Bank Code:

// SPDX-License-Identifier: MIT

// Specifies the license under which the code is shared

pragma solidity ^0.8.0;

// Specifies the Solidity compiler version to be used

// Contract definition

contract Bank {

address public accHolder; // Public variable to store the account holder's address

uint256 balance = 0; // Variable to store the balance of the account, initialized to 0

// Constructor function, executed once when the contract is deployed

constructor() {

accHolder = msg.sender; // Sets the account holder's address to the address deploying the contract

}

// Function to withdraw a specified amount from the account balance

function withdraw(uint256 amount) public {

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

// Ensures that only the account holder can withdraw funds

require(amount > 0, "Withdrawal amount should be greater than 0.");

// Ensures the withdrawal amount is positive

require(amount <= balance, "Insufficient balance.");

// Checks that the withdrawal amount does not exceed the available balance

balance -= amount; // Reduces the balance by the withdrawal amount

payable(msg.sender).transfer(amount); // Transfers the specified amount to the account holder’s address

}

// Function to deposit funds into the account

function deposit() public payable {

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

// Ensures that only the account holder can deposit funds

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

// Ensures the deposit amount is positive

balance += msg.value; // Adds the deposited amount to the account balance

}

// Function to view the current balance

function showBalance() public view returns(uint256) {

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

// Ensures that only the account holder can view the balance

return balance; // Returns the current account balance

}

}

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'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;

    }

}