SOLIDITY SMART CONTRACT DEVELOPMENT ESSENTIALS

Chulliyev Shokhrukh Ibadullayevich, *TUIT*
Normuminov Akbar Kamol ugli, *TUIT*

Abstract: This article provides a comprehensive overview of Solidity, the programming language used for creating smart contracts on the Ethereum blockchain. It explains the fundamental concepts of Solidity, such as contracts, objects, modifiers, events, and interfaces, with clear and concise examples. The article also highlights the importance of Solidity in enabling the automation of actions on the blockchain through self-executing conditions. Additionally, it discusses how Solidity compares to other languages like C++ and JavaScript, emphasizing its unique syntax and purpose for Ethereum Virtual Machine (EVM) code generation. Overall, this article serves as an excellent introduction to Solidity, making it accessible to beginners while also offering insights for more experienced developers looking to build decentralized applications (dApps) on the Ethereum platform.

Key words: Solidity, smart contracts, Ethereum, blockchain, programming language, contracts, objects, modifiers, events, interfaces, automation, Ethereum Virtual Machine (EVM), C++, JavaScript, decentralized applications (dApps).

Solidity is the most popular language for programming on the blockchain, used to create "smart contracts" or "autonomous conditions." Solidity is specifically designed for creating smart contracts on the Ethereum blockchain platform.

Smart contracts are conditions that operate on the blockchain, executing actions automatically. These conditions execute based on the specified terms and persist until completed, known as "self-executing."

Similar to C++ and JavaScript in syntax, Solidity is tailored for creating code for the Ethereum Virtual Machine (EVM). Smart contracts created using Solidity operate on the Ethereum blockchain when deployed and are executed automatically among users.

Other blockchain platforms, such as Cardano, Polkadot, Solana, and others, exist for creating cryptocurrencies. Each provides their own programming languages and capabilities for creating smart contracts.

Solidity is a programming language specifically designed for creating smart contracts for the Ethereum blockchain. This language is used to write code for the Ethereum Virtual Machine (EVM) and deploy smart contracts for execution.

Key Features of Solidity Programming Language:

- Contracts: In Solidity, the keyword "contract" is used to create smart contracts.
- Objects and Structs: Solidity supports objects and structs.

•Modifiers: Conditions can be added to functions using "modifiers."

•Events: Events are used in Solidity to support variables.

•Interfaces: Solidity supports interfaces for collaboration with other smart contracts or objects on the Ethereum platform.

The following example demonstrates a simple Solidity smart contract code. It creates and deploys a smart contract named "HelloWorld" and puts it into operation:

1.Example of a simple smart contract in Solidity

```
// Contract name HelloWorld
contract HelloWorld {
// string variable that stores the welcome message
string private welcomeMessage;
// Function executed when the contract is created
constructor() {
   welcomeMessage = "Hello, friends!";
}
// Function to read the welcome message
function getWelcomeMessage() public view returns (string memory) {
   return welcomeMessage;
}
// Function to change the welcome message
function setWelcomeMessage(string memory newMessage) public {
   welcomeMessage = newMessage;
}
}
```

In this code, a simple smart contract named HelloWorld is created. This contract has a string variable named welcomeMessage, which contains the value "Hello, friends!" when created. The getWelcomeMessage function returns this welcome message, while the setWelcomeMessage function changes it.

This example illustrates the basic way to create a simple smart contract and deploy it in Solidity.

2.Creating a Smart Contract

```
pragma solidity ^0.8.0;
// Smart contract name
contract MyToken {
// Balances of customers
mapping(address => uint256) public balances;
// Name of the smart contract
string public name = "My Token";
string public symbol = "MTK";
```

```
uint8 public decimals = 18;
uint256 public totalSupply = 1000000 * 10 ** uint256(decimals);
// Commands executed when the smart contract is created
constructor() {
    // Assigning the customer's balance when the smart contract is created
    balances[msg.sender] = totalSupply;
}
// Sending money to customers
function transfer(address _to, uint256 _value) public returns (bool success) {
    require(balances[msg.sender] >= _value, "Sender does not have enough balance");
    balances[msg.sender] -= _value;
    balances[_to] += _value;
    emit Transfer(msg.sender, _to, _value);
    return true;
}
// Event used to correctly reflect changes in balances within the smart contract
event Transfer(address indexed _from, address indexed _to, uint256 _value);
}
```

This code creates a simple smart contract called "MyToken." This smart contract represents a financial instrument on the Ethereum blockchain with the symbol "MTK." The smart contract includes a function for transferring funds between customers. This example is simple but demonstrates how to use smart contracts in Solidity.

References:

1."Mastering Ethereum: Building Smart Contracts and DApps" by Andreas M. Antonopoulos and Gavin Wood (2018)

2."Blockchain Basics: A Non-Technical Introduction in 25 Steps" by Daniel Drescher (2017)

3."Building Ethereum Dapps: Decentralized Applications on the Ethereum Blockchain" by Roberto Infante (2018)

4."Ethereum Programming: Solidity for Beginners" by Victor Finch (2018)