Lab9: Bespoke Ethereum Blockchain Tokens

Introduction

Ethereum's blockchain technology has a wide range of applications, from supply chain management to real estate sales. In this Lab9, we'll focus on creating bespoke Ethereum tokens (jetons Ethereum sur mesure), adhering to the ERC-20 standard.

Setting Up

1. **Access Remix IDE**: Visit remix.ethereum.org for the online development environment.
2. **New UI**: If prompted, switch to the new Remix interface.
3. **File Explorer**: Use the file explorer to manage your source files.

Creating Utility Functions

1. **New File**: Create a file named `Utils.sol`.
2. **Utility Code**: Add the following code to `Utils.sol`:

```
pragma solidity ^0.8.0;
library Utils {
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }
}
```

Defining Your ERC-20 Token

1. ERC-20 Token File: Create a file named ERC20.sol.
2. Import Utils: Add import "./Utils.sol"; to ERC20.sol.
3. Token Contract: Define your token contract,

```
pragma solidity ^0.8.0;
import "./Utils.sol";
contract Lab9Token {
    using Utils for *;
    string public constant name = "DA51-Lab9";
    string public constant symbol = "Lab9";
```

```
     uint8 public constant decimals = 18;

     event Approval(address indexed tokenOwner, address indexed spender,
uint tokens);
     event Transfer(address indexed from, address indexed to, uint tokens);
     mapping(address => uint256) balances;
     mapping(address => mapping (address => uint256)) allowed;
      uint256 totalSupply_;
     // Additional functions and constructor here...
 }
```

Implementing ERC-20 Functions

1. Add Events: Include Approval and Transfer events.
2. Manage Balances: Use mapping for balances and allowances.
3. Token Functions: Implement the standard ERC-20 functions like transfer, approve, and balanceOf.

Testing and Deployment

1. Compile: Compile your token in Remix.
2. Deploy: Deploy on a test network for trials.
3. Interact: Test token functionalities like transfers and approvals.

Conclusion

By the end of this Lab9, you'll be able to create a custom ERC-20 token for your Ethereum-based applications, understanding the requirements and functionalities of the ERC-20 standard.

Hint 1: simple code examples for the ERC-20 functions transfer, approve, and balanceOf

1. The transfer Function: This function allows a token holder (the "owner") to transfer a certain number of tokens to another Ethereum address.

```
    function transfer(address to, uint tokens) public returns (bool) {
        require(to != address(0), "Invalid address");
        require(balances[msg.sender] >= tokens, "Insufficient balance");

        balances[msg.sender] -= tokens;
        balances[to] += tokens;
        emit Transfer(msg.sender, to, tokens);
        return true;
    }
```

2. The approve Function: This function enables the token owner to authorize another address (the "spender") to withdraw a certain number of tokens from their account.

```
    function approve(address spender, uint tokens) public returns (bool) {
        require(spender != address(0), "Invalid spender address");

        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }
```

3. The balanceOf Function: This function returns the balance of tokens held by a given address.

```
    function balanceOf(address tokenOwner) public view returns (uint) {
        return balances[tokenOwner];
    }
```

Hint 2: To add event declarations (emit) as well to enable monitoring of transfers and approvals on the blockchain.

These events can be monitored on the Ethereum blockchain, and their data can be used to track token transfers and approvals.The events are used to log transfers of tokens and approvals on the Ethereum blockchain so that they can be monitored and tracked. In this example, the Transfer event is emitted when tokens are transferred from one address to another, and the Approval event is emitted when an address approves another address to spend tokens on its behalf.

```
    pragma solidity ^0.8.0;
    contract MyToken {
        string public name = "My Token";
        string public symbol = "MTK";
        uint8 public decimals = 18;
        uint256 public totalSupply;

        mapping(address => uint256) public balanceOf;
        mapping(address => mapping(address => uint256)) public allowance;

        event Transfer(address indexed from, address indexed to, uint256 value);
        event Approval(address indexed owner, address indexed spender, uint256
value);

        constructor(uint256 initialSupply) {
            totalSupply = initialSupply * 10 ** uint256(decimals);
            balanceOf[msg.sender] = totalSupply;
        }
```

```
        function transfer(address to, uint256 value) public returns (bool) {
            require(to != address(0), "Invalid address");
            require(balanceOf[msg.sender] >= value, "Insufficient balance");

            balanceOf[msg.sender] -= value;
            balanceOf[to] += value;
            emit Transfer(msg.sender, to, value);
            return true;
        }

        function approve(address spender, uint256 value) public returns (bool) {
            require(spender != address(0), "Invalid spender address");

            allowance[msg.sender][spender] = value;
            emit Approval(msg.sender, spender, value);
            return true;
        }

        function transferFrom(address from, address to, uint256 value) public
returns (bool) {
            require(from != address(0), "Invalid from address");
            require(to != address(0), "Invalid to address");
            require(balanceOf[from] >= value, "Insufficient balance");
            require(allowance[from][msg.sender] >= value, "Allowance exceeded");

            balanceOf[from] -= value;
            balanceOf[to] += value;
            allowance[from][msg.sender] -= value;
            emit Transfer(from, to, value);
            return true;
        }
    }
```

Additional Notes

Ensure thorough testing before deploying on the main network. Be mindful of legal and security aspects in token creation.