInARow == 3)  // 3 same letters in a row
            return false;

        if (currentChar == SPACE_CHAR && previousChar == SPACE_CHAR)    // 2 spaces in a row
            return false;

        previousChar = currentChar;

    }

    // If the word makes it through all that, return true.

    return true;
```

verifyWord Tests

```
@Test
void testForThreeConsecutiveSameLetters() {

    assertFalse(Game.verifyWord("This is an invaaalid entry"));
    //                                      ^^^
}

@Test
void testForTwoConsecutiveSpaces() {

    assertFalse(Game.verifyWord("This is an  invalid entry"));
                                        // ^^
}

@Test
void testForFiveLetters() {

    assertFalse(Game.verifyWord("f a l s "));
    //                       Too few letters
}

@Test
void testForMoreThanFortyCharacters() {

    assertFalse(Game.verifyWord("abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"));
    //                              Too many letters
}

@Test
void testForNonLetterChars() {

    assertFalse(Game.verifyWord("This is a fals3 entry"));
                                        //    ^
}

@Test
void testValidWordEntry() {

    assertTrue(Game.verifyWord("This is a valid entry"));

}
```

**Six tests for the same method. The first five test the five different scenarios in which the method returns false. The last tests the scenario in which the method returns true. The fourth, "testForMoreThanFortyCharacters()", has had its code commented out in order to fulfill the requirement of having one test fail.*

Test Report

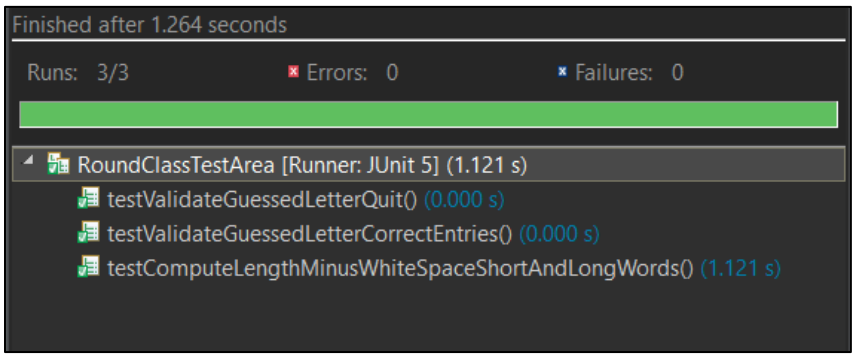
Manual Test Cases

Test Traceability and Success Matrix

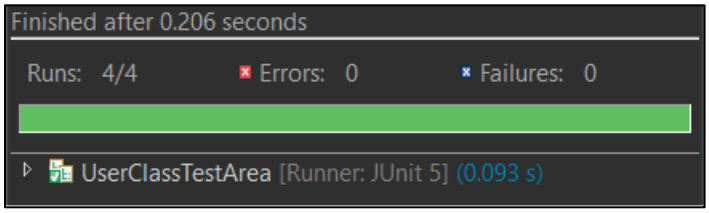
Test	UC1	UC5	UC6
TC1.1	1/OK	0	0
TC1.2	1/OK	0	1/OK
TC1.3	1/OK	1/OK	0
TC1.4	1/OK	0	0
TC5.1	0	1/OK	0
TC5.2	0	1/OK	0
TC6.1	0	0	1/OK
TC6.2	0	0	1/OK
Coverage and Success	4/OK	3/OK	3/OK

Automated Test Cases

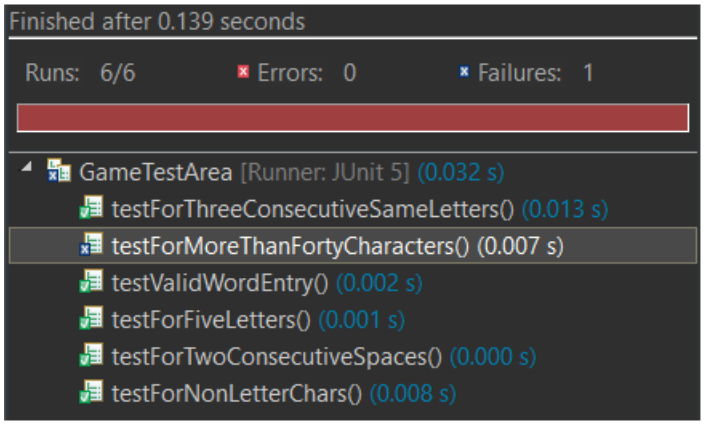
Round Class Test Results



User Class Test Results



Game Class Test Results



*Note that this test tests the same method six times, but there is only one failure, intentionally done, in the 'testForMoreThanFortyCharacters()' test. Note that this does not affect the outcome of any of the other tests.

Reflection

This has been an eye-opening experience. My gut tells me that we haven't been directed to make our code testable in the past so that we could get to this assignment and realize how difficult it is to test your code when you don't write code with testing in mind. I have done automated testing on all the methods I found to be testable in that manner. That turns out to only be four. In the future I will definitely keep testing in mind when writing code, and hope to get a better grasp on all this in the future.