Centurion UNIVERSITY
Shaping Lives... Empowering Communities...

School:	Campus:		
Academic Year:	Subject Name:	Subject Code:	
Semester:	Program:	Branch:	Specialization: ....
Date:			

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Solidity Patterns – Advanced Inheritance

Objective/Aim:

To understand and implement advanced inheritance patterns in Solidity, including multiple inheritance, method overriding, and the use of super and virtual functions.

Apparatus/Software Used:

- **Ethereum Test Network (Sepolia)**
- **Remix IDE** (for writing and deploying smart contracts)
- **Metamask** (for wallet and transaction management)
- **Etherscan / Block Explorer** (to view contract deployment and execution details)

Theory:

Inheritance allows one contract to **reuse code** from another contract.
It enables modularity, reusability, and logical structure in smart contract systems.

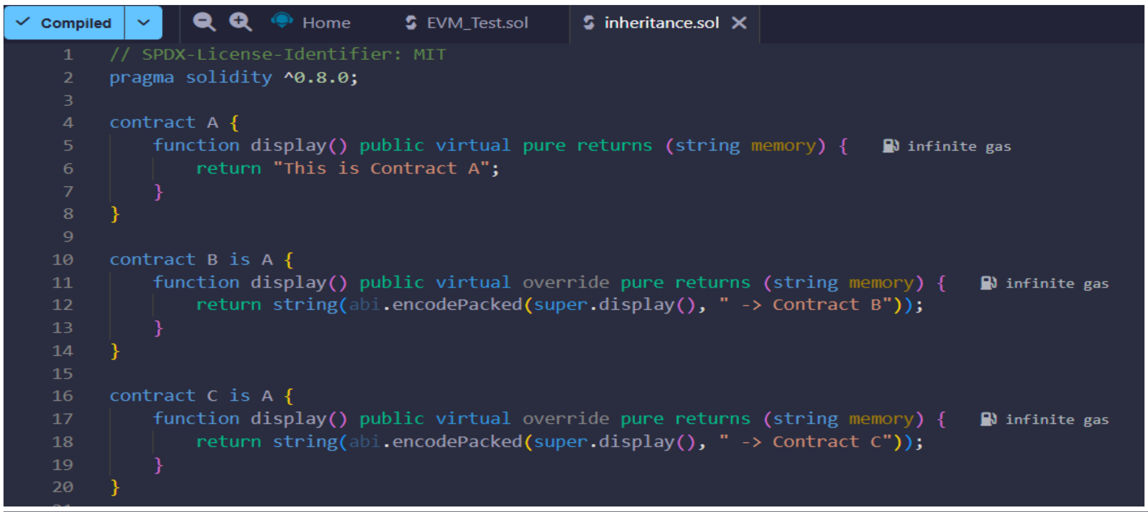
Types of Inheritance Patterns

1. **Single Inheritance:** One contract inherits from another.
2. **Multilevel Inheritance:** A contract inherits from a derived contract.
3. **Multiple Inheritance:** A contract inherits from multiple base contracts.
4. **Hierarchical Inheritance:** Multiple contracts inherit from a single base contract.

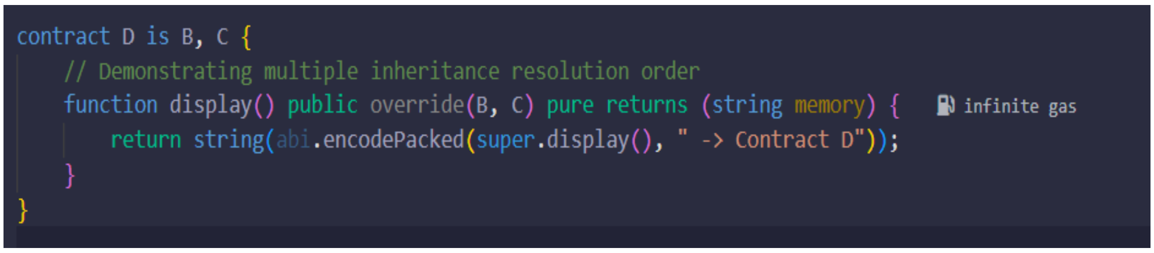
Procedure:

1.**Open Remix IDE** and create a new Solidity file named AdvancedInheritance.sol.

