School:.....Campus:.....

AcademicYear:.....SubjectName:.....SubjectCode:.....

Semester:.....Program:.....Branch:.....Specialization:.....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Solidity Patterns – Advanced Inheritance

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Inheritance is one of the most powerful **object-oriented programming features** in Solidity. It allows one smart contract to **reuse, extend, and modify** the behavior of another, promoting modularity and reducing code duplication.
In Solidity, inheritance works similarly to other programming languages like Java or C++, but with blockchain-specific rules for execution and storage.

1. Purpose of Inheritance

The main goal of inheritance in Solidity is to:

- Enable **code reuse** by allowing derived contracts to use base contract logic.
- Facilitate **modular design**, where functionalities are divided into multiple reusable contracts.
- Support **hierarchical relationships** (Base → Derived → Final).
- Simplify maintenance and upgrades of smart contracts.

* Softwares used

- Remix IDE
- MetaMask Wallet
- OpenZeppelin Contracts (optional)
- Etherscan Testnet
- Brave / Chrome Browser

*Asapplicableaccordingtotheexperiment.
Twosheetsperexperiment(10-20)tobeused.

* Implementation Phase: Final Output (no error

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

/* 1□ Single Inheritance */
contract A {
    function showA() public pure returns (string memory) {
        return "Contract A";
    }
}

contract B is A {
    function showB() public pure returns (string memory) {
        return "Contract B inherits A";
    }
}

/* 2□ Multilevel Inheritance */
contract X {
    function showX() public pure returns (string memory) {
        return "Contract X";
    }
}

contract Y is X {
    function showY() public pure returns (string memory) {
        return "Contract Y inherits X";
    }
}

contract Z is Y {
    function showZ() public pure returns (string memory) {
        return "Contract Z inherits Y and X";
    }
}

/* 3□ Multiple Inheritance */
contract Parent1 {
    function greet() public pure virtual returns (string memory) {
        return "Hello from Parent1";
    }
}

contract Parent2 {
    function greet() public pure virtual returns (string memory) {
        return "Hello from Parent2";
    }
}

contract Child is Parent1, Parent2 {
    function greet() public pure override(Parent1, Parent2) returns (string memory) {
        return string(abi.encodePacked(super.greet(), " & Child"));
    }
```

* Observations

1. Multi-level inheritance allows a contract to reuse logic from parent contracts.
2. The virtual and override keywords enable function customization across levels.
3. The super keyword can be used to call the parent contract's implementation.
4. Remix IDE efficiently handles inheritance hierarchies with no deployment issues.
5. Gas usage remains low since only final contracts are deployed on-chain.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**Asapplicableaccordingtotheexperiment. Twosheetsperexperiment(10-20)tobeused.*