IT465 Assignment 1 - Report

Sachin Prasanna - 211IT058

August 29, 2024

1 Problem Statement

1) Build a basic Ethereum smart contract for a very simple cell phone contract between a cell company and a subscriber.

The subscriber pays a monthly fee in ether to the contract, and the cell phone company can check if the account is paid in full. The contract should be deployed with an established monthly cost and accept ether bill payments from a subscriber. The company should be able to check the status of the account on a given date. Additionally, the company should have the ability to withdraw funds from the contract to transfer them to their own accounts.

2) Write a Simple Bank smart contract code in Solidity programming language.

2 Solution to Question 1

2.1 Code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.26;

contract CellPhoneContract {
    address public company;
    uint public monthlyFee;
    uint private companyBalance;
    mapping(address => mapping(uint => bool)) private payments;

    constructor(uint _monthlyFee) {
        company = msg.sender;
        monthlyFee = _monthlyFee;
    }

    modifier onlyCompany() {
        require(msg.sender == company, "Only the company can perform this
            action");
        _;
    }
```

```
19
20      modifier onlySubscriber() {
21          require(msg.sender != company, "Company cannot perform this action");
22          _;
23      }
24
25      function payBill(uint month) public payable onlySubscriber {
26          require(msg.value == monthlyFee, "Incorrect fee amount");
27          require(!payments[msg.sender][month], "Already paid for this month");
