Here is a simple contract that you can get, increment and decrement the count store in this contract.

*// SPDX-License-Identifier: MIT*

pragma solidity ^0.8.17;

contract Counter {

uint public count;

*// Function to get the current count*

function get() public view returns (uint) {

return count;

}

*// Function to increment count by 1*

function inc() public {

count += 1;

}

*// Function to decrement count by 1*

function dec() public {

*// This function will fail if count = 0*

count -= 1;

}

}

**Gas Limit**

There are 2 upper bounds to the amount of gas you can spend

* gas limit (max amount of gas you're willing to use for your transaction, set by you)
* block gas limit (max amount of gas allowed in a block, set by the network)

*// SPDX-License-Identifier: MIT*

pragma solidity ^0.8.17;

contract Gas {

uint public i = 0;

*// Using up all of the gas that you send causes your transaction to fail.*

*// State changes are undone.*

*// Gas spent are not refunded.*

function forever() public {

*// Here we run a loop until all of the gas are spent*

*// and the transaction fails*

while (true) {

i += 1;

}

}

}

## Ether Wallet

An example of a basic wallet.

* Anyone can send ETH.
* Only the owner can withdraw.

*// SPDX-License-Identifier: MIT*

pragma solidity ^0.8.17;

contract EtherWallet {

address payable public owner;

constructor() {

owner = payable(msg.sender);

}

receive() external payable {}

function withdraw(uint \_amount) external {

require(msg.sender == owner, "caller is not owner");

payable(msg.sender).transfer(\_amount);

}

function getBalance() external view returns (uint) {

return address(this).balance;

}

}