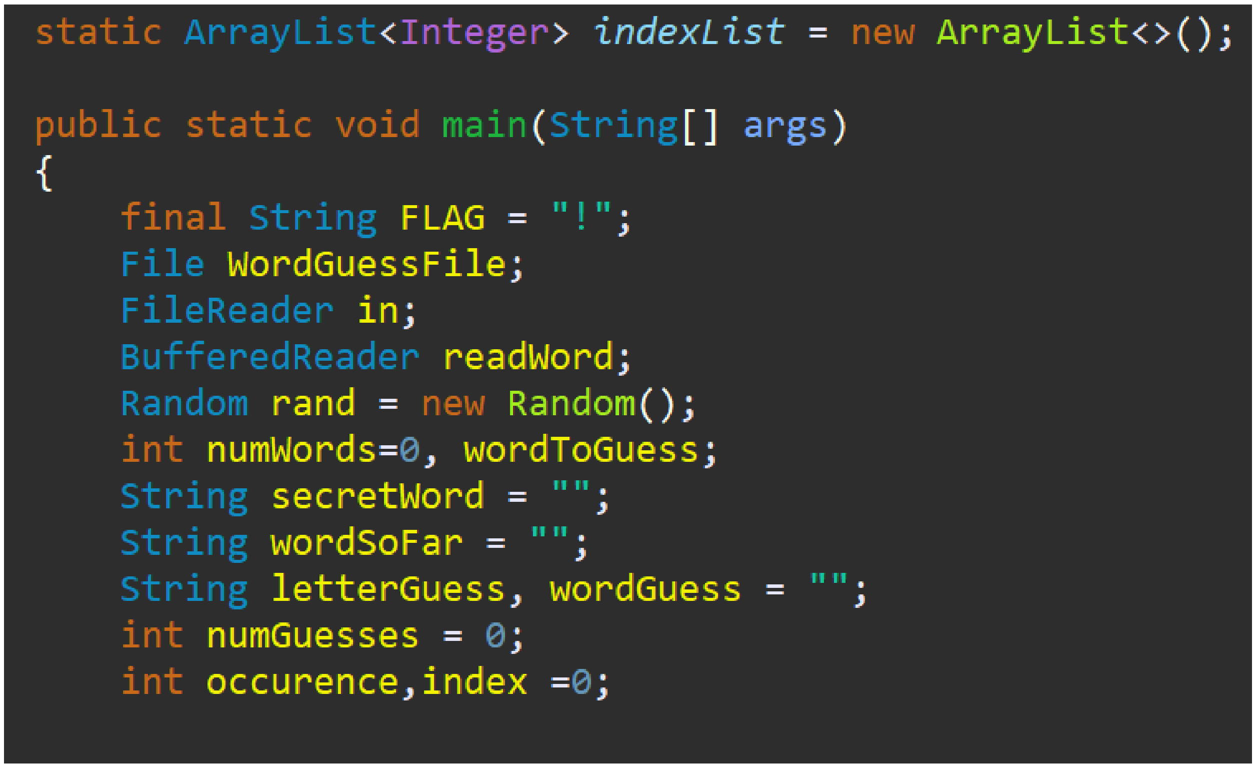
Credit Name: CSE 2130- File Structure and Exception Handling

Assignment: wordGuess

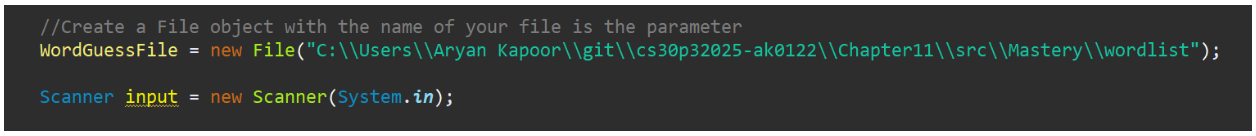
How has your program changed from planning to coding to now? Please Explain



```
static ArrayList<Integer> indexList = new ArrayList<>();

public static void main(String[] args)
{
    final String FLAG = "!";
    File WordGuessFile;
    FileReader in;
    BufferedReader readWord;
    Random rand = new Random();
    int numWords=0, wordToGuess;
    String secretWord = "";
    String wordSoFar = "";
    String letterGuess, wordGuess = "";
    int numGuesses = 0;
    int occurence,index =0;
```

