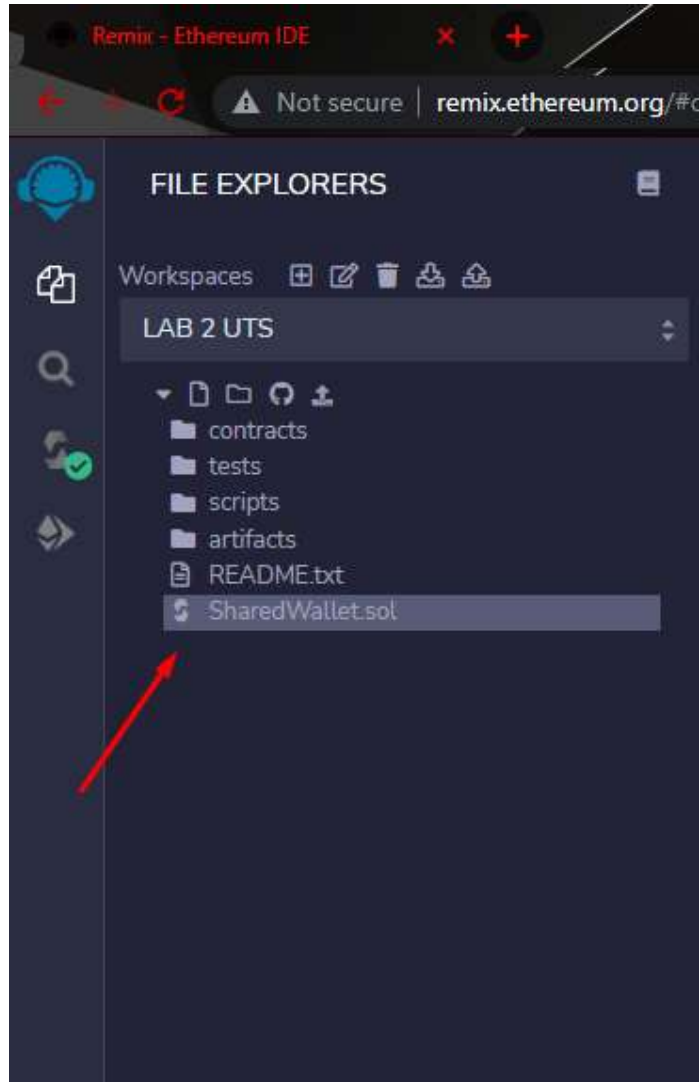


Nama	Abhista Rizky Pratama Wibowo
NIM	1103184108

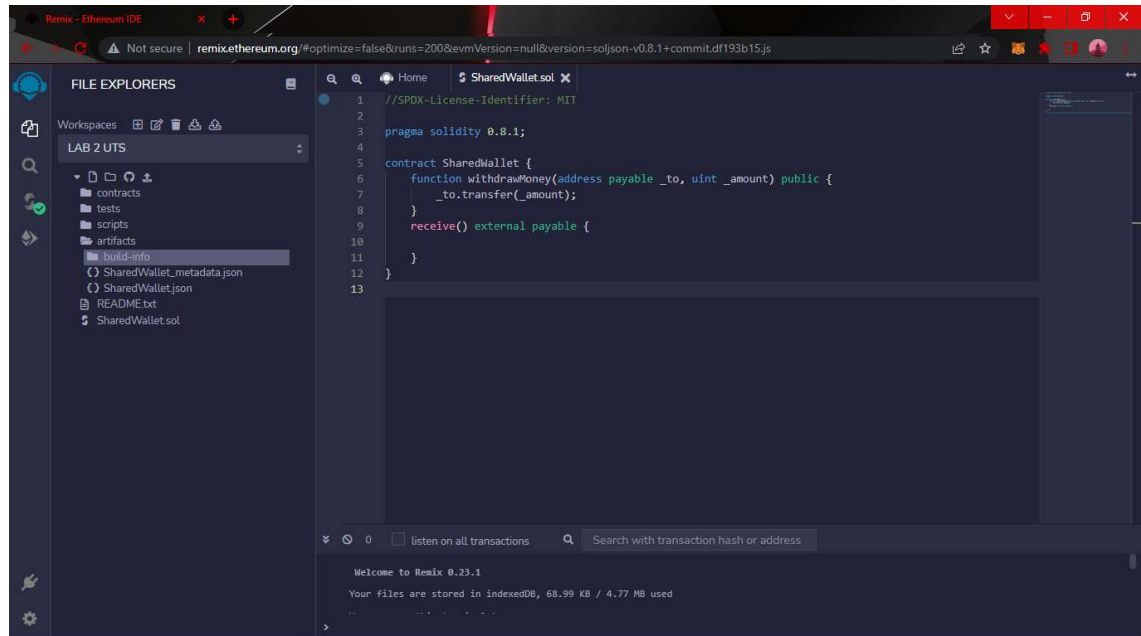
UTS Blockchain

LAB 2: Shared Wallet

1. Buka Remix pada browser dan buat file baru pada Remix dengan nama SharedWallet.sol



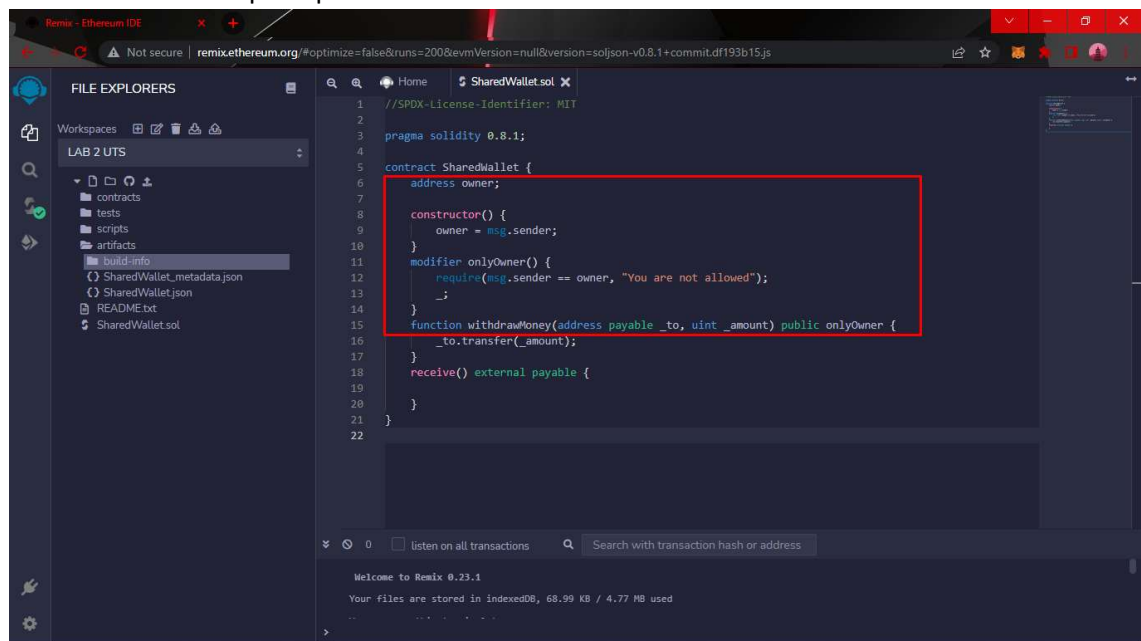
2. Setelah file berhasil dibuat, ikuti code sebagai berikut:



The screenshot shows the Remix IDE interface. On the left, the 'FILE EXPLORERS' panel displays the project structure for 'LAB 2 UTS', including folders for 'contracts', 'tests', 'scripts', and 'artifacts', and files for 'build-info', 'SharedWallet_metadata.json', 'SharedWallet.json', 'README.txt', and 'SharedWallet.sol'. The main editor window shows the 'SharedWallet.sol' file with the following Solidity code:

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 contract SharedWallet {
6     function withdrawMoney(address payable _to, uint _amount) public {
7         _to.transfer(_amount);
8     }
9     receive() external payable {
10    }
11 }
12
13
```

3. Lalu tambahkan code berikut yang bertujuan untuk menambahkan perintah dalam Smart Contract yang sudah dibuat. Perintah yang dibuat yaitu untuk HANYA memberikan izin kepada pemilik Wallet untuk melakukan Withdraw



The screenshot shows the same Remix IDE interface, but the 'SharedWallet.sol' file has been updated. The code now includes an owner address, a constructor, a modifier for owner-only access, and an updated withdraw function. A red box highlights the new code additions:

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 contract SharedWallet {
6     address owner;
7
8     constructor() {
9         owner = msg.sender;
10    }
11     modifier onlyOwner() {
12         require(msg.sender == owner, "You are not allowed");
13         _;
14     }
15     function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
16         _to.transfer(_amount);
17     }
18     receive() external payable {
19    }
20 }
21
22
```

