

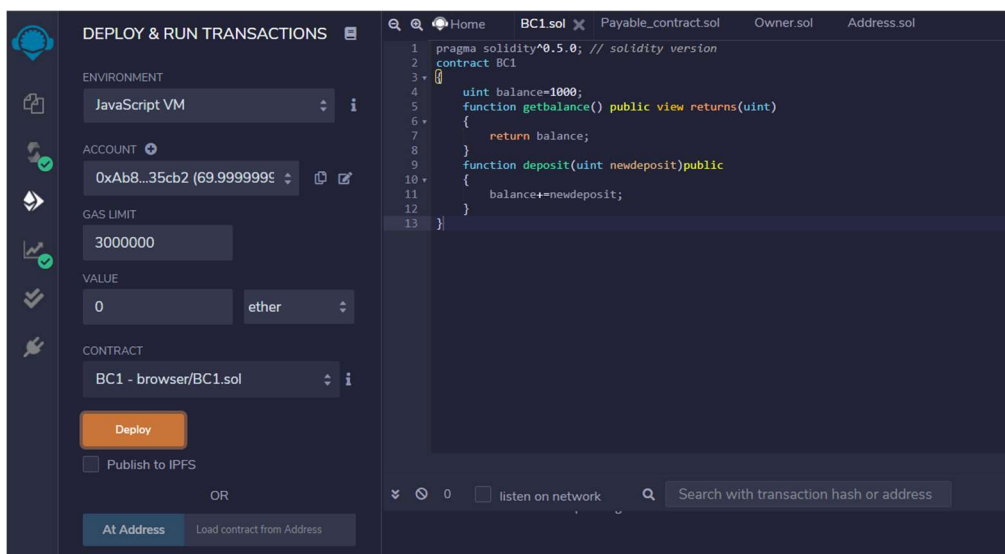
# CRYPTO CURRENCIES AND BLOCKCHAIN

## LAB\_2

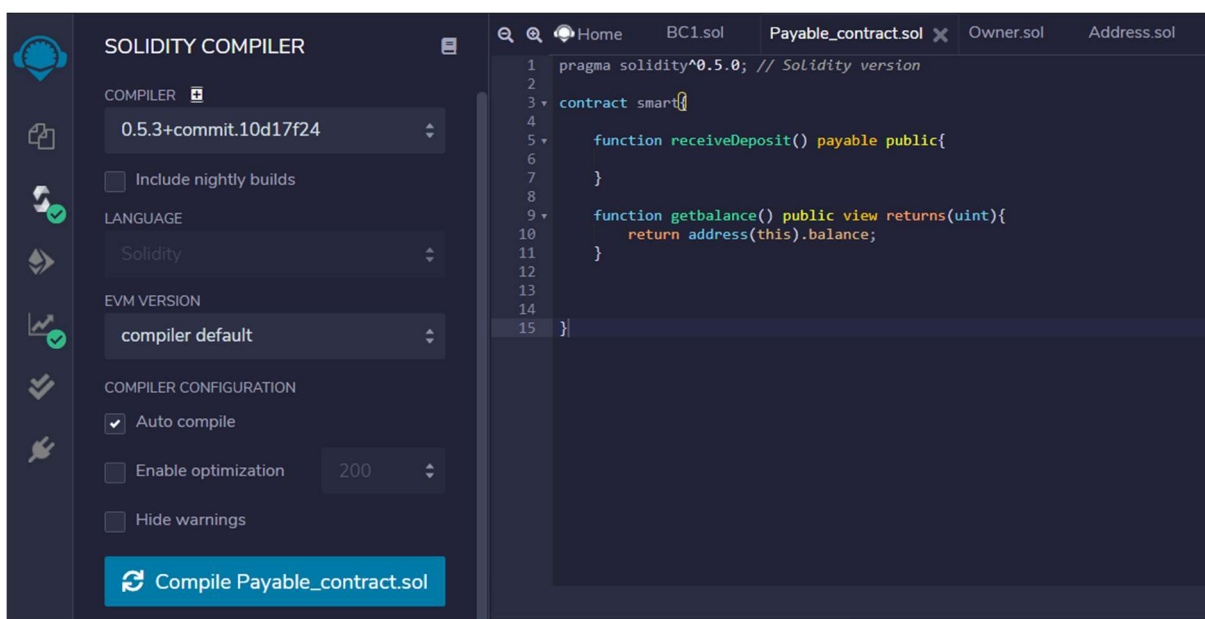
VIBHAV SHARMA

E18CSE206

Sample Test :



a) Transferring funds from an account to the contract using payable :



