



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 Experiment : **Smart Libraries – Libraries and Proxy Contracts**

Objective/Aim:

To study and implement **Smart Contract Libraries** and **Proxy Contracts** in Solidity, understand how shared logic is reused through libraries, and observe how proxy contracts enable upgradeability without changing contract addresses.

Apparatus/Software Used:

1. MetaMask Wallet
2. Brave / Chrome Web Browser
3. Remix IDE – <https://remix.ethereum.org>
4. Ethereum Sepolia Testnet
5. Solidity Compiler (0.8.x)

Theory/Concept:

Smart Contract Libraries

Libraries in Solidity are reusable pieces of logic that can be shared across multiple contracts.

- They help reduce code duplication
- Improve gas efficiency
- Allow cleaner, modular design

Libraries can be:

- Internal libraries – embedded at compile-time
- Deployed libraries – deployed once and linked to multiple contracts

Proxy Contracts

A proxy contract is a special contract used for upgradeability.

- The proxy contract holds the state
- The implementation contract holds the logic

The proxy uses `delegatecall` to execute logic while preserving its own storage. This allows developers to upgrade contract logic without changing the address.

Common patterns:

- **Transparent Proxy Pattern**

Procedure:

1. Open MetaMask and switch to Sepolia Testnet

2. Open Remix IDE

- Visit: <https://remix.ethereum.org>
- Create a new Solidity file: MathLibrary.sol

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.19;
3
4  library MathLibrary {
5      function add(uint256 a, uint256 b) external pure returns (uint256) { infinite gas
6          return a + b;
7      }
8  }
9

```

3. Create a Contract that Uses the Library

Create another file: UseLibrary.sol

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.19;
3
4  import "./MathLibrary.sol";
5
6  contract UseLibrary {
7      function calculate(uint256 x, uint256 y) public pure returns (uint256) {
8          return MathLibrary.add(x, y);
9      }
10 }
11

```

4. Deploy Library and Link It

1. Compile MathLibrary.sol
2. Deploy the library
3. Copy the deployed library address
4. Remix may ask for linking when deploying UseLibrary.sol
5. Deploy UseLibrary

5. Implement a Simple Proxy Contract

Create a file Proxy.sol:

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.19;
3
4  contract Proxy {
5      address public implementation;
6
7      constructor(address _impl) { infinite gas 65400 gas
8          implementation = _impl;
9      }
10
11     // ✅ Handles direct ETH transfers
12     receive() external payable {} undefined gas
13
14     // ✅ Handles calls + forwards to implementation
15     fallback() external payable {} undefined gas

```

6. Deploy Implementation Contract

Create ImplV1.sol:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.19;
3
4 contract ImplV1 {
5     uint256 public value;
6
7     function setValue(uint256 x) public { 22514 gas
8         value = x;
9     }
10 }
```

creation of ImplV1 errored: [TIMEOUT] Timeout for call deployMetadataOf from udapp

✓ [block:9552983 txIndex:12] from: 0x234...e3f45 to: ImplV1.(constructor) value: 0 wei
data: 0x608...e0033 logs: 0 hash: 0x94d...77831 Debug ▼

[view on Etherscan](#) [view on Blockscout](#)

Verification process started...
Verifying with Sourcify...
Verifying with Routerscan...
Etherscan verification skipped: API key not found in global Settings.
Sourcify verification successful.
<https://repo.sourcify.dev/11155111/0xaDbAa0ecE56bc593B9018c1BE8967c747442F6FB/>
Routerscan verification successful.
<https://testnet.routerscan.io/address/0xaDbAa0ecE56bc593B9018c1BE8967c747442F6FB/contract/11155111/code>

7. Deploy Proxy Contract

Deploy Proxy.sol using the ImplV1 contract address as the constructor parameter.

[view on Etherscan](#) [view on Blockscout](#)

✓ [block:9552998 txIndex:15] from: 0x234...e3f45 to: Proxy.(constructor) value: 0 wei
data: 0x608...2f6fb logs: 0 hash: 0x6c2...31c03 Debug ▼

Verification process started...
Verifying with Sourcify...
Verifying with Routerscan...
Etherscan verification skipped: API key not found in global Settings.
Sourcify verification successful.
<https://repo.sourcify.dev/11155111/0xe09c2D8b273d48a145787057E9bC73aF415C4031/>
Routerscan verification successful.
<https://testnet.routerscan.io/address/0xe09c2D8b273d48a145787057E9bC73aF415C4031/contract/11155111/code>

8. Interact With Implementation Through Proxy

In Remix:

- Select "Proxy"
- Use "At Address" to load the proxy with ABI of ImplV1
- Call setValue(100)
- Check value — stored in proxy storage

9. Upgrade Implementation

Deploy ImplV2.sol:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.19;
3
4 contract ImplV2 {
5     uint256 public value;
6
7     function setValue(uint256 x) public { 22514 gas
8         value = x * 2; // Upgraded logic
9     }
10 }
11
```

Observation

Step	Observation
Library deployed	Functions reused without re-writing logic
UseLibrary deployed	Correct linking with library
Proxy deployed	Proxy address remains constant
ImplV1 interaction	Value stored correctly
ImplV2 upgrade	Logic changed without changing proxy address
Final behavior	Upgradeable smart contract functioning successfully

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 Faculty:

Signature of the Student:

Name :

Regn. No.