Coding Challenge notes:

Final Time and class summaries at the end

Before Starting:

Early on, my thought is to structure the code into two sections: the setup and the game itself. I obviously need a place to store the actual word, and I need to display the guesser's progress on the word as well. The easiest way to store the word and the player's progress would be two strings of equal length. One string is the word itself, and the other a string composed of only '_' characters. When the user guesses a letter correctly, the program will replace the appropriate '_' characters with the correct letter.

I will also need a way to keep track of the letters that have already been guessed, and the user would likely appreciate a list of what has already been guessed too. There are a few ways I could do this. I could pre-define a map, containing each letter as a key, and then have a boolean value stored. That way, lookup time would be minimal. But that would increase space complexity by a lot more than I need. Since I already need to display guessed letters, I might just keep adding on to a string, then use the "contains" function on a string to see if the letter has already been added to the "guessedLetters" string. If it has, I print out something that tells the user it's already been guessed, and if it hasn't, I add the letter to the "guessedLetters" string and run the rest of the process on it.

Game loop:
In the main game loop, it takes in the letter that the user guessed, and checks if the word has it. If it does, it updates the word to display to the user, and increments the correct guesses by one. If it does not, it would mark one more incorrect letter and increment the incorrect guesses by one.

Words by number of letters:
3: map
4: nail
5: bench
6: knight
7: digging
8: absolute
9: infection
10: friendship
11: challenging
12: fabrications

After an hour and 30 minutes: I have the basic structure of the initialization and game loop code put down. What I outlined above worked pretty well, although there were a few structural aspects I did not mention there. I made three classes: one for the main function, one called "GameRound" that represents each round of the game, and one called "WordMap" that stores the list of words, and randomly picks one when Main wants to start a new game.

I have tested a few of the core functions, such as the initialization function for the gameplay loop, and the toString function of the gameplay loop, which will be used for player information output.

Two hours and 20 minutes in: I have a working loop that takes a character as an input, and ensures that input is valid. I have an “exit” command working, both so a player can exit without needing to terminate the program, and so I can test things easier without needing to terminate the program myself. Theoretically the function to parse a guess should work, but that is the next step of testing.

Two hours and 55 minutes in: I have successfully got the game loop working, with only a few bugs to work out. I guess letters, it accurately plugs them into the game and fills in the word when they are right, and adds them to a list of characters when they are wrong. However, I need to check to see if the letter has already been guessed, since the program does not handle that right now. Currently it simply adds a previously guessed letter to the end of the list of incorrect guesses, even if they are in the word, or already in the list of incorrect guesses. The program also currently handles bad outputs that are more than one character long, but it would not handle a guess that is not a letter (such as a number or a punctuation mark). Those are the two things I will be working on next.

Three hours and 25 minutes: I am finished! The bugs from before are handled. The program only takes two types of inputs: single letters, or the word “exit” (which terminates the program). That is until you finish a game, where you are prompted to either say “yes” to start a new game, or “no” (or “exit”) to terminate the program. My testing has not revealed any issues with the program, so I’m going to take all of the files and send the program over to GitHub!

Final Time: 3 hours and 30 minutes

Classes:
Main: This class runs the actual loop logic of the game, and initializes the game. It chooses a word from the WordMap class. Then it loops over input and parses whether that input is valid or not. It does not check for repeat characters though. Once it has determined that the input is valid, it passes it into the GameRound class.

WordMap: This class has a map of ten random words of different lengths. It tracks which words have been picked, and will randomly choose from the list of words that have not been picked yet. Once all ten words have been picked, it will reset it’s count and pick from the original list again.

GameRound: This class parses guesses, and then figures out if those guesses are correct or not. It stores the correct word, and two stringbuilders that track correct and incorrect guesses. This class also has the logic to format the output after each guess.

Thank you for reviewing my activity, I look forward to hearing back from you soon!