DEPLOY & RUN TRANSACTIONS

OR

At Address  Load contract from

Transactions recorded 2

All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

BC1 AT 0XAC4...3510D (MEMORY)

SMART AT 0X9EC...DDE84 (MEMORY)

receiveDeposit

getbalance

uint256: 0

Low level interactions

1 pragma solidity^0.5.0; // Solidity version

2

3 contract smart{

4

5 function receiveDeposit() payable public{

6 }

7

8

9 function getbalance() public view returns(uint){

10 return address(this).balance;

11 }

12

13

14 }

ContractDefinition smart 0 reference(s)

listen on network

Search with transaction hash or address

[vm] from: 0xAb8...35cb2 to: BC1.(constructor) value: 0 wei data: 0x608...50029 logs: 0 hash: 0x387...be472

creation of smart pending...

[vm] from: 0x4B2...C02db to: smart.(constructor) value: 0 wei data: 0x608...70029 logs: 0 hash: 0x85d...39e83

call to smart.getbalance

Starting Balance

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM

ACCOUNT

0x4B2...C02db (99.999999%)

GAS LIMIT

3000000

VALUE

10

ether

CONTRACT

smart - browser/Payable\_contract.sol

Deploy

Publish to IPFS

OR

At Address  Load contract from Address

Transactions recorded 2

Deployed Contracts

BC1 AT 0XAC4...3510D (MEMORY)

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM

ACCOUNT

0x4B2...C02db (89.999999%)

GAS LIMIT

3000000

VALUE

0

ether

CONTRACT

smart - browser/Payable\_contract.sol

Deploy

Publish to IPFS

OR

At Address  Load contract from Address

Transactions recorded 3

Deployed Contracts

BC1 AT 0XAC4...3510D (MEMORY)

1 pragma solidity^0.5.0; // Solidity version

2

3 contract smart{

4

5 function receiveDeposit() payable public{

6 }

7

8

9 function getbalance() public view returns(uint){

10 return address(this).balance;

11 }

12

13

14 }

ContractDefinition smart 0 reference(s)

listen on network

Search with transaction hash or address

[vm] from: 0xAb8...35cb2 to: BC1.(constructor) value: 0 wei data: 0x608...50029 logs: 0 hash: 0x387...be472

creation of smart pending...

[vm] from: 0x4B2...C02db to: smart.(constructor) value: 0 wei data: 0x608...70029 logs: 0 hash: 0x85d...39e83

call to smart.getbalance

1 pragma solidity^0.5.0; // Solidity version

2

3 contract smart{

4

5 function receiveDeposit() payable public{

6 }

7

8

9 function getbalance() public view returns(uint){

10 return address(this).balance;

11 }

12

13

14 }

ContractDefinition smart 0 reference(s)

listen on network

Search with transaction hash or address

transact to smart.receiveDeposit pending ...

[vm] from: 0x4B2...C02db to: smart.receiveDeposit() 0x9ec...dde84 value: 1000000000000000000000000 wei data: 0xe36...7ef5e logs: 0 hash: 0xc90...44935

call to smart.getbalance

[call] from: 0x4B209938c481177ec7E8f571ceCaE8A9e22C02db to: smart.getbalance() data: 0x4d9...b3d5d

