

# New Ether Wallet Results

Priyanka Reddy - 24th March 2023

# Deploy Contract

Owner account balance after deploying contract -

The screenshot shows the Remix IDE interface with the following details:

- Environment:** Set to "Remix VM (Berlin)".
- Account:** Shows a list of accounts with their addresses and ether balances, all containing 100 ether.
- Deployed Contracts:** A list of deployed contracts, with one entry for "NEWETHERWALLET AT 0xD91...3913E".
- Code Editor:** Displays the Solidity code for the "NewEtherWallet" contract, which implements a basic ether wallet with owner-only withdraw functionality.
- Output/Console:** Shows the deployment transaction details and the resulting contract address (0x5B3...eddC4) with its balance of 99.999999999999532695 ether.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Deploy Contract

Base state after deploying contract -

The screenshot shows the Truffle UI interface for deploying and running transactions. On the left, there's a sidebar with various icons for deployment, monitoring, and configuration. The main area has tabs for 'NewEtherWallet.sol' (selected) and 'NewEtherWallet\_v2.sol'. The code editor displays the Solidity source code for the 'NewEtherWallet' contract. The code defines a payable owner and receiver, a constructor that sets the owner, a receive function to handle ether, and withdraw and addReceiver functions. The 'Deploy & Run Transactions' panel on the left shows a 'Deploy' button, a checkbox for 'PUBLISH TO IPFS', and a dropdown for 'At Address' with a 'Load contract from Address' option. Below this are sections for 'Transactions recorded' (1) and 'Deployed Contracts'. The 'Deployed Contracts' section shows a deployed instance named 'NEWETHERWALLET AT 0xD91...39' with a balance of 0 ETH. It includes buttons for 'addReceiver', 'send', 'withdraw', 'getBalance', and 'owner'. At the bottom, there are sections for 'Low level interactions' and 'CALLDATA' with a 'Transact' button.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{

    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

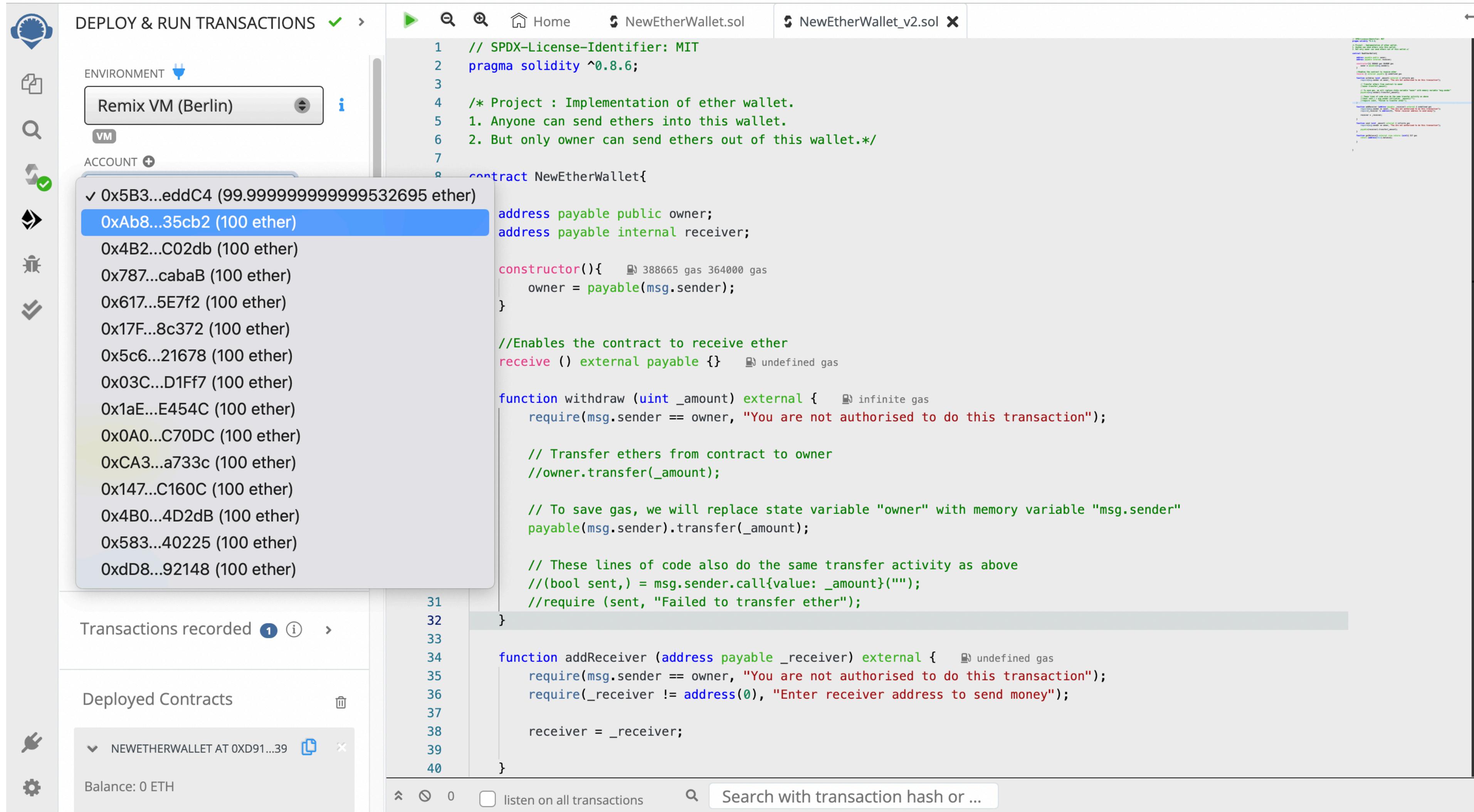
        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Receive Money

