// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Bank {

    address public accHolder;

    uint256 balance = 0;

    constructor() {

        accHolder = msg.sender;

    }

    function withdraw() payable public {

        require(msg.sender == accHolder, "You are not the account owner");

        require(balance > 0, "You don't have enough balance.");

        payable(msg.sender).transfer(balance);

        balance = 0;

    }

    function deposit() public payable {

        require(msg.sender == accHolder, "You are not the account owner");

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

        balance += msg.value;

    }

    function showBalance() public view returns(uint) {

        require(msg.sender == accHolder, "You are not the account owner");

        return balance;

    }

}

// SPDX-License-Identifier: MIT  
// Specifies the license for the contract to ensure open source compliance (MIT License)  
  
pragma solidity ^0.8.0;  
// Declares the Solidity compiler version to be 0.8.x  
  
contract Bank {  
    // Defines a contract named 'Bank'  
     
    address public accHolder;  
    // Declares a public variable of type 'address' to store the account holder's address  
     
    uint256 balance = 0;  
    // Declares a private variable 'balance' of type 'uint256' initialized to 0  
  
    constructor() {  
        // Constructor function that runs once when the contract is deployed  
        accHolder = msg.sender;  
        // Sets the account holder's address to the address that deployed the contract  
    }  
  
    function withdraw() payable public {  
        // Function to withdraw the entire balance  
        require(msg.sender == accHolder, "You are not the account owner");  
        // Checks that the caller is the account holder  
        require(balance > 0, "You don't have enough balance.");  
        // Ensures that there is a positive balance available to withdraw  
         
        payable(msg.sender).transfer(balance);  
        // Transfers the entire balance to the account holder  
        balance = 0;  
        // Sets the balance to 0 after transfer  
    }  
  
    function deposit() public payable {  
        // Function to deposit Ether into the contract  
        require(msg.sender == accHolder, "You are not the account owner");  
        // Checks that only the account holder can deposit funds  
        require(msg.value > 0, "Deposit amount should be greater than 0.");  
        // Ensures that the deposit amount is greater than 0  
  
        balance += msg.value;  
        // Increases the balance by the deposited amount  
    }  
  
    function showBalance() public view returns(uint) {  
        // Function to view the current balance  
        require(msg.sender == accHolder, "You are not the account owner");  
        // Checks that only the account holder can view the balance  
  
        return balance;  
        // Returns the current balance  
    }  
}

Practical 4: Student data

// SPDX-License-Identifier: Bhide License

pragma solidity ^0.8.0;

contract StudentRegistry {

    // Define a structure to hold student details

    struct Student {

        string name;  // Name of the student

        uint256 age;  // Age of the student

    }

    // Array to store student records

    Student[] private students;

    // Event to log when a student is added

    event StudentAdded(string name, uint256 age);

    // Event to log the received Ether value

    event ReceivedEther(address indexed sender, uint256 value);

    // Function to receive Ether directly

    receive() external payable {

        // Log the received Ether value

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

    }

    // Fallback function to receive Ether and log it

    fallback() external payable {

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

    }

    // Function to add a new student to the registry

    function addStudent(string memory name, uint256 age) public {

        students.push(Student(name, age));

        emit StudentAdded(name, age);  // Emit event for added student

    }

    // Function to retrieve a student&#39;s details by index

    function getStudent(uint256 index) public view returns (string memory, uint256) {

        require(index &lt; students.length, &quot;Student not found&quot;);

        return (students[index].name, students[index].age);

    }

    // Function to get the total count of students in the registry

    function getStudentCount() public view returns (uint256) {

        return students.length;

    }

}

// SPDX-License-Identifier: Bhide License

pragma solidity ^0.8.0;

contract StudentRegistry {

    // Define a structure to hold student details

    struct Student {

        string name;  // Name of the student

        uint256 age;  // Age of the student

    }

    // Array to store student records

    Student[] private students;

    // Event to log when a student is added

    event StudentAdded(string name, uint256 age);

    // Event to log the received Ether value

    event ReceivedEther(address indexed sender, uint256 value);

    // Function to receive Ether directly

    receive() external payable {

        // Log the received Ether value

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

    }

    // Fallback function to receive Ether and log it

    fallback() external payable {

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

    }

    // Function to add a new student to the registry

    function addStudent(string memory name, uint256 age) public {

        students.push(Student(name, age));

        emit StudentAdded(name, age);  // Emit event for added student

    }

    // Function to retrieve a student's details by index

    function getStudent(uint256 index) public view returns (string memory, uint256) {

        require(index < students.length, "Student not found");

        return (students[index].name, students[index].age);

    }

    // Function to get the total count of students in the registry

    function getStudentCount() public view returns (uint256) {

        return students.length;

    }

}