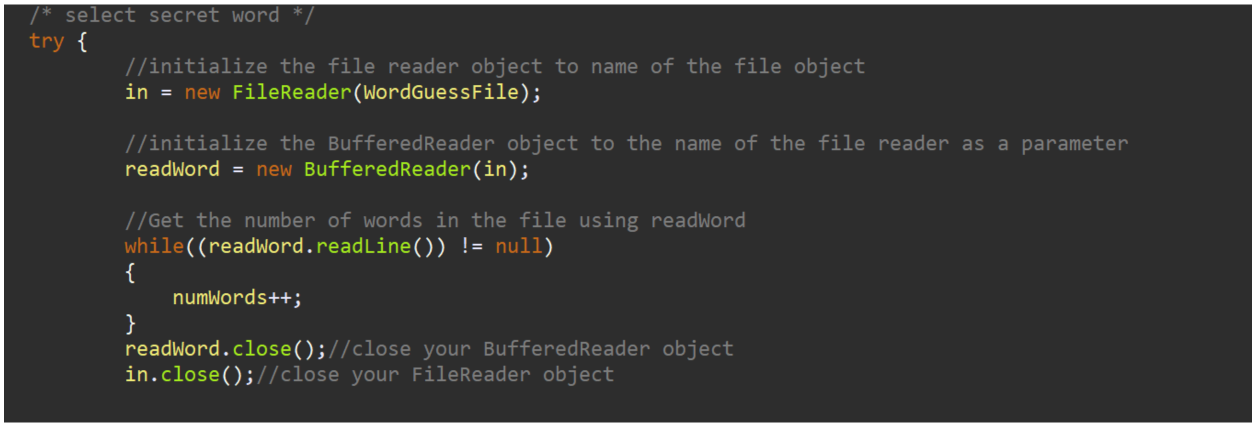
If first declared all my variables and objects:

1. ***The '!' is declared as final because we will use it as the indicator for when the user wants to guess the entire word, so we dont want it to be modified.***
2. ***The wordGuessFile is the object which we will use to link our text file, that has assorted words, one of which will be selected at random to be our chosen word.***
3. ***We have a file reader and buffered reader to read through our file to choose a word.***
4. ***A random variable rand to randomize our word selection.***
5. ***A series of int variables to: count total words in file, a selection number for our chosen word\, occurrences of guesses, indexes of a specific letter guess in the secret word and the total number of guesses.***
6. ***String variables for the secret word and the word so far as the player guesses it.***
7. ***An integer array list to hold index values for a specific letter guess.***



```
//Create a File object with the name of your file is the parameter
WordGuessFile = new File("C:\\Users\\Aryan Kapoor\\git\\cs30p32025-ak0122\\Chapter11\\src\\Mastery\\wordlist");

Scanner input = new Scanner(System.in);
```

First we link the file from which we get the words to the code
We use scanner to get user inputs

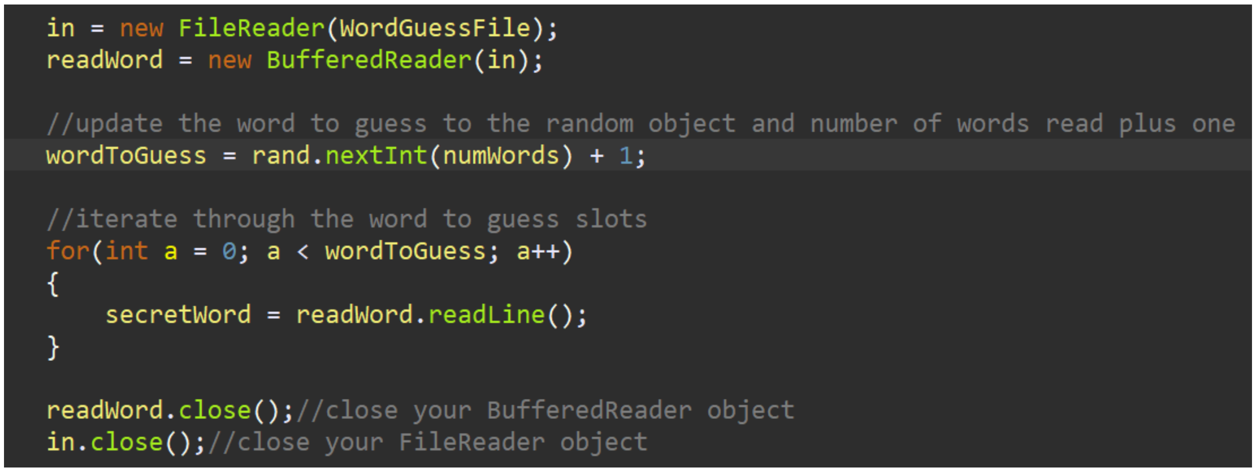


```
/* select secret word */
try {
        //initialize the file reader object to name of the file object
        in = new FileReader(WordGuessFile);

        //initialize the BufferedReader object to the name of the file reader as a parameter
        readWord = new BufferedReader(in);

        //Get the number of words in the file using readWord
        while((readWord.readLine()) != null)
        {
            numWords++;
        }
        readWord.close();//close your BufferedReader object
        in.close();//close your FileReader object
```