2. **Write the following Solidity code:**



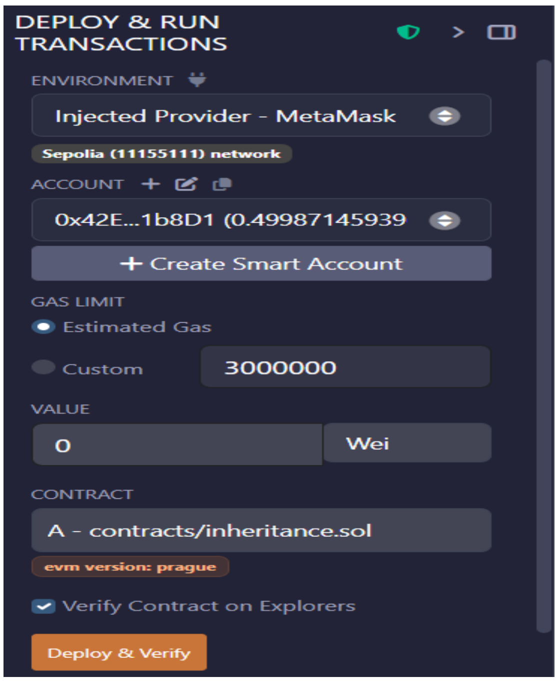
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract A {
    function display() public virtual pure returns (string memory) {    infinite gas
        return "This is Contract A";
    }
}

contract B is A {
    function display() public virtual override pure returns (string memory) {    infinite gas
        return string(abi.encodePacked(super.display(), " -> Contract B"));
    }
}

contract C is A {
    function display() public virtual override pure returns (string memory) {    infinite gas
        return string(abi.encodePacked(super.display(), " -> Contract C"));
    }
}
```



