Write a class called hangman that implements that basic options for the hangman game. Attached s all the code for this class except for the implementation of setUniqueWord() and guessLetter(). Use TDD to provide a complete set of test cases and implementation for these methods. For each test provide a clear common that describes what it does.

To help you see how the class might be used, also attached is an implementation called driveHangManHWSln() using my hangman solution class called HangmanHWSln.

Notice the code uses an interface with the following line.

HangManGetInputWords inW = **new** HangMainGetGameProductionData();

The realization of this interface is provided below. You should also use an interface to create several ‘stub’ realizations of the getInputWords() method.

**import** java.util.ArrayList;  
  
**class** HangMainGetGameProductionData **implements** HangManGetInputWords {  
 @Override  
 **public** ArrayList<String> getInputWords() {  
 ArrayList<String> inWords = **new** ArrayList<>();  
 inWords.add(**"apple"**);  
 inWords.add(**"baker"**);  
 inWords.add(**"charlie"**);  
 inWords.add(**"dog"**);  
 inWords.add(**"fox"**);  
 inWords.add(**"green"**);  
 **return** inWords;  
 }  
}

What to hand in:

1. Your hangman class with the 2 method completely implemented:

setUniqueWord() and guessLetter( **char** guess )

1. A set of test cases that completely test the 2 methods. That is, make sure to provide AT LEAST these test cases:
2. A method that tests that you can create a basic Hangman class called

**void** T1CanCreateBasicClass(){

1. A method that tests that you can create a basic Hangman that uses ONLY these input words:

Apple

Sauce

**void** T2CanRandomlyPickFrom2InWords(){

Note you can test if the output words is either Apple or sauce by:

*assertTrue*( w.equals(**"apple"**) || w.equals(**"sauce"**));

1. A method that tests that you can properly set the displayWord properly from these 2 words:

Apple

Sauce

**void** T3CanSetTheDisplaywordProperly(){

Note you can test if the output words is either Apple or sauce by:

HangmanHWSln hm = **new** HangmanHWSln(nGuess, inWords);  
 String w = hm.getUniqueWord();  
 w = hm.getCurrentWordDisplayString();  
 *assertEquals*(**"\*\*\*\*\*"**, w );

1. A method that tests that you can properly set the displayWord properly from these 1 word:

Apple

**void** T4CanSetTheDisplaywordProperly(){

Note you can test if the output words is either Apple or sauce by:

HangmanHW hm = **new** HangmanHW(nGuess, inWords);  
 String w = hm.getUniqueWord();

*//System.out.printf("\nW=%s", w);*

*assertEquals*(**"apple"**, w);

1. A method that tests that you can properly set the displayWord properly after a guess using only this word as input words

Apple

**void** T5canSetDsplayStringProperlyAfterGuess (){

Note you can test if the output words is either Apple or sauce by:

HangmanHW hm = new HangmanHW(nGuess, inWords);

String w = hm.getUniqueWord();

int c = hm.guessLetter('p');

String ds = hm.getCurrentWordDisplayString();

assertEquals( "\*pp\*\*", ds );

1. A method that tests that you can properly set the displayWord properly after 2 guesses using only this word as input words

Apple

**void** T6canSetDsplayStringProperlyAfterGuess (){

Note you can test if the output words is either Apple or sauce by:

HangmanHW hm = new HangmanHW(nGuess, inWords);

String w = hm.getUniqueWord();

System.out.printf("\n w=%s", w);

int c = hm.guessLetter('p');

c = hm.guessLetter('l');

String ds = hm.getCurrentWordDisplayString();

//System.out.printf("\nW=%s", w);

assertEquals( "\*ppl\*", ds );

1. A method that tests that you properly handle getting a 3rd words with following input list (2 words on the list and you try to get a 3rd word):

Apple

Sauce

**void** T7handleGettingMoreWorkdsThanonList (){

1. A method that tests that you properly handle setting the number of guesses remaining

**void** T8canSetNGuessessRemaining()

1. A method that tests what happens if you set nGuesses to 3 and make 4 wrong guesses

**void** T9TestMoreWrongGuessesThanAllowed()

1. A method that tests that you will not generate the same word twice:

**void** T10TestMoreWrongGuessesThanAllowed()

Note: you can test this by calling w = hm.getUniqueWord() more than once and making sure the same word is not generated twice.

1. A screenshot showing all of these tests completed successfully.