Node Hangman

With a focus on OOP

Objects in Solution

For each Object in your solution, establish:

- Concept:
  - What data does this object maintain?
  - What actions does this object need to perform?
- Pseudocode
  - What parameters are needed at Object construction?
  - What properties are needed to store this Object's data?
  - What Getters/Setters are needed?
  - What functions are needed to perform this Object's actions?
- Javascript
  - The fun part!

Letter

Concept:

- A Letter Object determines what text to display for the Hangman game, either itself or an underscore (“_”).

Letter

Pseudocode:

- Any parameters for construction?
  -
- Any properties?
  -
  -

Letter

Pseudocode:

- Any parameters for construction?
  - a letter string
- Any properties?
  - Boolean variable to track if this Object is guessed or not.
  - String variable to track the Object's "real" value (the secret letter)

Letter

Pseudocode:

- Any Getters & Setters?
  -
  -
  -
- Additional functions?
  -

Letter

Pseudocode:

- Any Getters & Setters?
  - Getter function for the Boolean
  - Getter function for the String secret value
  - Getter function for the display text (returns the secret value or underscore)
- Additional functions?
  - Check if a guess matches the secret value

Letter

- Parameter
- Properties
- Getter functions
- Actions:
  - checkMatch()

```
1  Letter = function(lettertext) {
2      var guessed = false;
3      var trueValue = lettertext;
4
5      this.isGuessed = function() {
6          return guessed;
7      }
8      this.getDisplayText = function() {
9          if(guessed) {
10             return trueValue;
11         }
12         else {
13             return "_";
14         }
15     }
16     this.checkMatch = function(guess) {
17         if(trueValue == guess) {
18             guessed = true;
19             return true;
20         }
21         return false;
22     }
23 }
24 module.exports = Letter;
```

Word

Concept:

- From the homework description, a Word Object “should contain all of the methods which will check the letters guessed versus the random word selected”.
- The main idea is that the Word Object can handle some of the guessing functionality.

Word

Pseudocode:

- Any parameters for construction?
  -
  -
- Any properties?
  -
  -

Word

Pseudocode:

- Any parameters for construction?
  - Letter Object definition required
  - a word string (selected randomly)
- Any properties?
  - Array of Letter Objects
  - String variable to track the word parameter

Word

Pseudocode:

- Any Getters & Setters?
  -
  -
- Additional functions?
  -

Word

Pseudocode:

- Any Getters & Setters?
  - Getter function for display text of the entire word
  - Getter function for the String secret value
- Additional functions?
  - Function to check if a guess matches any of this Object’s Letters

Word

```
1  var Letter = require("./letter.js");
2  Word = function (word) {
3      var actualValue = word.toUpperCase();
4      var letterList = [];
5
6      for(var i = 0; i < actualValue.length; i++) {
7          letterList.push(new Letter(actualValue[i]));
8      }
9      this.getActualValue = function() {
10         return actualValue;
11     }
12
```

- Letter Object definition
- Parameter
  - a word string (selected randomly)
- Properties
  - Array of Letter Objects
  - String variable to track the word parameter
- Getter Function
  - For the Word's secret value, no underscores

Word

```
13      this.getDisplayText = function() {
14          var wordDisplayText = "";
15          for(var i = 0; i < letterList.length; i++) {
16              var singleLetter = letterList[i];
17              wordDisplayText += singleLetter.getDisplayText() + " ";
18          }
19          return wordDisplayText;
20      }
21
```

For each Letter object

Get the Letter Object's display text

and concatenate the display text to a longer string, along with a space character.

Return the concatenated string.

Word

```
22      this.isGuessed = function() {
23          for (var i = 0; i < letterList.length; i++) {
24              var singleLetter = letterList[i];
25              if(singleLetter.isGuessed() == false) {
26                  return false;
27              }
28          }
29          return true;
30      }
31
```

For each Letter object

Check if the Letter Object has been guessed

If any Letter Object has not yet been guessed, we know the Word Object still is not guessed.

Else, if all Letters are guessed, the Word Object has been guessed.

Word

```
32      this.checkMatches = function(guess) {
33          var matchFound = false;
34          for(var i = 0; i < letterList.length; i++) {
35              var singleLetter = letterList[i];
36              if(singleLetter.checkMatch(guess)) {
37                  matchFound = true;
38              }
39          }
40          return matchFound;
41      }
```

Given a user guess,
Loop through each Letter Object:
    Check if the Letter Object matches the guess
    Record that at least one matching guess was found.
After looping, return whether any matching guesses were found
(true for > 1 matches, false for 0 matches)

Game

Concept:

- Randomly selects a word either from an array of words or external word bank.
- Tracks whether a particular guess has already been made in a previous round.
- Keeps track how many lives/chances left to guess.
- Can indicate whether game is won.