Value to transfer

→ Ether Left

→ Transaction Successful



The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible. The 'ENVIRONMENT' is set to 'JavaScript VM'. The 'ACCOUNT' is '0x4B2...C02db (89.999999%)'. The 'GAS LIMIT' is '3000000'. The 'VALUE' is '20 ether'. The 'CONTRACT' is 'FC1 - browser/Transfer\_funds.sol'. The 'Deploy' button is highlighted. Below it, 'Publish to IPFS' is an option. The 'At Address' section shows 'Load contract from Address'. The 'Transactions recorded' section shows '4' transactions. The 'Deployed Contracts' section shows 'BC1 AT 0xAc4...3510D (MEMORY)' and 'SMART AT 0x9EC...DDE84 (MEMORY)'. The 'ADDRESS\_SAVE AT 0x688...58CF7 (MEMORY)' is selected. The 'address\_show' button is highlighted with a red arrow. The 'Low level interactions' section shows 'CALLDATA' and a 'Transact' button. The main editor shows the Solidity code for 'address\_save'.

```

1 pragma solidity^0.5.0;
2
3 contract address_save{
4
5     address owner;
6     constructor() public{
7         owner = msg.sender;
8     }
9
10    function address_show() public view returns(address){
11        return owner;
12    }
13
14 }
15

```

The console shows the execution of the contract. The first transaction is 'creation of address\_save pending...'. The second transaction is '[vm] from: 0x4B2...C02db to: address\_save.(constructor) value: 0 wei data: 0x688...c0029 logs: 0 hash: 0x1ca...38b3c'. The third transaction is 'call to address\_save.address\_show'. The console also shows the 'Low level interactions' section with 'CALLDATA' and a 'Transact' button.

c) Withdrawing funds from the contract to an account :

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible. The 'ENVIRONMENT' is set to 'JavaScript VM'. The 'ACCOUNT' is '0x4B2...C02db (89.999999%)'. The 'GAS LIMIT' is '3000000'. The 'VALUE' is '20 ether'. The 'CONTRACT' is 'FC1 - browser/Transfer\_funds.sol'. The 'Deploy' button is highlighted. Below it, 'Publish to IPFS' is an option. The 'At Address' section shows 'Load contract from Address'. The 'Transactions recorded' section shows '5' transactions. The 'Deployed Contracts' section shows 'BC1 AT 0xAc4...3510D (MEMORY)' and 'SMART AT 0x9EC...DDE84 (MEMORY)'. The 'FC1 AT 0x688...58CF7 (MEMORY)' is selected. The 'address\_show' button is highlighted with a red arrow. The 'Low level interactions' section shows 'CALLDATA' and a 'Transact' button. The main editor shows the Solidity code for 'FC1'.

```

1 pragma solidity^0.5.0;
2
3 contract FC1{
4
5     address owner;
6     constructor() public{
7         owner = msg.sender;
8     }
9
10    function receiveDeposit() payable public{
11
12    }
13
14    function getBalance() public view returns(uint){
15        return address(this).balance;
16    }
17
18    function withdraw(uint funds) public {
19        msg.sender.transfer(funds);
20    }
21 }
22

```

The console shows the execution of the contract. The first transaction is 'creation of FC1 pending...'. The second transaction is '[vm] from: 0x4B2...C02db to: FC1.(constructor) value: 0 wei data: 0x688...68029 logs: 0 hash: 0x76d...b5c5'. The third transaction is 'call to address\_save.address\_show'. The console also shows the 'Low level interactions' section with 'CALLDATA' and a 'Transact' button.

d) Limiting the users' access to functions (Transfer some funds from the contract's balance to the owner). Only the contract's creator is permitted to withdraw.

```
1 pragma solidity ^0.5.0;
2
3 contract Fc1{
4
5     address owner;
6     constructor() public{
7         owner = msg.sender; // Helps to save the address of the owner(who gives the first call)
8     }
9
10    modifier only_owner(){
11        require(msg.sender == owner); // So that edit access will only be with owner
12    }
13
14    function receiveDeposit() payable public{
15
16    }
17
18    function getBalance() public view returns(uint){
19        return address(this).balance;
20    }
21
22    function withdraw(uint funds) public only_owner{ // Here after public "only_owner" calls and check whether the modifier is owner or not?
23        msg.sender.transfer(funds);
24    }
25 }
```

```
1 pragma solidity ^0.5.0;
2
3 contract Fc1{
4
5     address owner;
6     constructor() public{
7         owner = msg.sender; // Helps to save the address of the owner(who gives the first call)
8     }
9
10    modifier only_owner(){
11        require(msg.sender == owner); // So that edit access will only be with owner
12    }
13
14    function receiveDeposit() payable public{
15
16    }
17
18    function getBalance() public view returns(uint){
19        return address(this).balance;
20    }
21
22    function withdraw(uint funds) public only_owner{ // Here after public "only_owner" calls and check whether the modifier is owner or not?
23        msg.sender.transfer(funds);
24    }
25 }
```