We need to first select a word from the list, so we set up a bufferedReader to read strings from the file.

We get the total number of words in the list by using a while loop until the readLine() method returns a null statement. While this loop runs we simply add 1 to a counter for each iteration of the loop. (This works under the assumption that we have only 1 word per line)

Then we close the bufferedReader and fileReader.

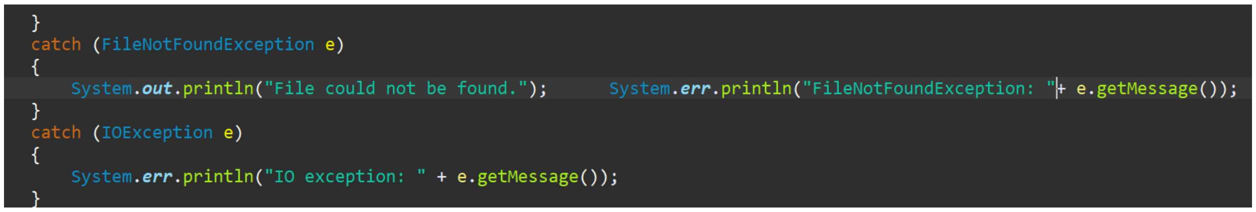


```
in = new FileReader(WordGuessFile);
readWord = new BufferedReader(in);

//update the word to guess to the random object and number of words read plus one
wordToGuess = rand.nextInt(numWords) + 1;

//iterate through the word to guess slots
for(int a = 0; a < wordToGuess; a++)
{
    secretWord = readWord.readLine();
}

readWord.close();//close your BufferedReader object
in.close();//close your FileReader object
```

We set up a new bufferedReader, this time to read through the file until we reach a random, but specified point which is will be our random word. To do this we generate a random number between 1 and the # of words in the list.

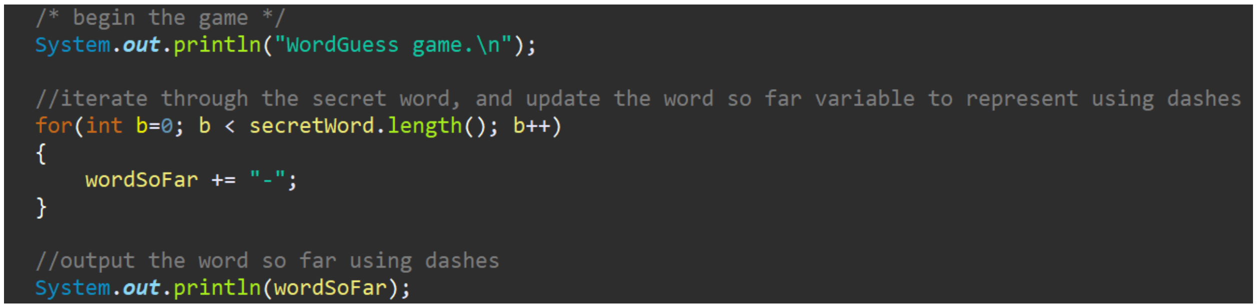
We then set up a for loop with a parameter running from 0 until the index 1 before this random number. Thus the random word is the word corresponding to whichever random line the for loop ends at.

Again, we close the bufferedReader and fileReader.