Game

Pseudocode:

- Any parameters for construction?
  -
  -
- Any properties?
  -
  -
  -

Game

Pseudocode:

- Any parameters for construction?
  - Word Object definition is required
  - word bank (can also be property)
- Any properties?
  - Randomly Selected Word Object
  - Array of previous guesses
  - Number of lives / tries remaining

Game

Pseudocode:

- Any Getters & Setters?
  -
  -
  -
- Additional functions?
  -
  -
  -

Game

Pseudocode:

- Any Getters & Setters?
  - A getter for the Game's display text
  - A getter for number of lives / tries remaining
  - A getter for the hidden word itself (for when game is over)
- Additional functions?
  - Function to determine if guess has already been made in a previous round
  - Function to check if a guess is in the random word.
  - Function to determine if user has won the game

Game

```
1  var Word = require("./word.js");
2  var wordList = require("./wordbank.js");
3
4  Game = function () {
5      var lettersUsed = [];//stores previous guesses
6      var triesLeft = 10;//attempts remaining
7
8      var randomIndex = Math.floor(Math.random() * (wordList.length - 0)) + 0;
9      var randomWord = wordList[randomIndex];
10     var secretWord = new Word(randomWord);
11
```

- Word Object Definition
- External Word Bank
- Properties
    - Array of previous guesses
    - Number of lives / tries remaining
    - Randomly Selected Word Object

Game

```
12      this.getTriesLeft = function() {
13          return triesLeft;
14      };
15
16      this.getActualValue = function() {
17          return secretWord.getActualValue();
18      }
19
20      this.getDisplayText = function() {
21         return secretWord.getDisplayText();
22      };
23
```

- Getters
  - A getter for the hidden word itself (for when game is over)
  - A getter for number of lives / tries remaining
  - A getter for the Game's display text

Game

```
24      this.isNewGuess = function(guess) {
25          guess = guess.toUpperCase().trim();
26          if (lettersUsed.indexOf(guess) === -1)
27              return true;
28          else
29              return false;
30      };
```

- Function to determine if guess has already been made in a previous round

Game

```
32      this.makeGuess = function(guess) {
33          guess = guess.toUpperCase().trim();
34
35          lettersUsed.push(guess);
36
37          //if the guess matched at least one of the word's letters
38          if (secretWord.checkMatches(guess)) {
39              return true;
40          }
41          //else, none of the word's letters matched the guess
42          else {
43              //Remove life
44              triesLeft--;
45              return false;
46          }
47      };
```

- Function to determine if guess has already been made in a previous round

Game

```
49      this.isWon = function() {
50          return secretWord.isGuessed();
51      };
```

- Function to determine if user has won the game

Main

Concept:

- Running it in Terminal/Bash will start the game.
- The app should end when a player guesses the correct word or runs out of guesses.

Main

Pseudocode:

- Any parameters for construction?
  -
  -
- Any properties?

Main

Pseudocode:

- Any parameters for construction?
  - Game Object Definition
  - Inquirer package
- Any properties?
  - None (for a simple implementation)

Main

Pseudocode:

- Any Getters & Setters?
  -
- Additional functions?
  -

Main

Pseudocode:

- Any Getters & Setters?
  - Not applicable
- Additional functions?
  - Recursive function to iterate through rounds of a game of hangman:

Main

Pseudocode:

- Recursive function to iterate through rounds of a game of hangman:

Function:

1. Display word blanks
2. Get guess with inquirer prompt (validate input & check if new guess)
3. Pass guess to Hangman Game Object, and have it check if guess was good.
    a. If guess was good, ask Game Object if user wins
        i. If game is won, ask Game Object to display the full word, and exit.
        ii. Else, go to next round
    b. If guess was bad, ask Game Object how many guesses left
        i. If user has no guesses left, ask Game Object to display the word, and exit.
        ii. Else, go to next round

Main

```
1  var inquirer = require('inquirer');
2  var Game = require('./Game.js');
3  var hangman = new Game();
4  gameRound();
5
```

- Game Object Definition
- Inquirer package

Main

```
6  function gameRound() {
7      console.log(hangman.getDisplayText());
8
9      inquirer.prompt([
10         {
11             name: "letterGuess",
12             message: "Guess a letter:",
13             type: "input",
14             validate: function(value) {
15                 if (value.length !== 1) {
16                     console.log("\nPlease enter a single letter as your guess.");
17                     return false;
18                 }
19                 else if (!/[a-zA-Z]/.test(value)) {
20                     console.log("\nPlease enter a letter from the alphabet as your guess.");
21                     return false;
22                 }
23                 else if(!hangman.isNewGuess(value)) {
24                     console.log("\nYou have already guessed " + value
25                         + "\nPlease guess a different letter.");
26                     return false;
27                 }
28                 return true;
29             }
30         }
```

Pseudocode:

Function:

1. Display word blanks
2. Get guess with inquirer prompt (validate input & check if new guess)

- Recursive function to iterate through rounds of a game of hangman

Main

```
31      ]).then(function (answer){
32
33              if(hangman.makeGuess(answer.letterGuess)) {
34                  //guess was good, so check if user won
35                  if(hangman.isWon()) {
36                      console.log(hangman.getActualValue());
37                      console.log("you win");
38                  }
39                  //else, next round
40                  else {
41                      gameRound();
42                  }
43              }
44              else {
45                 //guess was bad, so check if user lost
46                 if(hangman.getTriesLeft() == 0) {
47                      console.log(hangman.getActualValue());
48                      console.log("you lost");
49                 }
50                 //else, next round
51                 else {
52                      console.log("Tries Left: " + hangman.getTriesLeft());
53                      gameRound();
54                 }
55              }
56      });
```

Pseudocode:

Function:

1. Display word blanks
2. Get guess with inquirer prompt (validate input & check if new guess)
3. **Pass guess to Hangman Game Object, and have it check if guess was good.**
   a. **If guess was good, ask Game Object if user wins**
      i. **If game is won, ask Game Object to display the full word, and exit.**
      ii. **Else, go to next round**
   b. **If guess was bad, ask Game Object how many guesses left**
      i. **If user has no guesses left, ask Game Object to display the word, and exit.**
      ii. **Else, go to next round**

- Recursive function to iterate through rounds of a game of hangman

Demo