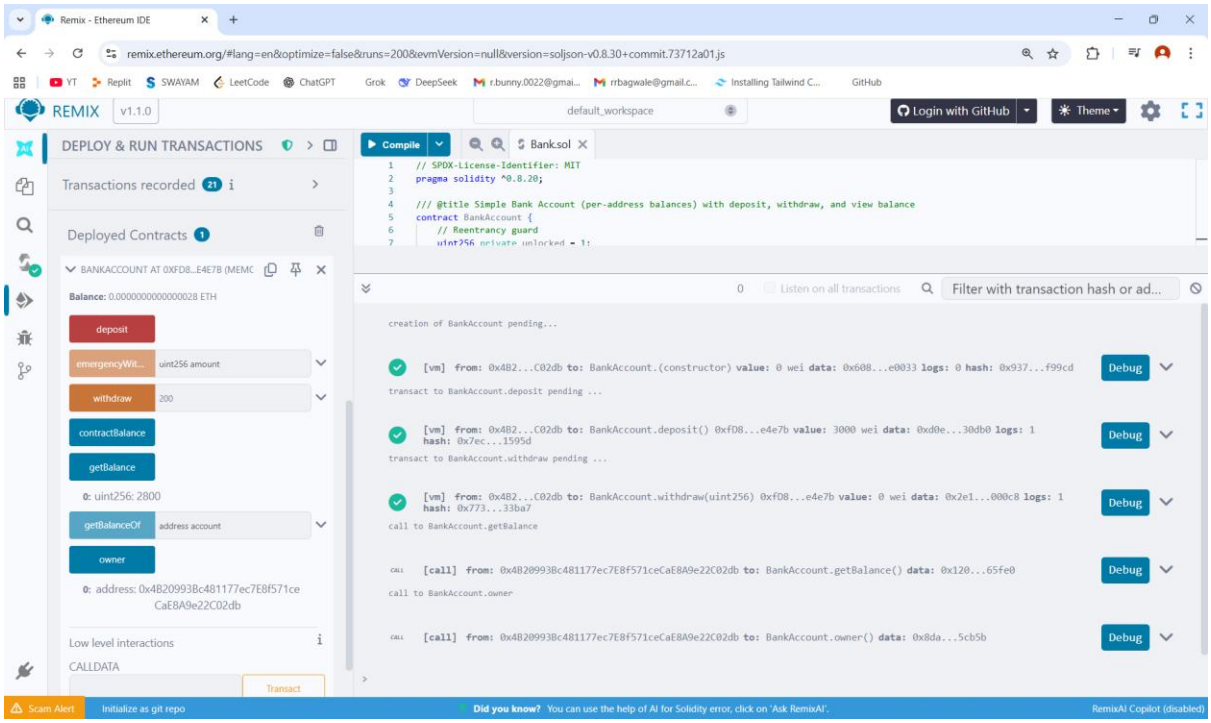
File : Bank.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

/// @title Simple Bank Account (per-address balances) with deposit, withdraw, and view balance
contract BankAccount {
    // Reentrancy guard
    uint256 private unlocked = 1;
    modifier nonReentrant() {
        require(unlocked == 1, "Reentrant call");
        unlocked = 0;
        _;
        unlocked = 1;
    }

    // Balances per user
    mapping(address => uint256) private balances;

    // Events
    event Deposited(address indexed account, uint256 amount);
    event Withdrawn(address indexed account, uint256 amount);
    event EmergencyWithdrawal(address indexed admin, uint256 amount);

    // Admin (owner) to allow emergency withdrawal if needed
    address public owner;

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

```
/// @notice Deposit ETH into your bank account
/// @dev msg.value is added to caller's balance
function deposit() external payable {
    require(msg.value > 0, "Must send ETH to deposit");
    balances[msg.sender] += msg.value;
    emit Deposited(msg.sender, msg.value);
}

/// @notice Withdraw `amount` wei from your account to your address
/// @param amount Amount in wei to withdraw
function withdraw(uint256 amount) external nonReentrant {
    require(amount > 0, "Amount must be > 0");
    uint256 bal = balances[msg.sender];
    require(bal >= amount, "Insufficient balance");

    // Effects first
    balances[msg.sender] = bal - amount;

    // Interaction
    (bool sent, ) = payable(msg.sender).call{value: amount}("");
    require(sent, "Transfer failed");

    emit Withdrawn(msg.sender, amount);
}

/// @notice View your balance (in wei)
function getBalance() external view returns (uint256) {
    return balances[msg.sender];
}
```

```
    /// @notice View balance of any account (publicly readable)
    /// @param account address to query
    function getBalanceOf(address account) external view returns (uint256) {
        return balances[account];
    }

    /// @notice Fallback receive function — accepts ETH but credits it to sender's balance
    receive() external payable {
        require(msg.value > 0, "Must send ETH");
        balances[msg.sender] += msg.value;
        emit Deposited(msg.sender, msg.value);
    }

    /// @notice Emergency function: owner can withdraw accidental ETH stored not mapped to users
    /// @dev Use sparingly; real banks would avoid this or implement more checks
    function emergencyWithdraw(uint256 amount) external {
        require(msg.sender == owner, "Only owner");
        (bool sent, ) = payable(owner).call{value: amount}("");
        require(sent, "Transfer failed");
        emit EmergencyWithdrawal(owner, amount);
    }

    /// @notice Convenience: total funds held in contract
    function contractBalance() external view returns (uint256) {
        return address(this).balance;
    }
}
```


Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.30+commit.73712a01.js
REMIX
v1.1.0
default_workspace
Login with GitHub
Theme
DEPLOY & RUN TRANSACTIONS
Transactions recorded 21
Deployed Contracts 1
BANKACCOUNT AT 0XFD8...E4E7B (MEMC
Balance: 0.000000000000000028 ETH
deposit
emergencyWit...
uint256 amount
withdraw
200
contractBalance
getBalance
0: uint256: 2800
getBalanceOf
address account
owner
0: address: 0x4B20993Bc481177ec7E8f571ce CaE8A9e22C02db
Low level interactions
CALLDATA
Transact
Compile
Bank.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
/// @title Simple Bank Account (per-address balances) with deposit, withdraw, and view balance
contract BankAccount {
// Reentrancy guard
uint256 private unlocked = 1;
Listen on all transactions
Filter with transaction hash or ad...
creation of BankAccount pending...
[vm] from: 0x4B2...C02db to: BankAccount.(constructor) value: 0 wei data: 0x608...e0033 logs: 0 hash: 0x937...f99cd
Debug
transact to BankAccount.deposit pending ...
[vm] from: 0x4B2...C02db to: BankAccount.deposit() 0xfD8...e4e7b value: 3000 wei data: 0xd0e...30db0 logs: 1 hash: 0x7ec...1595d
Debug
transact to BankAccount.withdraw pending ...
[vm] from: 0x4B2...C02db to: BankAccount.withdraw(uint256) 0xfD8...e4e7b value: 0 wei data: 0x2e1...000c8 logs: 1 hash: 0x773...33ba7
Debug
call to BankAccount.getBalance
[call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: BankAccount.getBalance() data: 0x120...65fe0
Debug
call to BankAccount.owner
[call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: BankAccount.owner() data: 0x8da...5cb5b
Debug
Scam Alert
Initialize as git repo
Did you know? You can use the help of AI for Solidity error, click on 'Ask RemixAI'.
RemixAI Copilot (disabled)