28
29          companyBalance += msg.value;
30          payments[msg.sender][month] = true;
31      }
32
33      function checkPaymentStatus(address subscriber, uint month) public view
            onlyCompany returns (bool) {
34          return payments[subscriber][month];
35      }
36
37      function getCompanyBalance() public view onlyCompany returns (uint) {
38          return companyBalance;
39      }
40
41      function withdraw() public onlyCompany {
42          uint amount = companyBalance;
43          require(amount > 0, "No funds available for withdrawal");
44          companyBalance = 0;
45          payable(company).transfer(amount);
46      }
47  }
```

2.2 Functionalities

The `CellPhoneContract` smart contract manages a cell phone subscription service. It allows subscribers to pay a monthly fee in ether, which is tracked and verified within the contract.

- **Payment Processing**: Subscribers can pay their monthly bill by sending the exact fee amount to the contract. The contract verifies the payment amount and ensures that each month is paid only once per subscriber.
- **Payment Verification**: The contract includes a function for the company to check whether a subscriber has paid for a specific month.
- **Balance Tracking**: The contract maintains a private balance for the company, accumulating all payments received from subscribers.
- **Funds Withdrawal**: The company can withdraw the accumulated funds from the contract, transferring them to its own account.
- **Access Control**: Access modifiers ensure that only the company can perform administrative actions like checking payment statuses and withdrawing funds, while subscribers can only pay bills.

2.3 Outputs

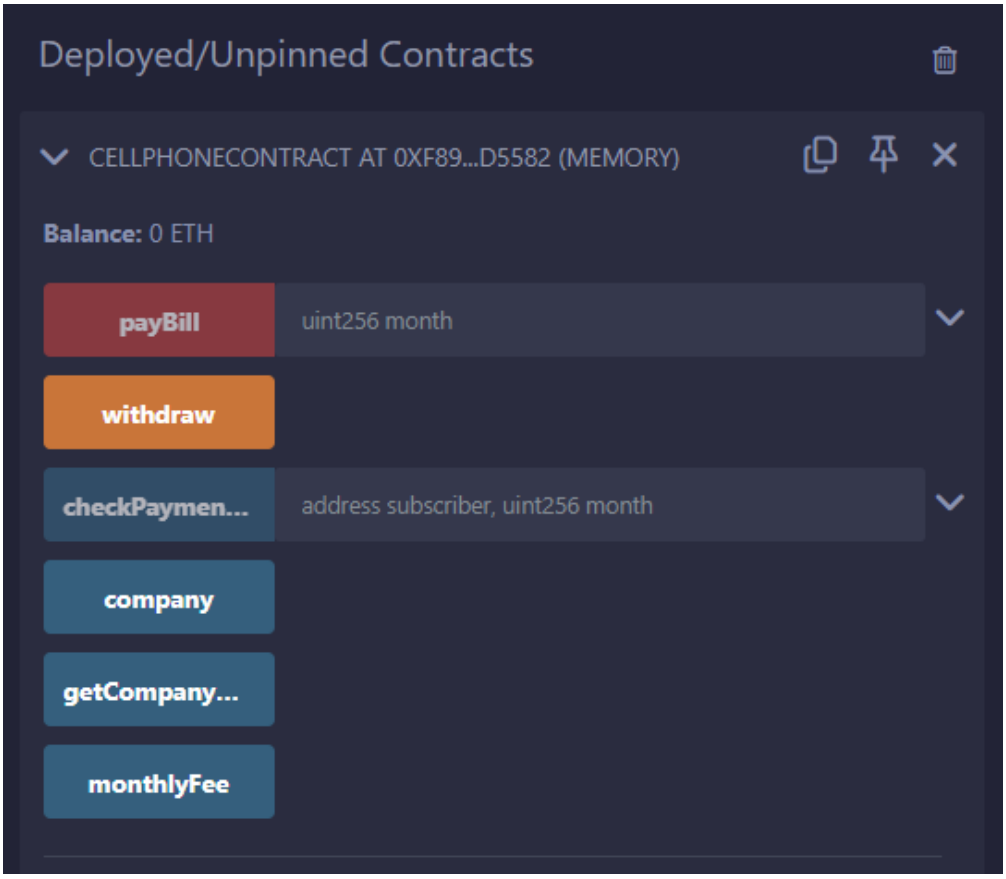

Figure 1: Confirmation of contract deployment. Contract Creator becomes the Company.

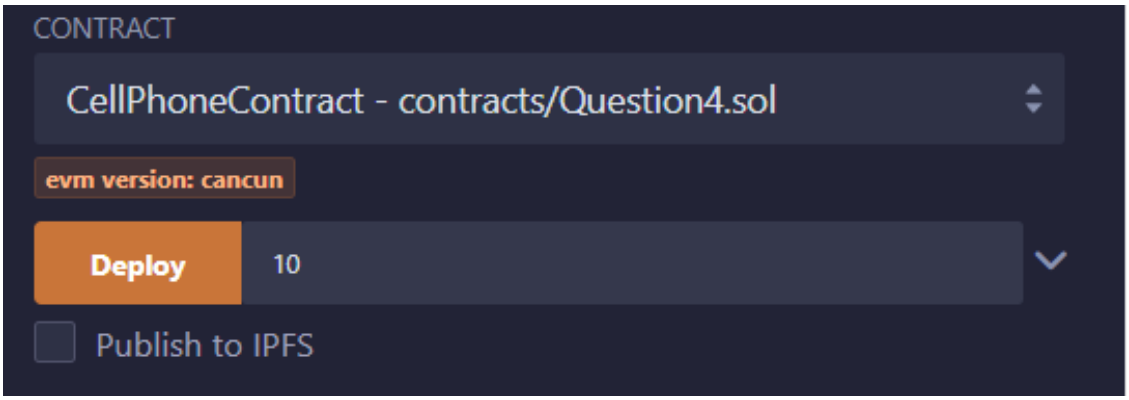

Figure 2: Setting Monthly Fee as 10 Wei in the Constructor

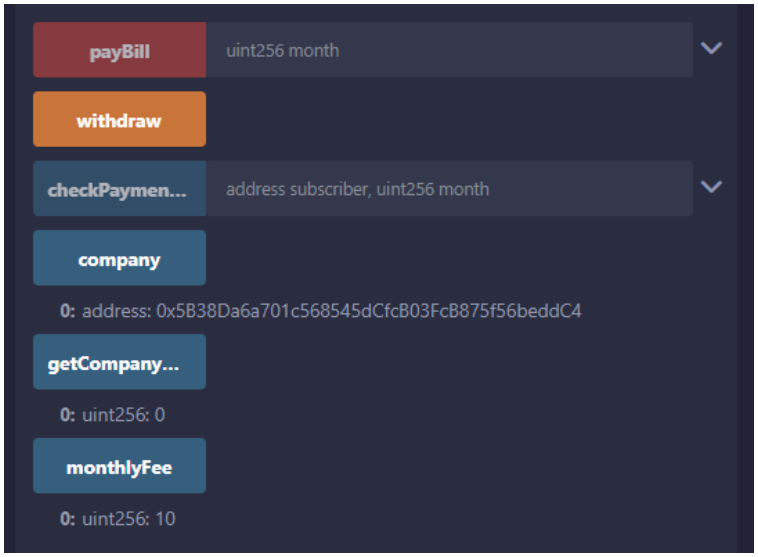

Figure 3: Viewing Company Address, Company Balance and the Monthly Fee

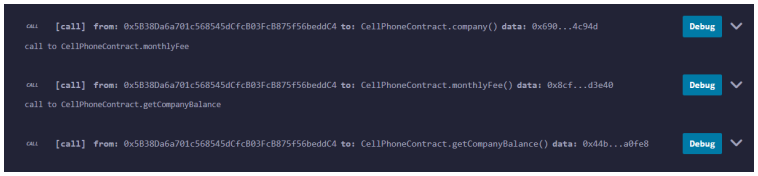

Figure 4: Corresponding Success Logs for the above 3 operations

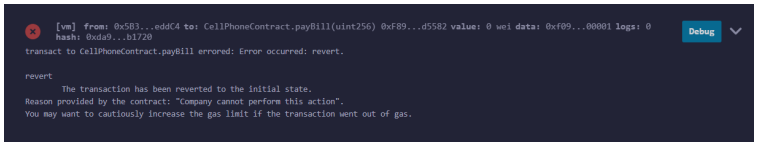

Figure 5: Error countered when company tries to pay bill, as it is not allowed

Figure 6: Switching to Subscriber Account and setting the monthly value of 10 Wei to pay

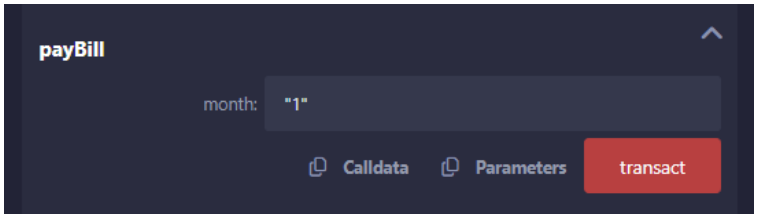

Figure 7: Paying for the first month (January)

Figure 8: Corresponding Success Log indicating success of transaction.

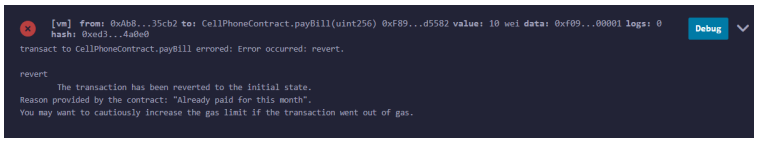

Figure 9: Error Handling Case: When the Subscriber tries to pay for the same month again

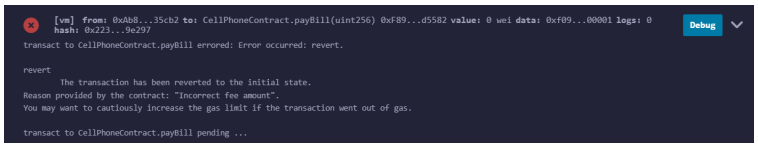

Figure 10: Error Handling Case: When the Subscriber tries to pay the incorrect monthly fee

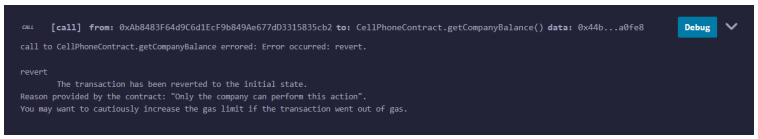

Figure 11: Blocking access to Company Balance to Subscribers

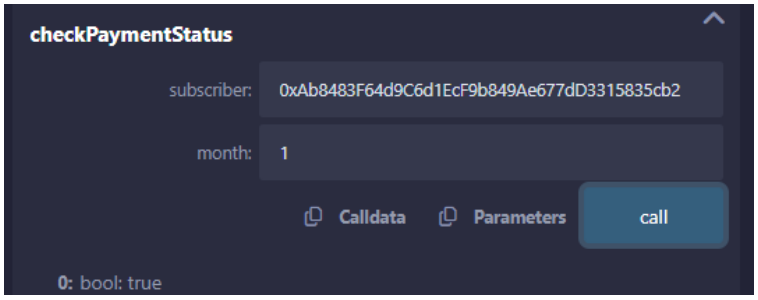

Figure 12: Checking Payment of Month 1 of the subscriber through Company, which return true

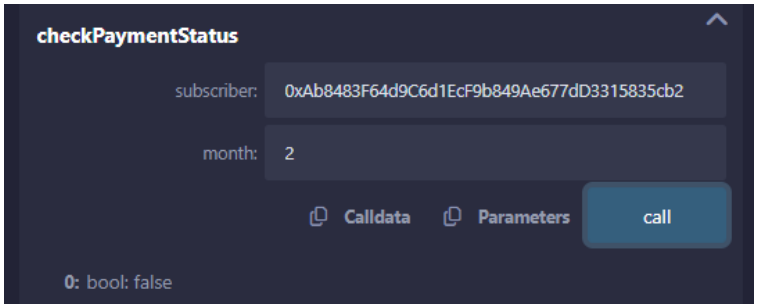

Figure 13: Checking Payment of Month 1 of the subscriber through Company, which return false, as it is not paid yet

