**Assignment 3**

**Title:**

Write a smart contract on test network ,for the bank account of customers for following operations:

1. Deposite Money
2. Withdraw Money
3. Show Balance

**Program :-**

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract SimpleBank {

    mapping(address => uint256) private balances;

    address owner;

    // Constructor is "payable" so it can receive ether,

    constructor() public payable {

        /\* Set the owner to the creator of this contract \*/

        owner = msg.sender;

    }

    /// @notice Deposit ether into bank, requires method is "payable"

    /// @return The balance of the user after the deposit is made

    function deposit(uint256 depoamount) public payable returns (uint256) {

        balances[msg.sender] += depoamount;

        payable(msg.sender).transfer(depoamount);

        //emit LogDepositMade(msg.sender, msg.value);

        return balances[msg.sender];

    }

    /// @notice Withdraw ether from bank

    /// @return The balance remaining for the user

    function withdraw(uint256 withdrawAmount) public returns (uint256) {

        // Check enough balance available, otherwise just return balance

        if (withdrawAmount <= balances[msg.sender]) {

            balances[msg.sender] -= withdrawAmount;

            payable(msg.sender).transfer(withdrawAmount);

        }

        return balances[msg.sender];

    }

    function balance() public view returns (uint256) {

        return balances[msg.sender];

    }

}

Output :-

**[vm]**

**from:** 0x5B3...eddC4

**to:** SimpleBank.(constructor)

**value:** 10000000000000000000 wei

**data:** 0x608...70033

**logs:** 0

**hash:** 0xfae...d6d8d

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xfaee06b600d41a504f9c9723bf8c830bcda4fc8b1054b21e1dbe878c3fad6d8d |
| from | 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 |
| **to** | SimpleBank.(constructor) |
| **gas** | 382650 gas |
| **transaction cost** | 332739 gas |
| **execution cost** | 332739 gas |
| **input** | 0x608...70033 |
| **decoded input** | {} |
| **decoded output** | - |
| **logs** | [] |
| **val** | 10000000000000000000 wei |

transact to SimpleBank.deposit pending ...

**[vm]**

**from:** 0x5B3...eddC4

**to:** SimpleBank.deposit(uint256) 0xd91...39138

**value:** 0 wei

**data:** 0xb6b...00064

**logs:** 0

**hash:** 0xefc...265c1

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xefc8bbd6351a99401198e3d5ba9a43a376ecbc5bed92190e244168afeb3265c1 |
| from | 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 |
| **to** | SimpleBank.deposit(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 59002 gas |
| **transaction cost** | 51306 gas |
| **execution cost** | 51306 gas |
| **input** | 0xb6b...00064 |
| **decoded input** | { "uint256 depoamount": "100" } |
| **decoded output** | { "0": "uint256: 100" } |
| **logs** | [] |
| **val** | 0 wei |

transact to SimpleBank.withdraw pending ...

**[vm]**

**from:** 0x5B3...eddC4

**to:** SimpleBank.withdraw(uint256) 0xd91...39138

**value:** 0 wei

**data:** 0x2e1...00046

**logs:** 0

**hash:** 0x941...ef037

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x9410fdb21f866d55917b68af4fa11310906d3a03c0285acf07d008fa1beef037 |
| from | 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 |
| **to** | SimpleBank.withdraw(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 39551 gas |
| **transaction cost** | 34392 gas |
| **execution cost** | 34392 gas |
| **input** | 0x2e1...00046 |
| **decoded input** | { "uint256 withdrawAmount": "70" } |
| **decoded output** | { "0": "uint256: 30" } |
| **logs** | [] |
| **val** | 0 wei |

call to SimpleBank.balance

