

Nama : Raihan Puspa Anggita Putri

NIM : 1103184065

Kelas : TK-42-PIL

Lesson 12 : Hardhat ERC20s

```
interface tokenRecipient {
    function receiveApproval(
        address _from,
        uint256 _value,
        address _token,
        bytes calldata _extraData
    ) external;
}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;

    // This creates an array with all balances
    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;

    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);

    // This generates a public event on the blockchain that will notify clients
    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint256 _value
    );

    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);
```

```

43  constructor(
44      uint256 initialSupply,
45      string memory tokenName,
46      string memory tokenSymbol
47  ) {
48      totalSupply = initialSupply * 10**uint256(decimals); // Update total supply with the decimal amount
49      balanceOf[msg.sender] = totalSupply; // Give the creator all initial tokens
50      name = tokenName; // Set the name for display purposes
51      symbol = tokenSymbol; // Set the symbol for display purposes
52  }
53
54  /**
55   * Internal transfer, only can be called by this contract
56   */
57  function _transfer(
58      address _from,
59      address _to,
60      uint256 _value
61  ) internal {
62      // Prevent transfer to 0x0 address. Use burn() instead
63      require(_to != address(0x0));
64      // Check if the sender has enough
65      require(balanceOf[_from] >= _value);
66      // Check for overflows
67      require(balanceOf[_to] + _value >= balanceOf[_to]);
68      // Save this for an assertion in the future
69      uint256 previousBalances = balanceOf[_from] + balanceOf[_to];
70      // Subtract from the sender
71      balanceOf[_from] -= _value;
72      // Add the same to the recipient
73      balanceOf[_to] += _value;
74      emit Transfer(_from, _to, _value);
75      // Asserts are used to use static analysis to find bugs in your code. They should never fail
76      assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
77  }

```

```

function transfer(address _to, uint256 _value) public returns (bool success) {
    _transfer(msg.sender, _to, _value);
    return true;
}

/**
 * Transfer tokens from other address
 *
 * Send `_value` tokens to `_to` on behalf of `_from`
 *
 * @param _from The address of the sender
 * @param _to The address of the recipient
 * @param _value the amount to send
 */
function transferFrom(
    address _from,
    address _to,
    uint256 _value
) public returns (bool success) {
    require(_value <= allowance[_from][msg.sender]); // Check allowance
    allowance[_from][msg.sender] -= _value;
    _transfer(_from, _to, _value);
    return true;
}

/**
 * Set allowance for other address
 *
 * Allows `_spender` to spend no more than `_value` tokens on your behalf
 *
 * @param _spender The address authorized to spend
 * @param _value the max amount they can spend
 */
function approve(address _spender, uint256 _value)
    public
    returns (bool success)
{

```

```

7 function burn(uint256 _value) public returns (bool success) {
8     require(balanceOf[msg.sender] >= _value); // Check if the sender has enough
9     balanceOf[msg.sender] -= _value; // Subtract from the sender
10    totalSupply -= _value; // Updates totalSupply
11    emit Burn(msg.sender, _value);
12    return true;
13 }
14
15 /**
16  * Destroy tokens from other account
17  *
18  * Remove `_value` tokens from the system irreversibly on behalf of `_from`.
19  *
20  * @param _from the address of the sender
21  * @param _value the amount of money to burn
22  */
23 function burnFrom(address _from, uint256 _value)
24     public
25     returns (bool success)
26 {
27     require(balanceOf[_from] >= _value); // Check if the targeted balance is enough
28     require(_value <= allowance[_from][msg.sender]); // Check allowance
29     balanceOf[_from] -= _value; // Subtract from the targeted balance
30     allowance[_from][msg.sender] -= _value; // Subtract from the sender's allowance
31     totalSupply -= _value; // Update totalSupply
32     emit Burn(_from, _value);
33     return true;
34 }
35 }
36

```

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";  
  
contract OurToken is ERC20 {  
    constructor(uint256 initialSupply) ERC20("OurToken", "OT") {  
        _mint(msg.sender, initialSupply);  
    }  
}
```