pragma solidity ^0.6.0;

contract Bank

{

mapping(address=> uint ) private \_balances;

address public owner;

event LogDepositeMade(address accountHoder, uint amount );

constructor () public

{

owner=msg.sender;

emit LogDepositeMade(msg.sender, 1000);

}

function deposite() public payable returns (uint)

{

require ((\_balances[msg.sender] + msg.value) > \_balances[msg.sender] && msg.sender!=address(0));

\_balances[msg.sender] += msg.value;

emit LogDepositeMade(msg.sender , msg.value);

return \_balances[msg.sender];

}

function withdraw (uint withdrawAmount) public returns (uint)

{

require (\_balances[msg.sender] >= withdrawAmount);

require(msg.sender!=address(0));

require (\_balances[msg.sender] > 0);

\_balances[msg.sender]-= withdrawAmount;

msg.sender.transfer(withdrawAmount);

emit LogDepositeMade(msg.sender , withdrawAmount);

return \_balances[msg.sender];

}

function viewBalance() public view returns (uint)

{

return \_balances[msg.sender];

}

}

pragma solidity ^0.8.0;

contract Bank {

mapping(address => uint) private \_balances;

address public owner;

event LogDepositMade(address accountHolder, uint amount);

event LogWithdrawalMade(address accountHolder, uint amount);

constructor() {

owner = msg.sender;

emit LogDepositMade(msg.sender, 1000); // Initialize owner with an example deposit (optional).

}

// Deposit function

function deposit() public payable returns (uint) {

require(msg.value > 0, "Deposit amount must be greater than 0");

\_balances[msg.sender] += msg.value; // Increase the sender's balance

emit LogDepositMade(msg.sender, msg.value); // Emit event for deposit

return \_balances[msg.sender]; // Return updated balance

}

// Withdraw function

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

require(\_balances[msg.sender] >= withdrawAmount, "Insufficient balance");

require(withdrawAmount > 0, "Withdrawal amount must be greater than 0");

uint balanceBefore = \_balances[msg.sender];

\_balances[msg.sender] -= withdrawAmount; // First, update the balance

payable(msg.sender).transfer(withdrawAmount); // Then, transfer funds

emit LogWithdrawalMade(msg.sender, withdrawAmount); // Emit event for withdrawal

return \_balances[msg.sender]; // Return updated balance

}

// View balance function

function viewBalance() public view returns (uint) {

return \_balances[msg.sender]; // Return the caller's balance

}

// Function to check the contract's balance (optional)

function contractBalance() public view returns (uint) {

return address(this).balance; // Return the contract's balance

}

}