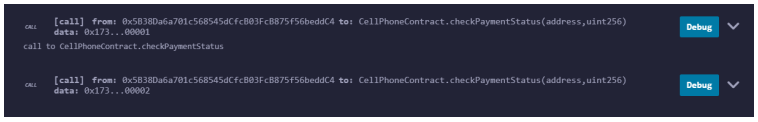

Figure 14: Corresponding Success Logs for the same

Figure 15: Fetching Companies Balance. It is 10 Wei as one subscriber just paid

Figure 16: Corresponding Success Log

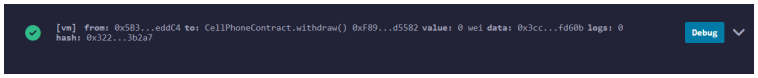

Figure 17: Company Withdraws all the Wei using the Withdraw button

Figure 18: Balance of Company after withdrawal of funds

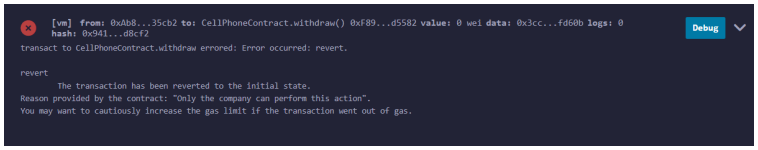

Figure 19: Result when Subscriber tries to Withdraw the money. Not allowed as only Company can withdraw its funds

3 Solution for Question 2

3.1 Code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.26;

contract SimpleBank {
    mapping(address => uint) private balances;

    function deposit() public payable {
        require(msg.value > 0, "Deposit amount must be greater than 0");
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint amount) public {
        require(amount > 0, "Withdrawal amount must be greater than 0");
        require(amount <= balances[msg.sender], "Insufficient balance");

        balances[msg.sender] -= amount;
        payable(msg.sender).transfer(amount);
    }

    function transfer(address to, uint amount) public {
        require(to != address(0), "Cannot transfer to zero address");
        require(amount > 0, "Transfer amount must be greater than 0");
        require(amount <= balances[msg.sender], "Insufficient balance");

        balances[msg.sender] -= amount;
        balances[to] += amount;
    }

    function checkBalance() public view returns (uint) {
        return balances[msg.sender];
    }
}
```

