Unit 6: OOP & TypeScript – Lab 2: Greetings

Assigned: 12/12/22

Due: 12/15/22

Completed: TBD

LMS: <https://lms.grandcircus.co/mod/assign/view.php?id=22793>

Google Doc: <https://docs.google.com/document/d/1gDCBhE5H2t1Uan9NfpQNIrwqYgfcu2ZKp6STR_bGhi8/preview>

GitHub:

**Overview:** Using TypeScript, create a Greeter class and multiple subclasses that each have a different way of greeting someone.

**Build Specifications**

* You will create a total of 4 greeter classes (1 parent and 3 children).
* Create each class in its own file.
* Each class must have at least two Jest test cases to ensure that its functionality is working correctly. Some classes will need more tests to cover all the features.
* In index.ts, import each of the 4 greeter classes and use them to console.log the greeting produced by each class, including the Greeter parent class.

**Greeter Class**

In its own file, create a class named Greeter. This is the parent class.

* Properties:
  + greeting (a string)
* Methods:
  + constructor: Sets the greeting property from a parameter.
  + greet: Has a string parameter called name. It returns the composed greeting based on the greeting property and name parameter. Use this format: "{greeting}, {name}!".
* Jest Tests: Test with different greetings and names.

**JavaScriptGreeter Class**

In its own file, create a class named JavaScriptGreeter. This is a subclass of Greeter.

* Additional Properties: n/a
* Additional Methods: n/a
* Override greet: Override the greet method to wrap the result in a console.log. For example, greet might return "console.log('Hello, Grant!')". Note, this is not actually calling console.log, just returning a string that looks like JavaScript.
* Jest Tests: Test with different greetings and names.

**LoudGreeter Class**

In its own file, create a class named LoudGreeter. This is a subclass of Greeter.

* Additional Properties:
  + extra (a string): Use the private modifier. This property keeps track of the extra exclamation points. Hard code an initial value.
* Additional Methods:
  + addVolume: No parameters. Returns void. Every time this is called, it adds an additional exclamation point to extra.
* Override greet: Override the greet method to add the extra exclamation points to the end. For example, greet might return "Hello, Grant!!!". Note, the original greet method already has one exclamation point, so LoudGreeter starts with two exclamation points even before calling addVolume.
* Jest Tests: Test with and without calling addVolume. Test calling addVolume different numbers of times.

**HtmlGreeter Class**

In its own file, create a class named HtmlGreeter. This is a subclass of Greeter.

* Additional Properties:
  + tagName (a string)
* Constructor Parameters:
  + greeting (a string): sets greeting on the superclass.
  + tagName (a string): sets the tagName property. This parameter is optional and has a default value of "h1".
* Override greet: Override the greet method to wrap the result in the specified HTML tag. For example, greet might return "<h1>Hello, Grant!</h1>".
* Jest Tests: Test with different greetings, names, and tagNames. Also test the default "h1" tagName when the second constructor argument is left off.

**Extended Challenges**

* Install and use the [promptly](https://www.npmjs.com/package/promptly) library in index.ts to prompt the user for their name and then greet them with all the different greeters. (Remember when using libraries with TypeScript also install their types: npm install @types/promptly)
* Install the [chalk](https://www.npmjs.com/package/chalk) library. Create an additional ChalkGreeter that uses chalk to format the output. Use a property on your class to pick the color or other formatting.

**Grading Criteria**

1. Greeter class works as expected.
2. Greeter class has two unique and meaningful test cases.
3. JavaScriptGreeter class works as expected.
4. JavaScriptGreeter is a subclass of Greeter and is in its own file.
5. JavaScriptGreeter class has two unique and meaningful test cases.
6. LoudGreeter is a subclass of Greeter and adds an exclamation point to greet().
7. LoudGreeter adds additional exclamation points for each call to addVolume.
8. LoudGreeter class has two unique and meaningful test cases, including one that calls addVolume and one that does not.
9. HtmlGreeter is a subclass of Greeter in its own file and works as expected when a tagName is specified.
10. HtmlGreeter defaults to h1 when no tagName is specified.
11. HtmlGreeter class has two unique and meaningful test cases.

Graphical user interface, text, application, chat or text message

Description automatically generated