Banking.sol

// SPDX-License-Identifier: MIT

// Declare the Solidity version to be used

pragma solidity ^0.8.0;

contract BankAccount {

// Mapping to store the balance of each account (address)

mapping(address => uint) private balances;

// Event emitted when a deposit is made

event Deposited(address indexed account, uint amount);

// Event emitted when a withdrawal is made

event Withdrawn(address indexed account, uint amount);

// Function to deposit Ether into the account

function deposit() public payable {

// Ensure the deposit amount is greater than 0

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

// Increase the sender's balance by the deposited amount

balances[msg.sender] += msg.value;

// Emit a Deposited event with the sender's address and amount deposited

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

}

// Function to withdraw Ether from the account

function withdraw(uint \_amount) public {

// Ensure the requested withdrawal amount does not exceed the balance

require(\_amount <= balances[msg.sender], "Insufficient balance");

// Deduct the withdrawal amount from the sender's balance

balances[msg.sender] -= \_amount;

// Transfer the specified amount to the sender's address

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

// Emit a Withdrawn event with the sender's address and amount withdrawn

emit Withdrawn(msg.sender, \_amount);

}

// Function to get the balance of the sender's account

function getBalance() public view returns (uint) {

// Return the balance of the sender

return balances[msg.sender];

}

}