Cairo Crash Course

Functions Visibility - Exercise 1

Let's practice testing the visibility of functions in our smart contracts. We'll start with implementing and testing private / internal functions 💪

Contract Implementation

In the `visibility_exercise_1.cairo` file, complete the contract as follows:

1. Create a `Struct` named `Storage` with the field:
    - `funds: u32`
2. Create a dedicated `impl` block `InternalImpl` that implements `InternalTrait` with the `#[generate_trait]` attribute containing:
    - A function `_get_funds` that reads and returns the `funds` from `Storage`.
3. Outside the `impl` block, create a function `_add_funds` that receives an amount `u32` and modifies the state by adding the `amount` to the `funds` in `Storage`, and returns the new `funds` that's in storage.

Writing Tests

In the `test_visibility_1.cairo` file, inside the `test_internal()` function, complete the following tests:

1. Declare a variable named `state` to hold the contract state. Look at the linked resources for help 😃
2. Use `state` to write `10` directly to the `funds` field.
3. Declare a variable named `value`, then use the `state` variable to call the internal function `_get_funds` and assign the result to `value`.
4. Declare a variable `other_value`. Assign to it the result of `_add_funds`, passing `state` by reference and an `amount` of `300`.
5. Make sure that both the returned values and the state are correct.

Check that all the code compiles and the tests pass by running the command `snforge test visibility_1`.

Useful Links

Starknet Foundry: Testing Contracts Internals

Contract functions