```
contract D is B, C {
    // Demonstrating multiple inheritance resolution order
    function display() public override(B, C) pure returns (string memory) {    infinite gas
        return string(abi.encodePacked(super.display(), " -> Contract D"));
    }
}
```

3.**Compile** the code using the Solidity compiler (version 0.8.x).

4.**Deploy contract D** using Remix's "Deploy & Run" tab.



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
Injected Provider - MetaMask
Sepolia (11155111) network

ACCOUNT
0x42E...1b8D1 (0.49987145939
+ Create Smart Account

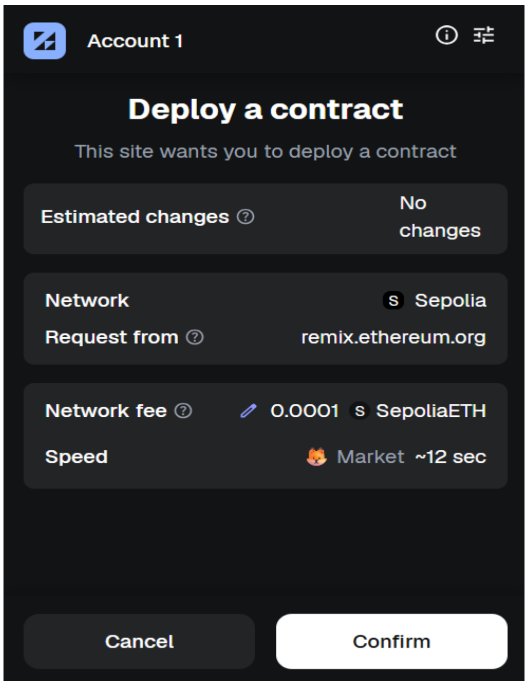
GAS LIMIT
Estimated Gas
Custom 3000000

VALUE
0 Wei

CONTRACT
A - contracts/inheritance.sol
evm version: prague

Verify Contract on Explorers

Deploy & Verify



Account 1

Deploy a contract

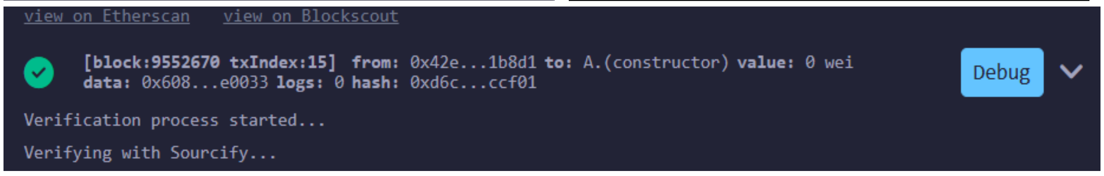
This site wants you to deploy a contract

Estimated changes: No changes

Network: Sepolia
Request from: remix.ethereum.org

Network fee: 0.0001 SepoliaETH
Speed: Market ~12 sec

Cancel   Confirm

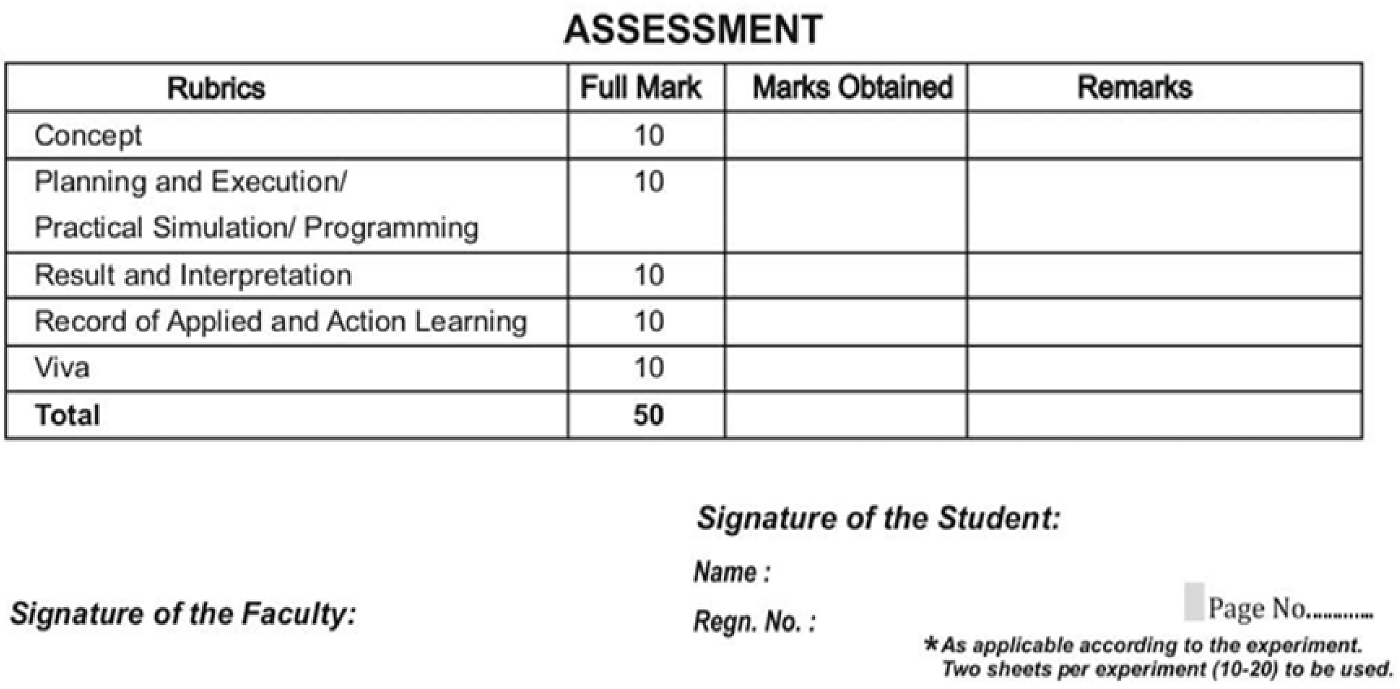


```
view on Etherscan    view on Blockscout

[block:9552670 txIndex:15] from: 0x42e...1b8d1 to: A.(constructor) value: 0 wei
data: 0x608...e0033 logs: 0 hash: 0xd6c...ccf01

Verification process started...
Verifying with Sourcify...
```

Observation:

This experiment demonstrates that:

- Solidity supports **multiple and multilevel inheritance**, enabling code reuse and organization.
- The virtual, override, and super keywords help manage **function overriding** safely.
- Solidity’s **C3 linearization algorithm** resolves the **diamond problem** by defining a clear and deterministic order of execution.
- Proper inheritance structure is essential to prevent ambiguity and unintended behavior in complex contracts.



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 :

Signature of the Faculty:

Regn. No. :

****As applicable according to the experiment. Two sheets per experiment (10-20) to be used.***