## Step 1 - Select a different user address



The screenshot shows the Remix IDE interface with the following details:

- Deploy & Run Transactions:** Shows a green checkmark and a green arrow icon.
- Environment:** Set to "Remix VM (Berlin)".
- Account:** A dropdown menu lists several Ethereum addresses with their balances:
  - ✓ 0x5B3...eddC4 (99.99999999999532695 ether)
  - 0xAB8...35cb2 (100 ether)** (highlighted in blue)
  - 0x4B2...C02db (100 ether)
  - 0x787...cabab (100 ether)
  - 0x617...5E7f2 (100 ether)
  - 0x17F...8c372 (100 ether)
  - 0x5c6...21678 (100 ether)
  - 0x03C...D1Ff7 (100 ether)
  - 0x1aE...E454C (100 ether)
  - 0x0A0...C70DC (100 ether)
  - 0xCA3...a733c (100 ether)
  - 0x147...C160C (100 ether)
  - 0x4B0...4D2dB (100 ether)
  - 0x583...40225 (100 ether)
  - 0xdD8...92148 (100 ether)
- Transactions recorded:** 1 transaction recorded.
- Deployed Contracts:** NEWETHERWALLET AT 0xD91...39
- Balance:** 0 ETH
- Code Area:** Displays the Solidity code for the NewEtherWallet\_v2 contract.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Receive Money

Step 2 - Enter amount and unit in the shown section.

The screenshot shows the Remix IDE interface. On the left, there's a sidebar with various icons for deploying, running, and managing contracts. The main area has tabs for 'NewEtherWallet.sol' (which is active) and 'NewEtherWallet\_v2.sol'. The code editor contains the following Solidity code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

On the left side of the main area, there are sections for 'ENVIRONMENT' (set to 'Remix VM (Berlin)'), 'ACCOUNT' (set to '0xAb8...35cb2 (100 eth)'), 'GAS LIMIT' (set to '3000000'), and a dropdown for 'Enter an amount and choose its unit' (set to '10 Ether'). Below these is a 'CONTRACT (Compiled by Remix)' section with a dropdown set to 'NewEtherWallet - priyanka/'. There are buttons for 'Deploy' and 'PUBLISH TO IPFS'. Below that is an 'OR' section with a 'At Address' button and a 'Load contract from Address' input field. At the bottom, there are sections for 'Transactions recorded' (1), 'Deployed Contracts' (listing 'NEWETHERWALLET AT 0xD91...39'), and a 'Balance: 0 ETH' status.

