Hints: Playing A Placed Token

There is a lot of code that you can write in this section. Here are some hints to help you in case you get stuck on any of the steps.

Implementing a "Column" class

Because the `Column` class *encapsulates* everything it means to be a column, you can implement it however you want. Here are two different ways to do it. It is important to note that *neither is the correct way to implement them*. They are both valid ways to write the class. This is something that makes software programming a treat is that you get to choose how you'd like to do things like this to match the way that *you* think about the problem.

A column with an array that grows as you need it

- Create and export a class named `Column`.
- Create a constructor for `Column` that takes no parameters, creates an instance variable named `tokens`, and sets it equal to an empty array.
- Create a method named `add` that takes a player number and uses the `push` method of the array in `tokens` to add it to the end of the array. The array will only have the number of tokens in it that have been "dropped" into the column by players. This means that the first token is in position `5`, the second token is in position `4`, and so on.
- Create a method named `getTokenAt` that takes a row number. Use the number passed into it to map to the correct space. The mapping goes like this:

Input row number	Maps to this array index
0	5
1	4
2	3
3	2
4	1
5	0

The trade off, here, is you get to have a simple `add` method that uses `push` to add a new token value onto the array. The complexity comes with having to figure out what token is at a specific index.

A column with an array of size six

- Create and export a class named `Column`.
- Create a constructor for `Column` that takes no parameters, creates an instance variable named `tokens`, and sets it equal to an array with six `null` values in it.
- Create a method named `getTokenAt` that takes a row position number. Return the value of the `tokens` array at that index.
- Create a method named `add` that takes a player number. Loop through the array from the end (index 5) to the beginning (index 0). In the first spot that has a `null` value, put the player number there and break out of the loop.

Here, you trade off a simple `getTokenAt` method that just returns a value for a more complex `add` method that has to find a `null` value in which to put the token.