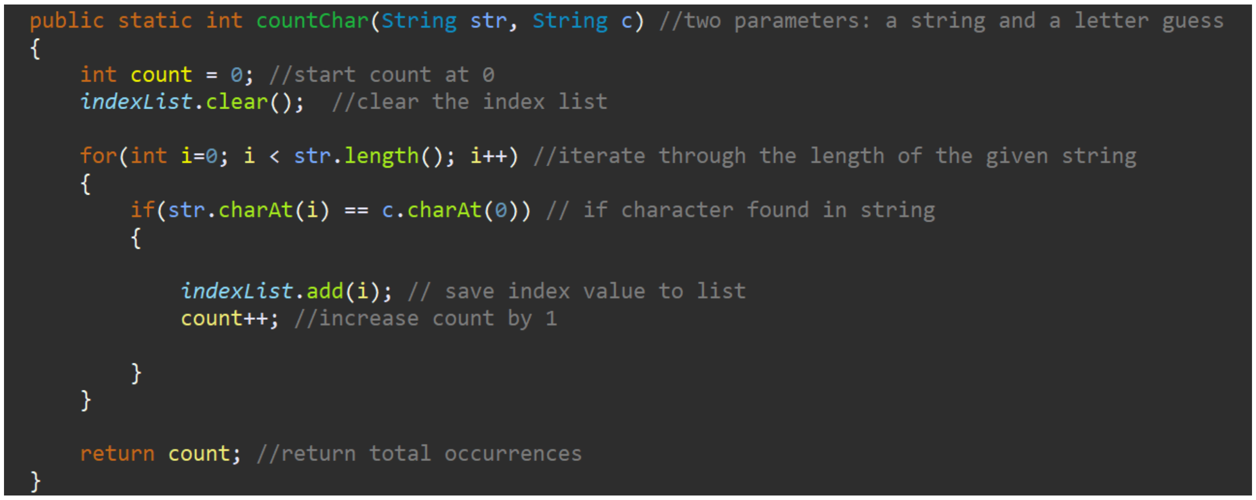
```java
}
catch (FileNotFoundException e)
{
    System.out.println("File could not be found.");      System.err.println("FileNotFoundException: "+ e.getMessage());
}
catch (IOException e)
{
    System.err.println("IO exception: " + e.getMessage());
}
```

Since we open the file readers we encased it in a try loop so we also need to catch for common exceptions.



```java
/* begin the game */
System.out.println("WordGuess game.\n");

//iterate through the secret word, and update the word so far variable to represent using dashes
for(int b=0; b < secretWord.length(); b++)
{
    wordSoFar += "-";
}

//output the word so far using dashes
System.out.println(wordSoFar);
```

Now we focus on coding the mechanics behind this game.
First we set up the wordSoFar(displayed word) to be as many dashes as there are characters in the secret word and print the result.

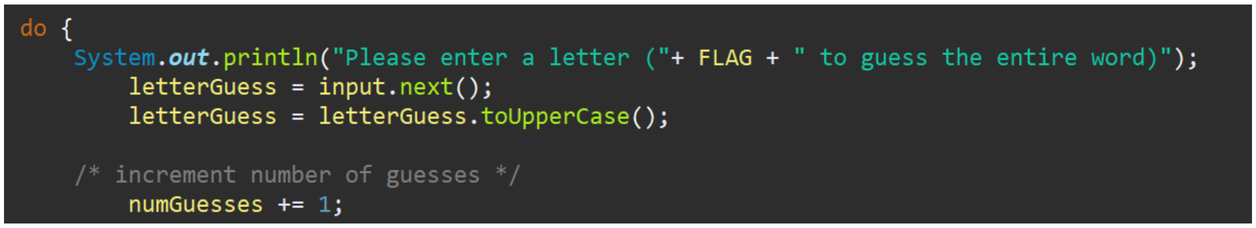


```java
public static int countChar(String str, String c) //two parameters: a string and a letter guess
{
    int count = 0; //start count at 0
    indexList.clear();  //clear the index list

    for(int i=0; i < str.length(); i++) //iterate through the length of the given string
    {
        if(str.charAt(i) == c.charAt(0)) // if character found in string
        {

            indexList.add(i); // save index value to list
            count++; //increase count by 1

        }
    }

    return count; //return total occurrences
}
```

Additionally I created a new method which helps when a player guesses a character with multiple occurrences in the secret word.
It takes in 2 parameters, a string and a character (defined as a string for ease of comparison in this method and for consistency with the String variable letterGuess)