# Receive Money

Step 3 - Hit the 'Transact' button.

The screenshot shows the Truffle UI interface. On the left, there's a sidebar with various icons. The main area has tabs for 'NewEtherWallet.sol' and 'NewEtherWallet\_v2.sol'. The code editor on the right contains the following Solidity code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

The left panel shows a 'Transactions recorded' section with one entry and a 'Deployed Contracts' section listing 'NEWETHERWALLET AT 0xD91...39'. Below this is a detailed view of the deployed contract with fields for 'addReceiver', 'send', 'withdraw', 'getBalance', and 'owner'. At the bottom of this panel is a 'Transact' button. The bottom navigation bar includes a 'Transact' button, a search bar, and a checkbox for 'listen on all transactions'.

# Receive Money

Result - Amount is deducted from sender's account

The screenshot shows the Remix IDE interface with the following details:

- Deploy & Run Transactions:** Shows a green checkmark and a green arrow icon.
- Environment:** Set to "Remix VM (Berlin)".
- Contract:** NewEtherWallet.sol (selected)
- Code:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        //((bool sent,) = msg.sender.call{value: _amount}(""));

        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```
- Transactions recorded:** 2 transactions recorded.
- Deployed Contracts:** NEWETHERWALLET AT 0xD91...39
- Balance:** 10 ETH

# Receive Money

Result - The amount is successfully transferred into the wallet, as shown in balance.

The screenshot shows the Truffle UI interface. On the left, there's a sidebar with various icons for deploying contracts, monitoring transactions, and interacting with deployed contracts. The main area is split into two panes: 'DEPLOY & RUN TRANSACTIONS' on the left and the code editor on the right.

**DEPLOY & RUN TRANSACTIONS** pane:

- At Address:** A dropdown menu where 'Load contract from Address' is selected.
- Transactions recorded:** Shows 2 recorded transactions.
- Deployed Contracts:** Shows a deployed contract named 'NEWETHERWALLET AT 0xD91...39'.
  - Balance:** 10 ETH.
  - Interactions:** Buttons for 'addReceiver', 'send', 'withdraw', and 'getBalance'.
  - Owner:** Address: 0x5B38Da6a701c568545 dCfcB03FcB875f56beddC4.
- Low level interactions:** A section for 'CALLDATA' with a 'Transact' button.

**Code Editor (NewEtherWallet.sol) pane:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{

    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        //((bool sent,) = msg.sender.call{value: _amount}(""));

        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

At the bottom of the code editor, there are search and filter options: 'listen on all transactions' and 'Search with transaction hash or ...'.

# Withdraw Money

Step 1 - Enter the amount to withdraw (default unit - Wei) and hit 'withdraw' button.

Result - Amount successfully deducted from wallet, as shown in balance.

The screenshot shows the Truffle UI interface for interacting with a Ethereum smart contract named `NewEtherWallet.sol`. The left sidebar contains icons for deploying, running transactions, and monitoring deployed contracts. The main area displays the Solidity code for the `NewEtherWallet` contract, which includes methods for adding receivers, sending ether, and withdrawing ether. A specific transaction is selected, showing its details: it's a withdrawal of 5000000000000000000 Wei (5 ETH) from the contract to the owner. The transaction status is shown as "Pending". The right side of the interface shows the transaction history and deployment details for the contract.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Withdraw Money

Result - Amount successfully transferred into owner account.