3.2 Functionalities

The `SimpleBank` contract provides the following functionalities:

- **Deposit:** Allows users to deposit ether into their account. The deposited amount is added to the user's balance.
- **Withdraw:** Enables users to withdraw a specified amount of ether from their account, provided they have sufficient balance.
- **Transfer:** Allows users to transfer ether from their account to another address. The transfer amount is deducted from the sender's balance and added to the recipient's balance.
- **Check Balance:** Allows users to view their current balance in the contract.

3.3 Outputs

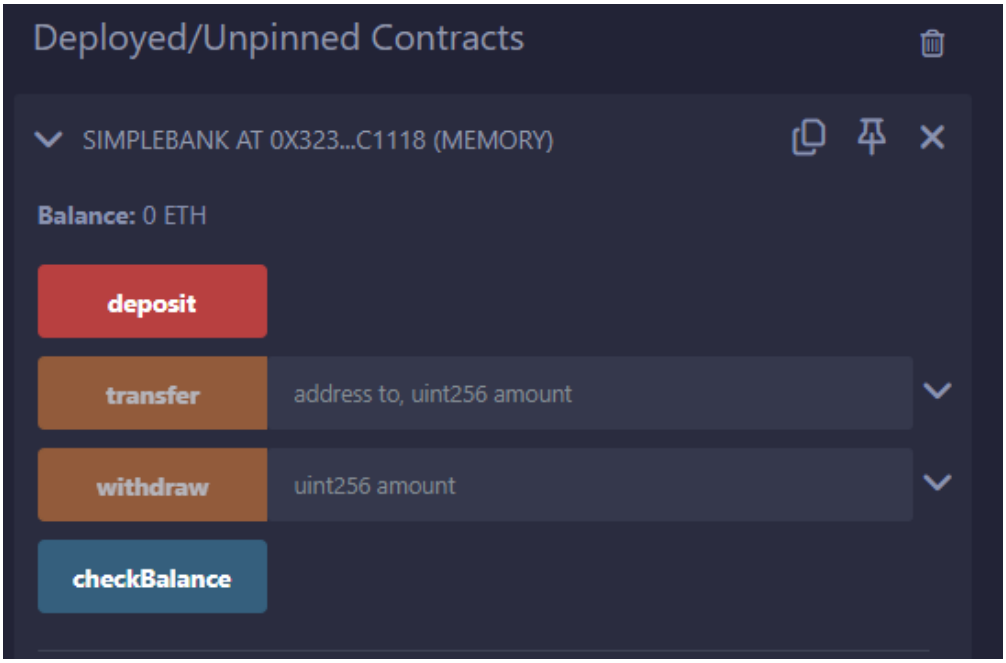

Figure 20: Confirmation of contract deployment

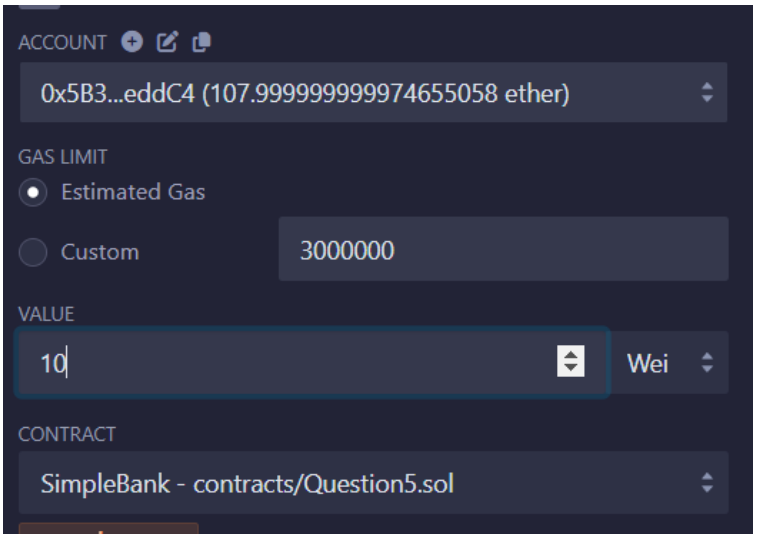

Figure 21: Deposition of 10 Wei into Account

Figure 22: Output after Deposition into Account

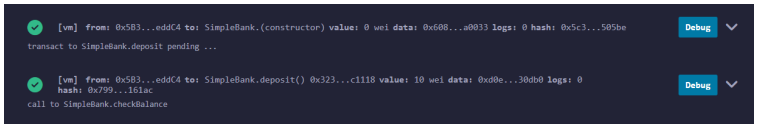

Figure 23: Corresponding Success log Messages of Deposition and Checking Balance

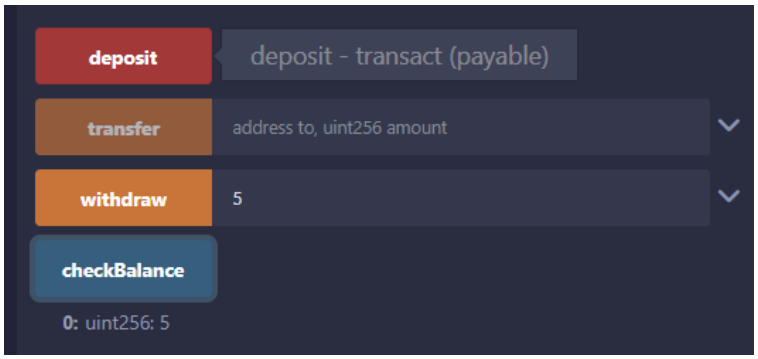

Figure 24: Output after Withdrawal of 5 Wei from Account

Figure 25: Corresponding Success log Messages of Withdrawal and Checking Balance

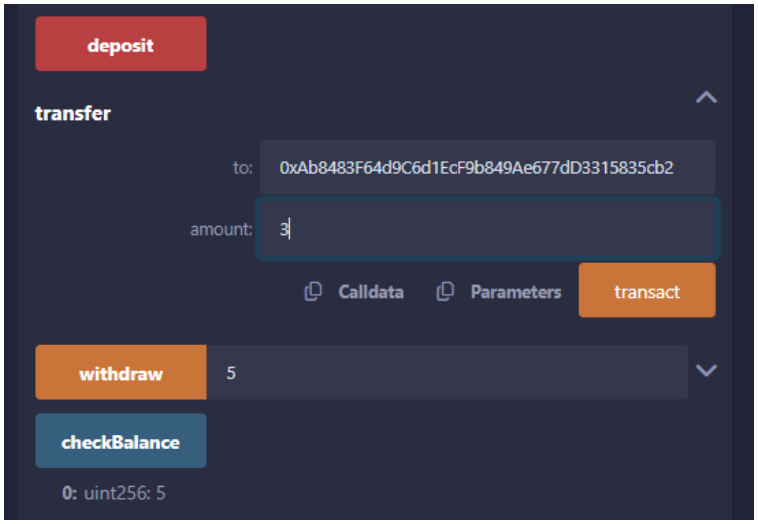

