Cairo Crash Course

Functions Visibility - Exercise 2

In this exercise, we'll test a combination of internal and public functions in our contract.

Contract Implementation

In the `visibility_exercise_2.cairo` file, follow these steps:

1. Define the contract interface `IFunctionVisibility` with 2 functions
    - `set_value` - Receives the contract state as reference, and a `message: u32`
    - `read_value` - Receives the contract state as snapshot and returns `u32`
2. Define the contract `FunctionVisibility` and it's storage with one `u32` called `value`
3. Create an impl block for internal functions that are not accessible in the contract ABI, and implement the following function:
    - `_get_value` will read the `value` from `Storage` and return it
4. Create an impl block for public functions that will be accessible in the contract ABI, and implement the following functions:
    - `set_value` - writes the `message` into the `Storage`
    - `read_value` returns `value` from `Storage` using the internal function `_get_value`

Writing Tests

In `test_visibility_2.cairo`, inside `test_internal_external()`, complete the following:

1. Declare the contract and get it's `class_hash`
2. Create a contract dispatcher
3. Create a variable `state` to hold and assign the contract state to it (check the useful links if you're not sure how)
4. Make sure that the public `read_value()` function returns the same value that is in the state
5. Call the public `set_value()` function to update the storage value.
6. Make sure that the value in the state and the value returned by `read_value()` are updated
7. Update the storage value using the contract state
8. Make sure that the value in the state and the value returned by `read_value()` are updated

Check the exercise by running the command `snforge test visibility_2`.

Useful Links

Starknet Foundry: Testing Contracts Internals

Contract functions