Helpful hint:

The two classes used in this project (Game and Phrase) interact with one another. That means when you build this project you'll be moving between the app.js, Game.js and the Phrase.js file instead of building out each file independently. This means you might have to build a little bit more than you're used to before you can test something. In the instructions you'll find notes that indicate good break points to try and test out your code.

Step 1

In the project files you downloaded, you'll see a directory called 'js' that includes the following files:

- app.js to perform basic DOM selection, add event handlers, and to reset the game when it ends
- Phrase.js to create a Phrase class to manage individual phrases
- Game.js to create a Game class to manage the functioning of the game: it has methods for starting and ending the game, handling interactions, getting random phrases, checking for a win, and removing a life counter.

Step 2

Declare your classes.

- Inside the Game.js file, declare the Game class.
- Inside the Phrase.js file, declare the Phrase class.

Step 3

Create a constructor method inside each class.

- The Game class constructor method doesn't receive any parameters. The Game class has the following properties:
  - missed: used to track the number of missed guesses by the player. The initial value is 0, since no one has made a guess yet.
  - phrases: an array of phrases to use with the game (you'll use a method to create new instances of the Phrase class). A phrase should only include letters and spaces — no numbers, punctuation or other special characters. The initial value will be a call to the Game class's `createPhrases()` method that you'll build in the next step.

- activePhrase: this is the Phrase object that's currently in play. The initial value is `null`.

- The Phrase class constructor method should receive one parameter: phrase. The phrase class has the following properties:
  - phrase: This is the actual phrase the Phrase object is represented. This property should convert the phrase to all lower case.
  - letterCount: This is number of letters inside the phrase (excluding spaces, punctuation, etc.)

Step 4

I am stuck on this part

Inside the Game class, create a method called `createPhrases()`. This method should create and then return an array with 5 new Phrase objects. These are the Phrase objects that the game will randomly choose from when showing a new phrase to the player.

When creating a Phrase object, don't forget to pass in the actual string phrase that the Phrase object is representing.

Here's the documentation for this method:

```
/**
 * Creates phrases for use in game
 * @return      {array}         An array of phrases that could be used in the game
 */
```

Step 5

Let's write the getRandomPhrase() method mentioned in the last step. This method goes inside the Game class in the Game.js file. This should method should select and then return a random phrase from the array of phrases stored in the Game's `phrases` property.

Here's the documentation for this method:

```
/**
 * Selects random phrase from phrases property
 * @return      {Object}        Phrase object chosen to be used
 */
```

Step 6

Switch gears for a moment and head to the Phrase class inside Phrase.js. Inside the Phrase class, create a method called addPhraseToDisplay(). This method adds letter placeholders to the display when the game starts. Each letter is presented by an empty box, one list item for each letter. See the example_phrase_html.txt file for an example of what the render HTML for a phrase should look like when the game starts. When the player correctly guesses a letter, the empty box is replaced with a the matched letter (see the showMatchedLetter() method below. Make sure the phrase displayed on the screen doesn't include spaces.

Here's the documentation for this method:

I am stuck on showMatchedLetter()

```
/**
 * Display phrase on game board
 */
```

Step 7

Head back to Game.js. Now that we can select a random phrase and display the placeholders for our selected phrase on the screen, let's create a `startGame()` method in the Game class that will put this all in motion for us. This method uses the `getRandomPhrase()` method to select a Phrase object from the Game object's array of phrases, and then calls the `addPhraseToDisplay()` method (which is a method on the Phrase class) on the selected Phrase object.

Here's the documentation for this method:

```
/**
 * Begins game by selecting a random phrase and displaying it to user
 */
```

Step 8

I think I did these steps, just in a different way

Now that you've built the basics, head over to the app.js file. This is where you'll create an event listener for the Start Game button that the user sees when they load your PhraseHunter game.

- Inside app.js, initialize a new `var` variable called `game` that's not set to anything.
- Then, add a click event listener to the HTML element `btn__reset`. Inside the callback function for this click event listener, use your `game` variable to initialize a new Game object. Call the `startGame()` method on this new Game object. Inside this callback function you should also hide the start overlay so the user can see the game instead of just the game title and start button.