Project 3: Sentence Decoder Game

Your task is to implement a game called Sentence Decoder. Your program will use an input file that contains a set of sentences. For simplicity, the sentences will be all lowercase and have no punctuation. Your program should randomly choose a sentence from the file for each round of the game (a sample is provided on Pilot, but you can also create your own). Let's say the first sentence chosen is `these are the voyages of the starship enterprise`. Your program will then randomly permute the alphabet and replace each letter of the sentence with its permutation. For example, the top line below contains the "real" alphabet, and the bottom line contains the permuted one. In this example, every 's' in the original sentence would be "encoded" as 'p'. Note that your permutation should be randomized so that it is different for every round of the game the user plays.

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
c g j b m v f o i u n t a s w d x k p y e z l q h r
```

Using the above permutation, the full sentence would be this:

```
yompm ckm yom zwhcfmp wv yom pyckpoid msymkdkipm
```

The user should be continually asked if they want to guess a letter or guess the entire sentence.

```
Do you want to 1) guess a letter or 2) guess the sentence?
```

If they choose to guess a letter, ask them for the letter they want to decode and what letter they think it actually is. Then tell them if they are right or not, redisplay the sentence, and ask them again if they want to guess a letter or the entire sentence. For example:

```
Encrypted letter? m
Actual letter? e
That's right!
yoepe cke yoe zwhcfep wv yoe pyckpoid esyekdkipe
```

If the player tries to decode the same letter twice, display an error message:

```
Encrypted letter? m
You have already decoded that letter!
yoepe cke yoe zwhcfep wv yoe pyckpoid esyekdkipe
```

If the player enters a duplicate for the actual letter (i.e. something they've already successfully decoded), just tell them they are incorrect:

```
Encrypted letter? x
Actual letter? e
Sorry, that's not right.
yoepe cke yoe zwhcfep wv yoe pyckpoid esyekdkipe
```

When the player chooses to guess the sentence, you should tell them if they are right or not (ignore capitalization). If they are right, output the number of correct and incorrect letter guesses they made to solve the puzzle. If they are incorrect, tell them to be more cautious next time. In both cases, ask if they want to quit or play another round of the game. Do not choose the same sentence from the file more than once in an execution of the game. If there are no more unused sentences left, tell the user that and quit.

```
Ok, what do you think the sentence is? These are the voyages of
the starship Enterprise
That's right! It took you 1 correct and 1 incorrect letter
guesses.
```

Rubric

Projects that don't compile will receive a zero.

[10 pts] A random sentence is chosen from the file each round.

[10 pts] The sentence is encoded using a random permutation of the alphabet.

[10 pts] Each turn allows the user to guess a letter or the whole sentence.

[10 pts] The same sentence cannot be chosen more than once in a session.

[10 pts] Correct letter guesses are recognized, and the sentence is updated accordingly.

[10 pts] Duplicate letter guesses are recognized, and an error message is displayed.

[10 pts] The program responds as described to correct and incorrect sentence guesses, including displaying the number of correct and incorrect letter guesses.

[10 pts] The program allows the user to play multiple rounds.

[20 pts] The program is clearly organized, meaningfully commented, and follows standard coding practices, including variable and class names. ***Methods are used effectively***. NOTE: programs written entirely within main will not receive full credit.