Bank:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8;

contract BankAccount {
    address public owner;
    uint256 public balance;

    constructor() {
        owner = msg.sender;
        balance = 0;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this
operation");
        _;
    }

    function deposit(uint256 amount) public onlyOwner {
        require(amount > 0, "Amount must be greater than zero");
        balance += amount;
    }

    function withdraw(uint256 amount) public onlyOwner {
        require(amount > 0, "Amount must be greater than zero");
        require(amount <= balance, "Insufficient balance");
        balance -= amount;
    }

    function getBalance() public view returns (uint256) {
        return balance;
    }
}
```

Student:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8;

contract StudentData {
    struct Student {
        uint256 studentID;
        string name;
        uint256 age;
    }

    Student[] public students;

    event StudentAdded(uint256 indexed studentID, string name, uint256 age);

    function addStudent(uint256 studentID, string memory name, uint256 age)
public {
        Student memory newStudent = Student(studentID, name, age);
        students.push(newStudent);
        emit StudentAdded(studentID, name, age);
    }

    function getStudentCount() public view returns (uint256) {
        return students.length;
    }

    function getStudent(uint256 index) public view returns (uint256, string
memory, uint256) {
        require(index < students.length, "Index out of bounds");
        Student memory student = students[index];
        return (student.studentID, student.name, student.age);
    }

    fallback() external {
        revert("Fallback function called. This contract doesn't accept
Ether.");
    }
}
```