The screenshot shows the Remix IDE interface with the following details:

- Deploy & Run Transactions:** Shows a green checkmark indicating success.
- Environment:** Set to "Remix VM (Berlin)".
- Account:** A list of deployed accounts with their addresses and ether balances:
  - ✓ 0x5B3...eddC4 (104.999999999999501955 ether)
  - 0xAb8...35cb2 (89.99999999999978945 ether)
  - 0x4B2...C02db (100 ether)
  - 0x787...cabaB (100 ether)
  - 0x617...5E7f2 (100 ether)
  - 0x17F...8c372 (100 ether)
  - 0x5c6...21678 (100 ether)
  - 0x03C...D1Ff7 (100 ether)
  - 0x1aE...E454C (100 ether)
  - 0x0A0...C70DC (100 ether)
  - 0xCA3...a733c (100 ether)
  - 0x147...C160C (100 ether)
  - 0x4B0...4D2dB (100 ether)
  - 0x583...40225 (100 ether)
  - 0xdD8...92148 (100 ether)
- Transactions recorded:** 3 transactions recorded.
- Deployed Contracts:** NEWETHERWALLET AT 0xD91...39
- Balance:** 5 ETH
- Code:** The deployed contract code is displayed in the main editor area.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Send Money

Step 1 - Select a user account to send money. Copy the address.

The screenshot shows the Truffle UI interface for deploying and running transactions. On the left, there's a sidebar with various icons for deployment, monitoring, and configuration. The main area has tabs for "DEPLOY & RUN TRANSACTIONS" (active) and "NewEtherWallet.sol".

In the "DEPLOY & RUN TRANSACTIONS" tab, there's a list of accounts with their addresses and ether balances:

- 0x5B3...eddC4 (104.99999999999501955 ether)
- 0xAb8...35cb2 (89.99999999999978945 ether)
- 0x4B2...C02db (100 ether)
- ✓ 0x787...cabAB (100 ether)** (selected)
- 0x617...5E7f2 (100 ether)
- 0x17F...8c372 (100 ether)
- 0x5c6...21678 (100 ether)
- 0x03C...D1Ff7 (100 ether)
- 0x1aE...E454C (100 ether)
- 0x0A0...C70DC (100 ether)
- 0xCA3...a733c (100 ether)
- 0x147...C160C (100 ether)
- 0x4B0...4D2dB (100 ether)
- 0x583...40225 (100 ether)
- 0xdD8...92148 (100 ether)

Below this list, there's an "OR" section with two options:

- At Address** (selected)
- Load contract from Address

Further down, there are sections for "Transactions recorded" (with 3 items), "Deployed Contracts" (listing "NEWETHERWALLET AT 0xD91...39"), and a balance indicator "Balance: 5 ETH".

The right side of the screen displays the Solidity code for the NewEtherWallet contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

project : Implementation of ether wallet.
anyone can send ethers into this wallet.
but only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        // (bool sent,) = msg.sender.call{value: _amount}("");
        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Send Money

Step 2 - Paste the address in 'addReceiver' section and hit that button.

The screenshot shows the Truffle UI interface. On the left, there's a sidebar with various icons. The main area has tabs for 'DEPLOY & RUN TRANSACTIONS' (selected) and 'NewEtherWallet.sol' (active). The 'Transactions recorded' section shows a deployed contract named 'NEWETHERWALLET AT 0xD91...39' with a balance of 5 ETH. The 'addReceiver' field is highlighted with a red box and contains the address '0x78731D3Ca6b7E34aC0F824c4'. Below it are fields for 'send' (amount: uint256 \_amount) and 'withdraw' (amount: 50000000000000000000). There are also buttons for 'getBalance' and 'owner'. The right side of the screen shows the Solidity code for the 'NewEtherWallet' contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{

    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        //((bool sent,) = msg.sender.call{value: _amount}(""));