Figure 26: Transfer of 3 Wei to Another Account

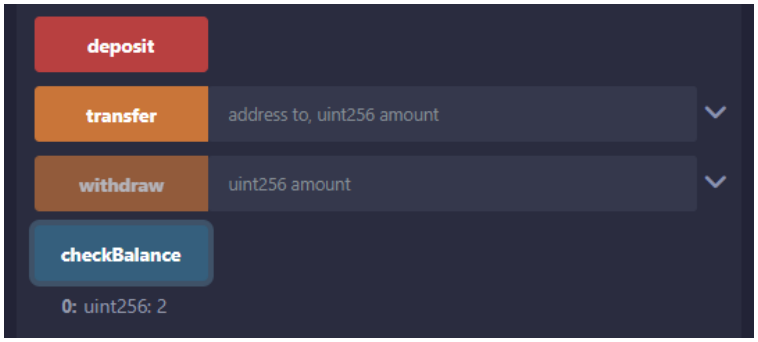

Figure 27: Balance of Account that sent 3 Wei

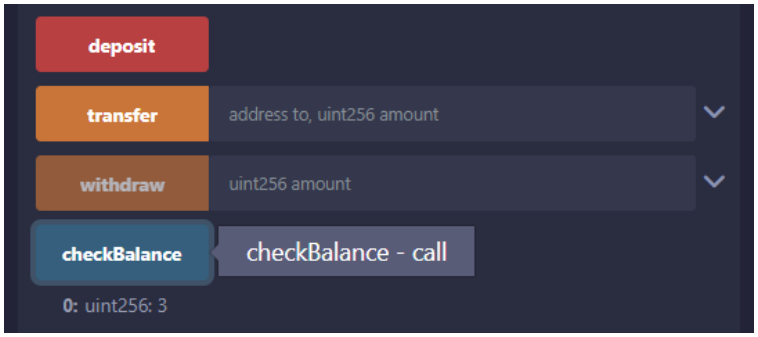

Figure 28: Balance of Account that received 3 Wei (which had no Wei before)

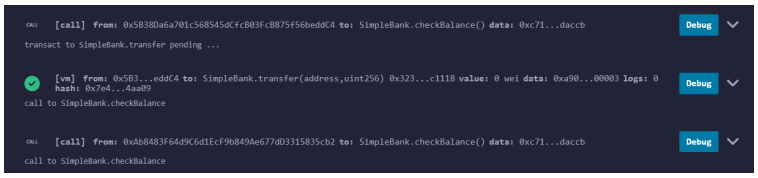

Figure 29: Corresponding Success log Messages of Transfer and Checking Balances

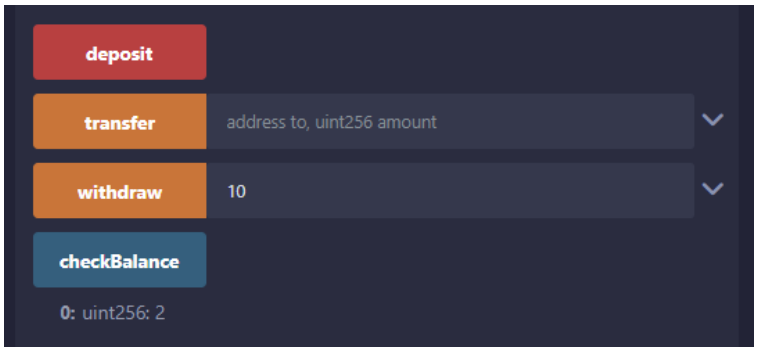

Figure 30: Error Handling Case: Attempting to withdraw more Wei than the available balance

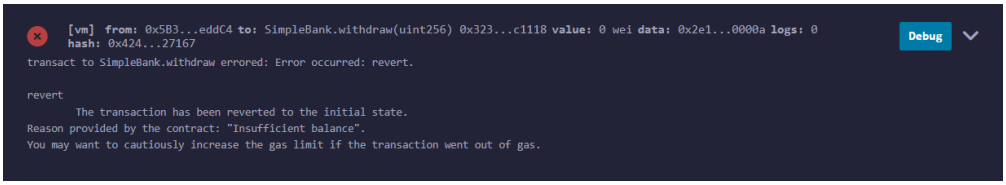

Figure 31: Corresponding Failure log Messages of the Error