WordleGame class: This class would manage the game mechanics and keep track of the player's guesses. It would have the following methods:

makeGuess(Guess guess): Takes in a Guess from user, verifies correct and incorrect postitions

getGuessCount(): Returns the number of guesses the player has made so far.

isGameOver(): Returns true if the player has either solved the puzzle or run out of guesses.

Guess class: This class would represent a single guess made by the player. It would have the following attributes:

guessText: A string representing the player's guess.

feedback: A string representing the result of the guess.

WordleUI class: This class would create the UI and update it with the current game state.

initialize(): Initializes the UI by making the wordle board.

updateGameDisplay(): Updates the display with the current state of the game (the player's guesses and the feedback for the guesses).

displayGameOver(): Displays the final result of the game (whether the player won or lost).

WordleController class: This class would handle user input and update the game

handleGuess(): Called when the player submits a guess.

Gets the text from the input field and creates a Guess object, which it passes to the WordleGame class's makeGuess() method. Then calls updateGameDisplay() to update the UI with the results

handleNewGame(): Called when the player clicks the "New Game" button. Resets the game, starting all data over

WordGenerator class: This class would be responsible for generating a random word for the player to guess. It would have the following method:

generateWord(): Generates a random five-letter word for the player to guess. This method could use an array of five-letter words as a data source.

We will probably use a file containing all words that are 5 letters long and randomly select one.