NAMA : SATYA YUDA PURNAMA

NIM : 1103184137

KELAS : TK-42-PIL

LAB 2 : Shared Wallet

-Smart contract sederhana : sharedwallet.sol

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
import "./Allowance.sol";
contract SharedWallet is Allowance {
    event MoneySent(address indexed _beneficiary, uint _amount);
    event MoneyReceived(address indexed _from, uint _amount);
    function withdrawMoney(address payable _to, uint _amount) public
ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough
money");
        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }
        emit MoneySent(_to, _amount);
        _to.transfer(_amount);
    }
    function renounceOwnership() public override onlyOwner {
        revert("can't renounceOwnership here"); //not possible with this smart
contract
    }
    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

-Mengizinkan

Pada langkah ini kita dapat membatasi pengeluaran saldo ke pemilik wallet

Pada code diatas kita juga dapat menambahkan fungsi “onlyOwner” untuk merubah ke fungsi “withdrawMoney”

Home | SendMoneyExample.sol | README.txt | Sharedwallet.sol | Allowance.sol

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
import "./Allowance.sol";
contract SharedWallet is Allowance {
    event MoneySent(address indexed _beneficiary, uint _amount);
    event MoneyReceived(address indexed _from, uint _amount);
    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");
        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }
        emit MoneySent(_to, _amount);
        _to.transfer(_amount);
    }
    function renounceOwnership() public override onlyOwner {
        revert("can't renounceOwnership here"); //not possible with this smart contract
    }
    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

-Menggunakan kontrak kembali dari OpenZeppelin

Mempunyai logika “owner-logic” langsung didalam smart contract bukan lah hal yang mudah untuk di audit. Maka dari itu cobalah untuk memecah menjadi bagian-bagian kecil dan menggunakan smart contract yang telah di audit dari OpenZeppelin. Pada build OpenZepplin yang terbaru sudah tidak memiliki fungsi “isOwner” maka dari itu kita menambahkannya sendiri.

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
import "./Allowance.sol";
contract SharedWallet is Allowance {
    event MoneySent(address indexed _beneficiary, uint _amount);
    event MoneyReceived(address indexed _from, uint _amount);
    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");
        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }
        emit MoneySent(_to, _amount);
        _to.transfer(_amount);
    }
    function renounceOwnership() public override onlyOwner {
        revert("can't renounceOwnership here"); //not possible with this smart contract
    }
    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

-Menambahkan pengeluaran untuk roles luar

Pada Langkah ini kita menambahkan mapping, jadi kita dapat menyimpan address => uint amounts. Ini akan seperti array Ketika disimpan.

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
contract Allowance is Ownable {
    event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmount);
    mapping(address => uint) public allowance;
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }
    function setAllowance(address _who, uint _amount) public onlyOwner {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
        allowance[_who] = _amount;
    }
    modifier ownerOrAllowed(uint _amount) {
        require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
        _;
    }
    function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
        allowance[_who] -= _amount;
    }
}
```

Allowance.sol

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
```

```
import "https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/access/Ownable.sol";
contract Allowance is Ownable {
    event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint
_oldAmount, uint _newAmount);
    mapping(address => uint) public allowance;
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }
    function setAllowance(address _who, uint _amount) public onlyOwner {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
        allowance[_who] = _amount;
    }
    modifier ownerOrAllowed(uint _amount) {
        require(isOwner() || allowance[msg.sender] >= _amount, "You are not
allowed!");
        _;
    }
    function reduceAllowance(address _who, uint _amount) internal
ownerOrAllowed(_amount) {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who]
- _amount);
        allowance[_who] -= _amount;
    }
}
```