**Setting The Project**

**Introduction:**

In this `StorageFactory` setup, we’ll explore what composability means, showing its ability to deploy and interact with external `SimpleStorage` contracts.

**StorageFactory setup:**

You can begin by visiting the Github repository of the previous section (<https://github.com/cyfrin/remix-simple-storage-cu>) and copying the contract `SimpleStorage` inside Remix.

This contract allows to store a favorite number, a list of people with their favorite number, a mapping and different functionalities to interact with them.

This lesson aims to create a new contract that can deploy and interact with `SimpleStorage`.

**Important:** One of the fundamental aspects of blockchain development is the seamless and permissionless interaction between contracts, known as \*\*composability\*\*. This is particularly crucial in decentralized finance (DeFi), where complex financial products interact effortlessly through common smart contract interfaces.

Let’s set up the backbone of the code, that contains the function `createSimplestorageContract`. This function will deploy a `SimpleStorage` contract and save the result into a storage variable:

// SPDX-License-Identifier: MIT

Pragma solidity ^0.8.18;

Contract StorageFactory {

Function createSimplestorageContract() public {

// How does StorageFactory know what SimpleStorage looks like?

}

}

We need to establish a connection between the two contracts, since `StorageFactory` needs to have a complete knowledge of `SimpleStorage`. One first approach could be copying the `SimpleStorage` contract above `StorageFactory`.

**Note:** It’s allowed to have multiple contracts in the same file. As best practice, however, it’s recommended to use only one file for each contract

**Tip:** You can avoid confusion by keeping open only the file(s) you’re currently working on.

**Conclusion:**

In this setup, we’ll delve into the concept of **composability** and develop the `StorageFactory` contract, which will be capable of deploying and interacting with a `SimpleStorage` contract.