Blockchain Study Notes Day 5:

Module 2 - Solidity Basics
Chapter 1 - Introduction to Solidity & First Program

Introduction to Solidity

What is Solidity?

- Solidity is a high-level, statically-typed programming language designed specifically for writing smart contracts on the Ethereum blockchain.
- Influenced by languages like JavaScript, Python, and C++.

Key Features:

- **Contract-Oriented**: Focused on creating and managing smart contracts.
- **Type-Safe**: Variables must be declared with specific data types.
- **Ethereum Compatibility**: Supports features unique to the Ethereum blockchain, like ether transactions and gas management.

Basic Structure of a Solidity Program

A typical Solidity program consists of:

1. **Pragma Directive**: Specifies the compiler version.
2. **Contract Definition**: Contains state variables, functions, and logic.
3. **Functions**: Define the behavior of the contract.

First Solidity Program: "Hello, World!"

Step 1: Open Remix IDE

- Go to Remix IDE.

Step 2: Create a New File

- Create a new file named `HelloWorld.sol`.

Step 3: Write the Program

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HelloWorld {
    // State variable to store the message
    string public message;

    // Constructor to set the initial message
    constructor() {
        message = "Hello, World!";
    }

    // Function to get the current message
    function getMessage() public view returns (string memory) {
        return message;
    }
}
```

Explanation:

1. **`pragma solidity ^0.8.0;`**
    - Ensures the code is compatible with Solidity version 0.8.0 or higher.
2. **`contract HelloWorld`**
    - Defines a contract named `HelloWorld`.
3. **State Variable (`string public message`)**
    - Stores the message "Hello, World!" and makes it publicly accessible.
4. **Constructor**
    - Initializes the `message` variable when the contract is deployed.
5. **Function (`getMessage`)**
    - Returns the current value of `message`.

Deploy and Test the Program

1. **Compile the Program**:
    - Use the "Solidity Compiler" tab to compile `HelloWorld.sol`.
2. **Deploy the Contract**:
    - Go to the "Deploy & Run Transactions" tab.
    - Select the environment (e.g., JavaScript VM).
    - Click "Deploy."
3. **Interact with the Contract**:
    - After deployment, call the `getMessage` function to display "Hello, World!" in the output panel.

Home Task

1. **Modify the Program**:
    - Add a new function `setMessage` to update the message.

```
function setMessage(string memory _newMessage) public {
    message = _newMessage;
}
```

2. **Deploy and Test**:
    - Update the message using `setMessage` and verify it using `getMessage`.
3. **Explore Solidity Syntax**:
    - Learn about data types (e.g., `uint`, `bool`, `address`) and their usage in smart contracts.

Conclusion

Solidity is the foundational language for Ethereum smart contract development. By starting with a simple "Hello, World!" program, you've taken the first step in understanding how to write and deploy smart contracts. As you progress, you'll learn to build more complex and interactive blockchain applications.

Day 5 Notes

Prepared by Munawar Johar