

Experiment No 4

Title:

Write a program in solidity to create Student data. Use the following constructs:

- Structs
- Arrays
- Fallbacks

Procedure:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0;

contract Student_management {
    struct Student {
        int stud_id;
        string Name;
        string Department;
    }

    Student[] Students;

    function add_stud(int stud_id, string memory Name, string memory
Department) public {
        Student memory stud = Student(stud_id, Name, Department);
        Students.push(stud);
    }

    function getStudent(int stud_id) public view returns (string memory,
string memory) {
        for (uint i = 0; i < Students.length; i++) {
            Student memory stud = Students[i];
            if (stud.stud_id == stud_id) {
                return (stud.Name, stud.Department);
            }
        }
        return ("Name Not Found", "Department Not Found");
    }

    // Receive function to handle incoming Ether transactions
    receive() external payable {
        // Handle incoming Ether transactions, if needed
    }
}
```

```
}
```

Certainly, I'll explain the provided Solidity code in detail, line by line:

```
// SPDX-License-Identifier: MIT
```

This line is a SPDX license identifier comment, which specifies that the code is licensed under the MIT License.

```
pragma solidity >=0.7.0;
```

This line is a compiler version pragma, indicating that the code is written for Solidity version 0.7.0 or higher.

```
contract Student management {
```

This line defines the beginning of a Solidity smart contract named `Student_management`. A smart contract is a self-executing contract with the terms of the agreement directly written in code.

```
struct Student {  
    int stud_id;  
    string Name;  
    string Department;  
}
```

This code defines a struct named `Student`, which represents the data structure for a student. The struct has three fields: `stud_id` (an integer), `Name` (a string), and `Department` (a string). It's used to store information about a student.

```
Student[] Students;
```

This declares a dynamic array called `Students` that will hold instances of the `Student` struct. It's used to store information about multiple students.

```
function add_stud(int stud_id, string memory Name, string memory
Department) public {
    Student memory stud = Student(stud_id, Name, Department);
    Students.push(stud);
}
```

This is a function named `add_stud` that allows you to add a new student to the `Students` array. It takes three parameters: `stud_id` (an integer), `Name` (a string), and `Department` (a string). Inside the function, it creates a `Student` struct named `stud` and initializes it with the provided parameters. Then, it pushes this `stud` struct into the `Students` array.

```
function getStudent(int stud_id) public view returns (string memory,
string memory) {
    for (uint i = 0; i < Students.length; i++) {
        Student memory stud = Students[i];
        if (stud.stud_id == stud_id) {
            return (stud.Name, stud.Department);
        }
    }
    return ("Name Not Found", "Department Not Found");
}
```

This is a function named `getStudent`, which is used to retrieve the name and department of a student given their `stud_id`. It takes a single parameter `stud_id`, searches through the `Students` array to find a matching student, and returns the name and department if found. If no matching student is found, it returns default values "Name Not Found" and "Department Not Found."

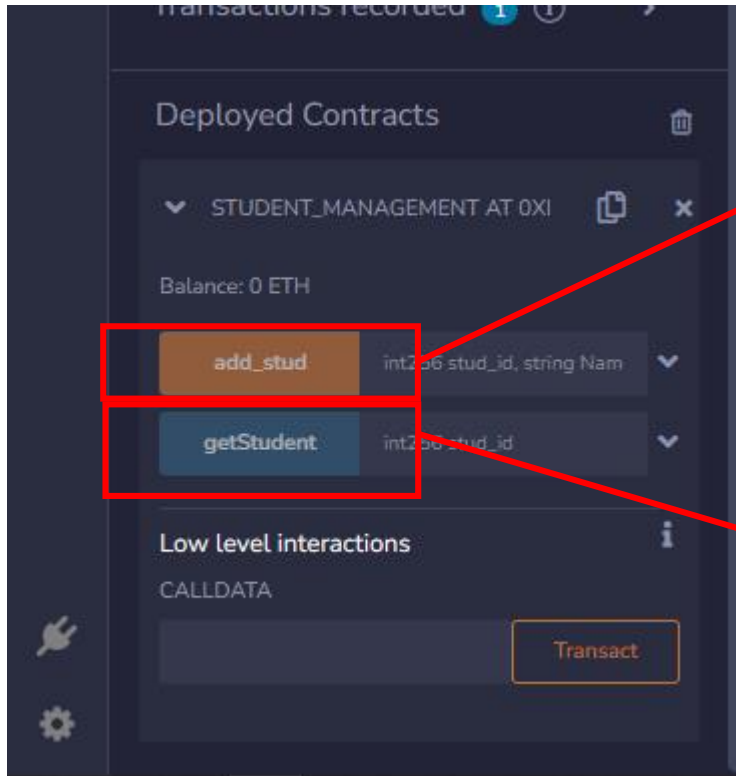
```
receive() external payable {
    // Handle incoming Ether transactions, if needed
}
```

This is a receive function, which is used to handle incoming Ether transactions. In this contract, it's currently empty and does not perform any specific actions when Ether is sent to the contract. You can add custom logic to handle incoming Ether transactions here if needed.

Overall, this contract allows you to add and retrieve information about students on the Ethereum blockchain. Students are stored as struct instances

in an array, and you can look up their information by providing their unique `stud_id`.

Interact With Code:



Deployed Contracts

STUDENT_MANAGEMENT AT 0XI

Balance: 0 ETH

add_stud int256 stud_id, string Nam

getStudent int256 stud_id

Low level interactions

CALLDATA

Transact

We can add student just by filling the details

Get the details just by using student ID