Let's try to withdraw this ether from another address:

Remix - Ethereum IDE

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.3+commit.10d17f24.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM

ACCOUNT

0x4B2...C02db (89.999999%)

GAS LIMIT

3000000

VALUE

0 ether

CONTRACT

Fc1 - browser/Owner.sol

Deploy

Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded

Deployed Contracts

BC1 AT 0xACA...3510D (MEMORY)

SMART AT 0x9EC...DDE84 (MEMORY)

```
1 pragma solidity^0.5.0;
2
3 contract Fc1{
4
5     address owner;
6     constructor() public{
7         owner = msg.sender; // Helps to save the address of the owner(who gives the first call)
8     }
9
10    modifier only_owner(){ // So that edit access will only be with owner
11        require(msg.sender == owner);
12    }
13
14    function receiveDeposit() payable public{
15
16    }
17
18    function getBalance() public view returns(uint){
19        return address(this).balance;
20    }
21
22    function withdraw(uint funds) public only_owner{ // Here after public "only_owner" calls and check whether the modifier is owner or not?
23        msg.sender.transfer(funds);
24    }
25 }
```

Execution cost: 601 gas FunctionDefinition getBalance 0 reference(s)

listen on network Search with transaction hash or address

transaction to Fc1.receiveDeposit pending ...

[vm] from: 0xAb8...35cb2 to: Fc1.receiveDeposit() 0x843...40486 value: 1800000000000000 wei data: 0xe36...7ef5e logs: 0 hash: 0x4df...3c7cb Debug

call to Fc1.getBalance

[call] from: 0xAb8483f64d9C6d1FcF9b849a677d03315835cb2 to: Fc1.getBalance() data: 0x120...65fe0 Debug

Remix - Ethereum IDE

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.3+commit.10d17f24.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM

ACCOUNT

0x4B2...C02db (89.999999%)

GAS LIMIT

3000000

VALUE

0 ether

CONTRACT

Fc1 - browser/Owner.sol

Deploy

Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded

Deployed Contracts

BC1 AT 0xACA...3510D (MEMORY)

SMART AT 0x9EC...DDE84 (MEMORY)

ADDRESS\_SAVE AT 0x688...58CF7 (MEMORY)

FC1 AT 0x929...5447E (MEMORY)

FC1 AT 0x843...40406 (MEMORY)

receiveDeposit

withdraw

funds: 10000000000000000000

transaction

getBalance

0 uint256: 1000000000000000000000000

Low level interactions

CALLDATA

Transaction

```
1 pragma solidity^0.5.0;
2
3 contract Fc1{
4
5     address owner;
6     constructor() public{
7         owner = msg.sender; // Helps to save the address of the owner(who gives the first call)
8     }
9
10    modifier only_owner(){ // So that edit access will only be with owner
11        require(msg.sender == owner);
12    }
13
14    function receiveDeposit() payable public{
15
16    }
17
18    function getBalance() public view returns(uint){
19        return address(this).balance;
20    }
21
22    function withdraw(uint funds) public only_owner{ // Here after public "only_owner" calls and check whether the modifier is owner or not?
23        msg.sender.transfer(funds);
24    }
25 }
```

Execution cost: 601 gas FunctionDefinition getBalance 0 reference(s)

listen on network Search with transaction hash or address

transaction to Fc1.withdraw pending ...

[call] from: 0xAb8483f64d9C6d1FcF9b849a677d03315835cb2 to: Fc1.getBalance() data: 0x120...65fe0 Debug

transaction to Fc1.withdraw failed ...

[vm] from: 0x4B2...C02db to: Fc1.withdraw(uint256) 0x843...40486 value: 0 wei data: 0x2e1...40000 logs: 0 hash: 0xe2f...c0905 Debug

transaction to Fc1.withdraw errored: VM error: revert. revert The transaction has been reverted to the initial state. Note: The called function should be payable if you send value and the value you send should be less than your current balance. Debug the transaction to get more information.

Only owner can withdraw.