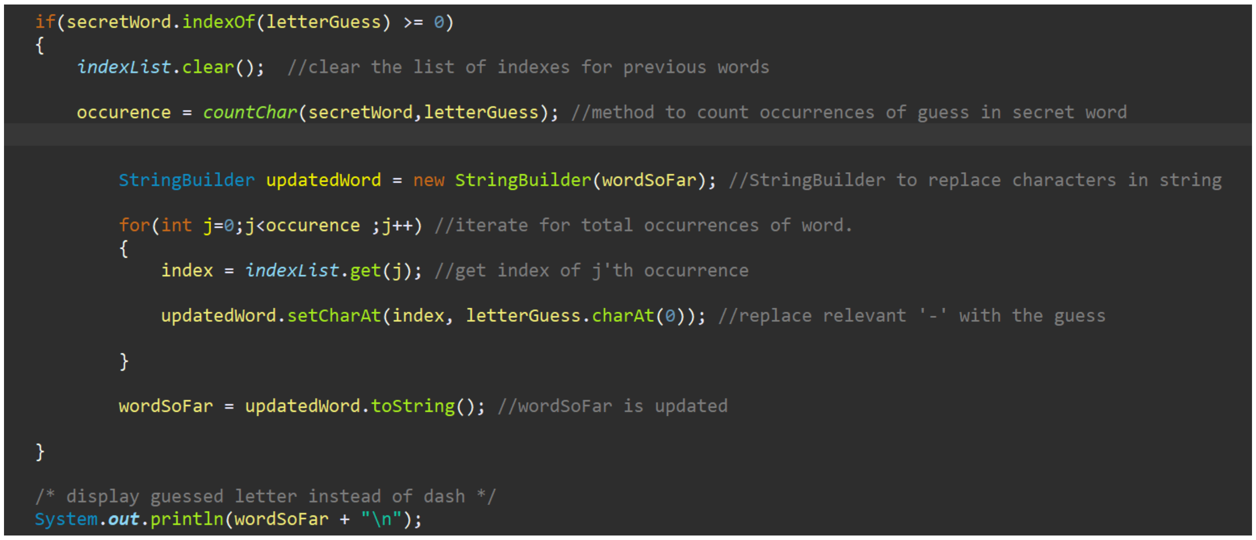
We search for the occurrences of the character c in the string str:
We set up for loop which runs for the length of the string as we want to check all characters in the string. We have an if statement which adds 1 to a counter of occurrences as well as saving the index where the i'th occurrence of the guess is in the secret word.

At the end we return the total number of occurrences.



```
do {
    System.out.println("Please enter a letter ("+ FLAG + " to guess the entire word)");
        letterGuess = input.next();
        letterGuess = letterGuess.toUpperCase();

    /* increment number of guesses */
        numGuesses += 1;
```

Now back in the main method, we do a do-while loop to allow the user to continue guessing until they choose the guess the word or the guess all letters without guessing the word.
In the loop we first ask the user to input their guess and then we convert their response to uppercase and increase the number of guesses counter by 1.