        //require (sent, "Failed to transfer ether");
    }

    function addReceiver (address payable _receiver) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");
        require(_receiver != address(0), "Enter receiver address to send money");

        receiver = _receiver;
    }
}
```

# Send Money

Step 3 - Enter the amount to be sent (default unit - Wei), and hit send button.

The screenshot shows the Truffle UI interface for interacting with a deployed Ethereum contract. On the left, there's a sidebar with various icons for deployment, monitoring, and configuration. The main area is divided into sections:

- DEPLOY & RUN TRANSACTIONS**: Shows a green checkmark and a progress bar. Below it, there are buttons for "At Address" and "Load contract from Address".
- Transactions recorded**: Displays 4 recorded transactions.
- Deployed Contracts**: Shows a list of deployed contracts, with "NEWETHERWALLET AT 0xD91...39" expanded. It displays the contract's balance (5 ETH) and several interaction methods:
  - addReceiver**: Address 0x78731D3Ca6b7E34aC0F824c4
  - send**: Value 500000000000000000 (uint256 \_amount)
  - withdraw**: Value 500000000000000000
  - getBalance**: Value 0: uint256: 50000000000000000000000000000000
  - owner**: Address 0x5B38Da6a701c568545 dCfcB03FcB875f56beddC4
- Low level interactions**: A section for interacting with the contract at a lower level, including a "Transact" button.
- Code Editor**: The right pane shows the Solidity code for the `NewEtherWallet` contract. The `send` function is highlighted with a blue selection bar.
- Logs**: At the bottom, it shows a log entry for a transaction: "[vm] from: 0x5B3...eddC4 to: NewEtherWallet.addReceiver(address) 0xd91...39138 value: 0 wei data: 0x69d...cabab logs: 0 hash: 0x339...4d033".

# Send Money

Result - Amount successfully deducted from wallet, as shown in balance.

The screenshot shows the Truffle UI interface for interacting with a Ethereum smart contract named `NewEtherWallet_v2.sol`. The left sidebar contains icons for deploying, running transactions, viewing deployed contracts, and low-level interactions. The main area has tabs for `Home`, `NewEtherWallet.sol`, and `NewEtherWallet_v2.sol`.

**Contract Deployment & Run Transactions:**

- Deployed Contracts:** Shows a deployed contract named `NEWETHERWALLET AT 0xD91...39` with a balance of 0 ETH.
- Transactions recorded:** Shows 5 recorded transactions.
- Low level interactions:** Shows a `Transact` button for interacting with the contract.

**Contract Code (`NewEtherWallet_v2.sol`):**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

/* Project : Implementation of ether wallet.
1. Anyone can send ethers into this wallet.
2. But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{

    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    //Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        //owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);

        // These lines of code also do the same transfer activity as above
        //((bool sent,) = msg.sender.call{value: _amount}(""));

        //require (sent, "Failed to transfer ether");
    }
}
```

**Call Log:**

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: NewEtherWallet.getBalance() data: 0x120...65fe0
```

**Debug:** A `Debug` button is located at the bottom right of the call log.

# Send Money

Result - Amount successfully added into receiver account.

The screenshot shows the Truffle UI interface for deploying and running transactions. The main window displays the Solidity code for the `NewEtherWallet` contract. The code includes a constructor that sets the `owner` to the msg.sender and a `withdraw` function that requires the sender to be the owner before transferring ether to the owner's address. A transaction history panel on the left lists various ether transfers, with one transaction highlighted: `✓ 0x787...cabAB (105 ether)`. Below this, there are options to "At Address" or "Load contract from Address". The bottom section shows the deployed contracts, with `NEWETHERWALLET AT 0xD91...39` listed. The balance for this contract is shown as 0 ETH. A transaction log at the bottom indicates a `CALL` from address `0x5B38Da6a701c568545dCfcB03FcB875f56beddC4` to the `getBalance()` function of the deployed contract, returning a data value of `0x120...65fe0`.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

// Project : Implementation of ether wallet.
// Anyone can send ethers into this wallet.
// But only owner can send ethers out of this wallet.*/

contract NewEtherWallet{
    address payable public owner;
    address payable internal receiver;

    constructor() {
        owner = payable(msg.sender);
    }

    // Enables the contract to receive ether
    receive () external payable {}

    function withdraw (uint _amount) external {
        require(msg.sender == owner, "You are not authorised to do this transaction");

        // Transfer ethers from contract to owner
        // owner.transfer(_amount);

        // To save gas, we will replace state variable "owner" with memory variable "msg.sender"
        payable(msg.sender).transfer(_amount);
    }
}
```

Thank You !