```java
/* Wordle Test Class
* Tester: Jordan Lim
* Purpose: To test and ensure a 90% or greater class coverage of the
provided Wordle game.
*/

package test;
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

import main.WordleGame;
import main.Feedback;

public class WordleTest {

//Feedback on all upper case guesses
@Test
void testGameCaps() {
var game = new WordleGame();
game.setSecretWord("APPLE");
var fb = new Feedback("CIDER", game.makeGuess("CIDER").getPattern() ,
"APPLE");
assertEquals("BBBYB",fb.getPattern(),"Feedback on all uppercase
guesses");
}

//Feedback on all lower case guesses
@Test
void testLowerCase() {
var game = new WordleGame();
game.setSecretWord("APPLE");
var fb = new Feedback("brave", game.makeGuess("brave").getPattern() ,
"APPLE");
assertEquals("BBYBG",fb.getPattern(),"Feedback on all lowercase
guesses");
}
```

```
//Feedback on duplicate letter guesses
@Test
void testDupes() {
var game = new WordleGame();
game.setSecretWord("APPLE");
var fb = new Feedback("EAGLE", game.makeGuess("EAGLE").getPattern() ,
"APPLE");
assertEquals("BYBGG",fb.getPattern(),"Feedback on duplicate letter
guesses");
}

//Feedback on other than a 5 letter word
@Test
void testLetterAmount() {
var game = new WordleGame();
IllegalArgumentException thrown = assertThrows(
 IllegalArgumentException.class,
 () -> game.makeGuess("APPLES")
 );
assertEquals("Guess must be exactly 5
letters.",thrown.getMessage(),"Feedback on a > 5 letter word");
thrown = assertThrows(
 IllegalArgumentException.class,
 () -> game.makeGuess("APPL")
 );
assertEquals("Guess must be exactly 5
letters.",thrown.getMessage(),"Feedback on a < 5 letter word");
}

//Feedback on a intelligible word
@Test
void testDictionary() {
var game = new WordleGame();
IllegalArgumentException thrown;
String[] inputs = {"TLPAE","23ADS", "WOOL!"};
for(String a: inputs) {
thrown = assertThrows(
 IllegalArgumentException.class,
```

```
 () -> game.makeGuess(a)
 );
assertEquals("Word not found in
dictionary.",thrown.getMessage(),"Feedback on a intelligible word");
}
}

//Test attempt counter
@Test
void testAttempts() {
var game = new WordleGame();
game.setSecretWord("APPLE");
for(int i = 0; i < 5; i++) {
game.makeGuess("EAGLE").getPattern();
}
assertEquals(5,game.getAttempts(),"Test attempt counter");
}

//Check if game is over attempts & state exception
@Test
void testGameOver() {
var game = new WordleGame();
game.setSecretWord("APPLE");
for(int i = 0; i < 6; i++) {
game.makeGuess("EAGLE").getPattern();
}
assertEquals(false,game.isGameOver());
var game2 = new WordleGame();
game2.setSecretWord("APPLE");
for(int i = 0; i < 7; i++) {
game2.makeGuess("EAGLE").getPattern();
}
assertEquals(true,game2.isGameOver(),"Check if game is over
attempts");
IllegalStateException thrown = assertThrows(
 IllegalStateException.class,
 () -> new Feedback("EAGLE", game2.makeGuess("EAGLE").getPattern() ,
"APPLE")
 );
```