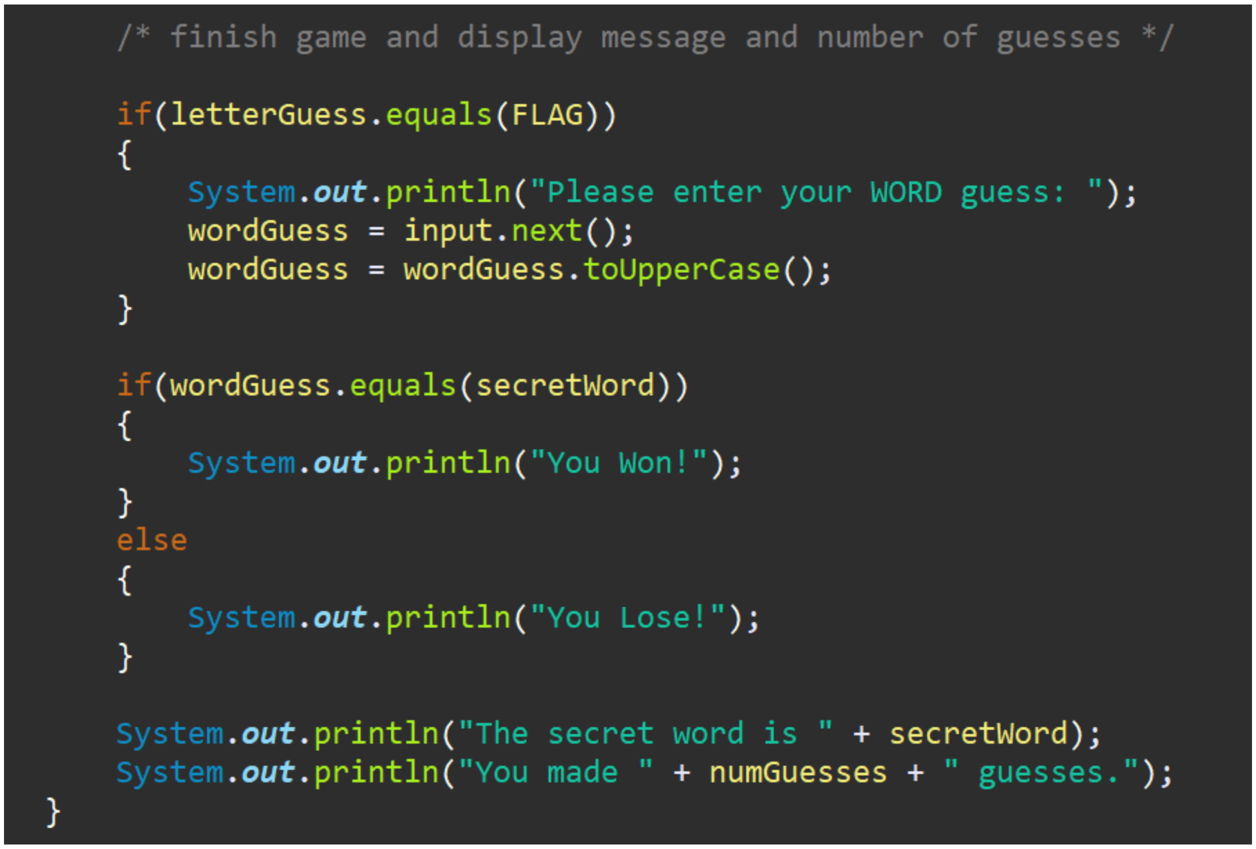
```
if(secretWord.indexOf(letterGuess) >= 0)
{
    indexList.clear();  //clear the list of indexes for previous words

    occurence = countChar(secretWord,letterGuess); //method to count occurrences of guess in secret word


        StringBuilder updatedWord = new StringBuilder(wordSoFar); //StringBuilder to replace characters in string

        for(int j=0;j<occurence ;j++) //iterate for total occurrences of word.
        {
            index = indexList.get(j); //get index of j'th occurrence

            updatedWord.setCharAt(index, letterGuess.charAt(0)); //replace relevant '-' with the guess

        }

        wordSoFar = updatedWord.toString(); //wordSoFar is updated

}

/* display guessed letter instead of dash */
System.out.println(wordSoFar + "\n");
```

If they guess incorrectly the indexOf method will return -1 but if they guess correctly, the index will be greater than or equal to 0. Thus we only consider the case in which they guess correctly in the if statement.
We clear the index lists of any entries for previous letter guesses and the get the number of occurrences of the secret word and their position in the secret word.

We set up a Stringbuilder to be able to replace characters in a string easier and more efficiently than using substrings.

In a for loop which runs for the number of occurrences of the guess, we get the index from the list and then replace the relevant dash in wordSoFar with the letterGuess.

This repeats for all the occurrences in the word and then the wordSoFar is updated to the updatedWord and is then printed for the player to see.



```
        /* finish game and display message and number of guesses */

        if(letterGuess.equals(FLAG))
        {
            System.out.println("Please enter your WORD guess: ");
            wordGuess = input.next();
            wordGuess = wordGuess.toUpperCase();
        }

        if(wordGuess.equals(secretWord))
        {
            System.out.println("You Won!");
        }
        else
        {
            System.out.println("You Lose!");
        }

        System.out.println("The secret word is " + secretWord);
        System.out.println("You made " + numGuesses + " guesses.");
    }
```

Lastly, we handle the end of the game. If the letter guess is an !, then we prompt them to enter the word guess. Again to ignore the case, we convert their guess to uppercase and then have an if else statement where they win if their wordGuess is the same as the secretWord or they lose otherwise. Finally, the secret word is revealed and the number of guesses is shown.