Codewords Game

You are going to create a Codewords game in Python using classes, attributes and methods.

**Rules of The Game:**

For those of you who are unfamiliar with the rules of Codewords, here is a brief explanation. A word or phrase is selected and the letters substituted as other letters (so T might be substituted with E and H for J and so on) A phase like ELDER SCROLLS might end up as TXFTH ZLHYXXZ. The object of the game is to substitute letters to end up with the correct phrase.

Note: you will NOT be using graphics in the assignment. You will use a library like colorama to display colours on the console.

**To do:**

* Create a class diagram for the CodeWordsGame class.
* Create test cases for the CodeWordsGame.
* Code and test the CodeWordsGame.

**The Class Diagram**

BEFORE coding create a class diagram for the class. Make sure you indicate what attributes and methods are private, protected and public. Complete the table below.

|  |
| --- |
| CodeWordsGame |
| + secret\_word: str  + coded\_word: str  + max\_hints: int  - \_hints\_used: int  + max\_guesses: int  + guesses\_used: int  + words: str  + shuffled\_alpha: str  + encoding\_map: dict  + decoding\_map: dict  + letter: str  + substitute: str  + random\_letter: str  + decoded\_letter: str |
| + \_\_init\_\_()  + read\_file() -> list  + select\_word() -> str  + encode\_word(word: str) -> str  + change\_letter(letter: str, substitute: str) -> bool  + get\_hint() -> tuple  + is\_won() -> bool  + is\_lost() -> bool  + \_\_str\_\_() -> str  + \_\_repr\_\_() -> str |

**Test Cases**

BEFORE coding create a test plan for the test cases you are going to use to test the program. Include things like, what you are going to do if the word is not exactly 5 characters long, if the user runs out of guesses or (if you implement this) if the word is not a valid word. Complete the table below. Add as many test cases as you think are necessary to completely test the game.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan for CodeWords** | | | |
| ***Test Case*** | | ***Input Data*** | ***Expected output*** |
|  | Game starts | Word input file | Game starts and displays the instructions. A word is randomly choen from a text file and then encoded. secret\_word and coded\_word are assigned respectively |
|  | Hint given | '?' in text box | Game correctly displays a valid hint and replaces the effected letters in the coded word. \_hints\_used is increased by 1 |
|  | Hint denied (Out of hints) | '?' in text box while max\_nints > \_hints\_used | Game tells the player they are out of hints and doesn't change anything about the coded word |
|  | Guess mode | '!' in text box | Game switches the prompt and allows the user to guess the entire word at once |
|  | Correct gess | Word in text box that is the same as secret\_word | Game ends and congratulates the user for guessing correctly |
|  | Incorrect guess | Word in text box that is not the same as secret\_word | Game tells the user they are incorrect and increases guessed\_used by 1 |
| * a | Game over | Wen the user has used the amount of guesses defined by max\_guesses | Game ends, displays game over, and reveals what the secret word was |
| * E | Exit game | '.' in text box | Game says goodbye and then ends |
|  | Debug info | '\*' in text box | Game displays the information from \_\_repr\_\_ (Secret Word, Coded Word, Max Hints, Hints Used, Max Guesses, Guesses Used+ |
|  | Letter Selection | Letter in text box | Selects a letter for substitution |
|  | Letter Substitution | Letter in text box | Selects a letter to replace the first selection |
|  | Validity checks | Substitution input and replacement input | Checks whether the letter to be substituted in present in coded\_word. Checks to ensure the replacement letter is not present in coded\_word. If both are true, the letters are successfully substituted. |
|  | Game Won | coded\_word matches secret\_word | Congratulates the user and ends the game |
|  |  |  |  |

**The CodeWordsGame Class**

* The CodeWordsGame class has at least the following attributes:
* secret\_word: String with the value of the secret word IN UPPER CASE.
* coded\_word; String with the value of the secret word with substituted letters
* max\_hints: int with the value of the maximum number of attempts. Default is 3.
* hints\_used: int that is not passed to constructor. Set initially to the value of max\_attempts. This should be private or at least protected.
* The CodeWordsGame class has at least the following methods:
* read\_file: Returns a list of the phrases in the provided file codewords.txt
* select\_word: Returns a string of a random phrase selected from the list of words read in from the file.
* Encode\_word: Returns the encoded version of the word or phrase chosen
* make\_guess(guess): Checks if the guess is valid and updates the game state
* mix\_word: Takes the word and encodes it using a random mix of letters.
* is\_won: Returns True if the game is won and False otherwise.
* is\_lost: Returns True if the game is lost (no more guesses allowed) and False otherwise.
* \_\_str\_\_: When object of the class is printed, display the coded word.
* \_\_repr\_\_: Displays the information about the class in the form object\_name(word, coded\_word, max\_hints, hints\_used) in that format.
* \_\_eq\_\_: (maybe) determines if guessed word in the same as the secret\_word
* Other dunder methods as needed
* Other methods as needed

**How the Game Works**

* You have been provided a list of 50 words. When the program starts, initialize an object of type CodeWordsGame.
* The constructor should NOT have to pass anything. The constructor should call the method read\_file to read the file.
* Once the file is read, the constructor calls the method select\_word to select the secret word for this round. .
* The constructor then uses the encode\_word method encode the letters in the phrase. The method uses randomized alphabet to substitute each letter in the phrase for the matching mixed letter. For example (using only the letters A-F, the randomized alphabet might end up as EDABFC so the original word BAD would be coded as DEB and the word FACE would be coded as CEAF
* The constructor then initializes all the remaining attributes of the class.
* The user is provided a welcome message and the coded phrase is displayed.
* The user is prompted to enter which letter to substitute and which letter to substitute it for.
* When the letters are entered, validate that it uses only the letters in the coded phrase and no other letters.
* Substitute the matching letters in the encoded word for the letter it was changed to.
* If the phrase is not completely decoded, prompt the user to enter their next substitution
* If it does match, display a message congratulating the user
* Prompt the user if they want to play again.
* An example of the Console window for the program is shown here.

Welcome to CodeWords. I have encoded a phrase as.

TXFTH ZLHYXXZ

You may get a random hint by specifying ? when prompted for a letter. You have 3 hints remaining

What letter do you want to substitute? T

What letter do you want to replace ‘T’ with? E

**E**XF**E**H ZLHYXXZ

You may get a random hint by specifying ? when prompted for a letter. You have 3 hints remaining

Not solved yet…

What letter do you want to substitute? X

What letter do you want to replace ‘T’ with? L

**EL**F**E**H ZLHY**LL**Z

You may get a random hint by specifying ? when prompted for a letter. You have 3 hints remaining

Not solved yet…

What letter do you want to substitute? ?

I will substitute Z for S. Now your phrase is:

**EL**F**E**H **S**LHY**LLS**

You may get a random hint by specifying ? when prompted for a letter. You have 2 hints remaining

... until the last substation is made

Excellent! You guessed the hidden phrase was ELDER SCROLLS

Would you like to play again (Y/N)? N

* Some extra things to do to challenge yourself:
* Keep track of how many hints were used for each time and provide the user with a list of the number of guesses for each round. Something like:  
  Guesses Count  
   1   
   2 \*\*\* (3)  
   3 \*\*\*\*\*\* (6)
* Provide a way for the user to reset the current phrase back to the start
* Prompt the user for their name so you can refer to them by name.
* Anything else…check with me.