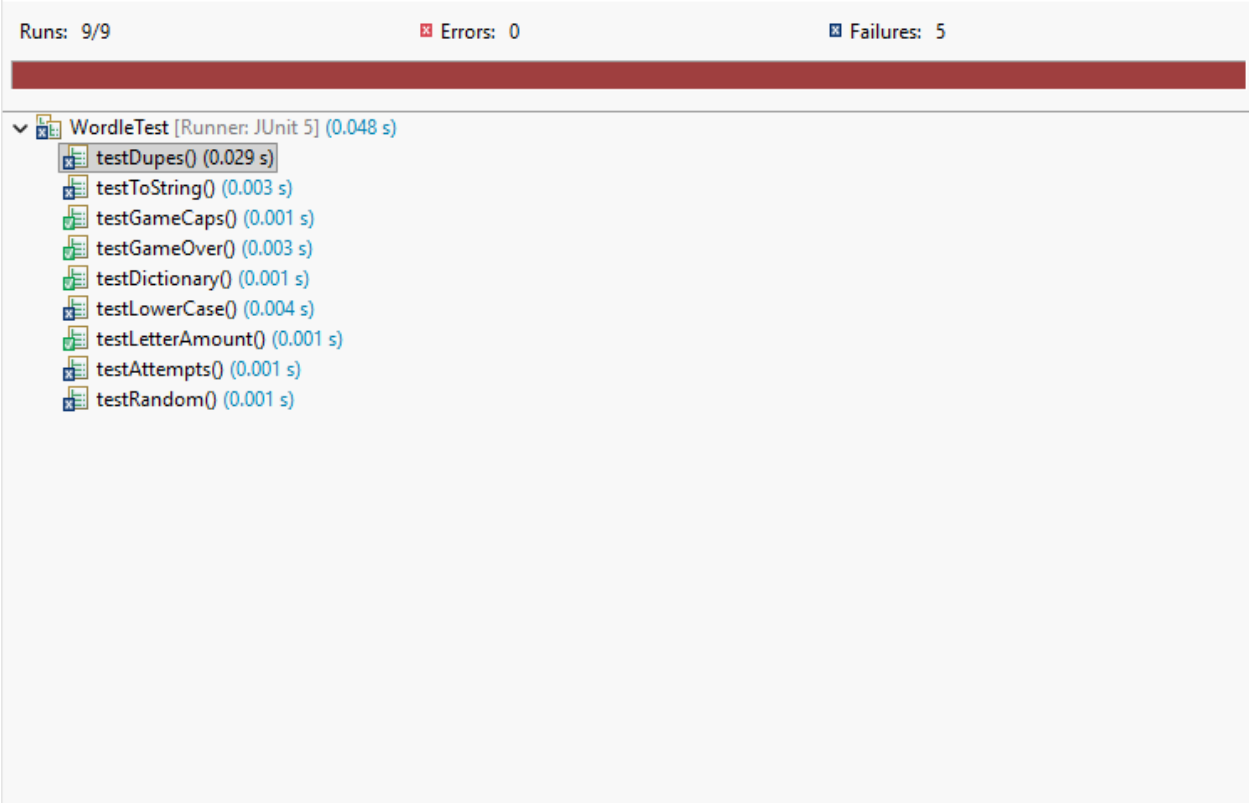
```
assertEquals("Game already ended.",thrown.getMessage(),"Exception
thrown if input ongoing to a ended game");

}

      //Testing the output to terminal
      @Test
      void testToString() {
      var game = new WordleGame();
      game.setSecretWord("APPLE");
      var fb = new Feedback("CIDER",
game.makeGuess("CIDER").getPattern() , "APPLE");
      assertEquals("CIDER\nBBBYB",fb.toString(),"Output Test");
      }

      //Testing random selection
      @Test
      void testRandom() {
var game = new WordleGame();
game.startGame();
String word1 = game.getSecretWord();
      game.startGame();
      String word2 = game.getSecretWord();
      game.startGame();
      String word3 = game.getSecretWord();
      assertEquals(false,word1.equals(word2) &&
word1.equals(word3),"Randomized Selection on Secret Word");

}
}
```



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
src	81.6 %	492	111	603
main	82.1 %	261	57	318
Main.java	0.0 %	0	55	55
Dictionary.java	96.4 %	53	2	55
Feedback.java	100.0 %	27	0	27
WordleGame.java	100.0 %	181	0	181

jlim1336/CS483F25_Assignment-3B

Bugs:

- Duplicate letters showed incorrect pattern
- Mixed upper and lower case guesses not considered
- Dictionary was mixed with upper and lower case words
- Randomized secret letter not implemented
- Output for word and pattern was not optimized for readability
- Did not end the game when the word was guessed right
- Testing attempts were not counted correctly

Experience:

I enjoyed doing code coverage and faced challenges when bug fixing it as I had a little difficulty with the duplicate letter bug but was able to find a suitable algorithm for it. Overall, I have a decent idea of testing and how important it is to do coverage on the entire code.