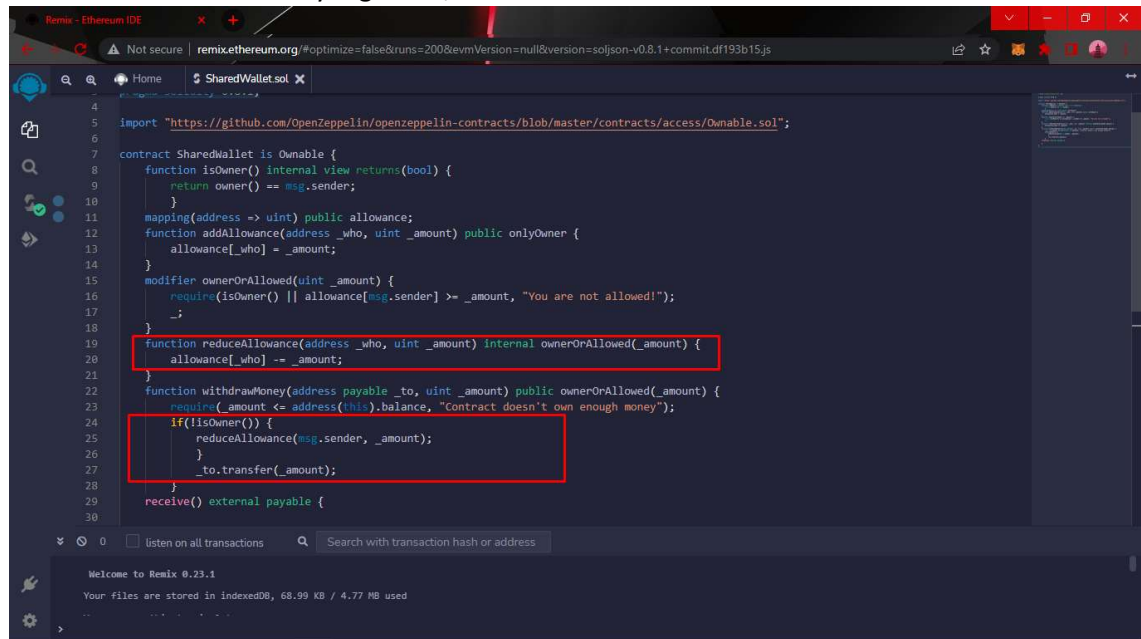
4. Selanjutnya bagaimana jika kita ingin menggunakan Smart Contract yang dapat digunakan Kembali atau *Re-useable Smart Contract*. Maka kita tambahkan code berikut yang dapat menambahkan code yang diperlukan dari Github OpenZeppelin

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import 'https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol';
6
7 contract SharedWallet is Ownable {
8     function isOwner() internal view returns(bool) {
9         return owner() == msg.sender;
10    }
11
12    function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
13        _to.transfer(_amount);
14    }
15    receive() external payable {
16
17    }
18 }
```

5. Selanjutnya kita akan menambahkan perintah untuk memberikan izin untuk menambahkan dana dari luar. Maka kita masukkan code seperti berikut

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import 'https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol';
6
7 contract SharedWallet is Ownable {
8     function isOwner() internal view returns(bool) {
9         return owner() == msg.sender;
10    }
11    mapping(address => uint) public allowance;
12    function addAllowance(address _who, uint _amount) public onlyOwner {
13        allowance[_who] = _amount;
14    }
15    modifier ownerOrAllowed(uint _amount) {
16        require(isOwner() || allowance[msg.sender] >= _amount, 'You are not allowed!');
17        _;
18    }
19    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
20        require(_amount <= address(this).balance, 'Contract doesn't own enough money');
21        _to.transfer(_amount);
22    }
23    receive() external payable {
24
25    }
26
27 }
```

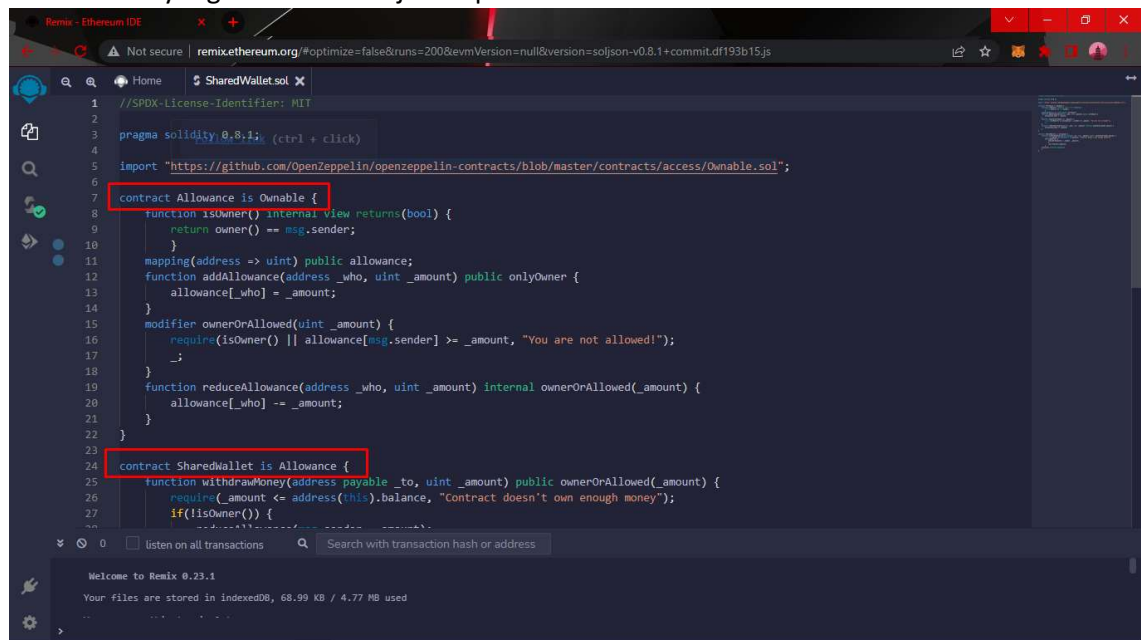
6. Selanjutnya kita akan memberikan pengembangan supaya tidak terjadi pengeluaran berkali-kali dalam waktu yang sama, maka kita akan tambahkan code berikut



```
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract SharedWallet is Ownable {
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }
    mapping(address => uint) public allowance;
    function addAllowance(address _who, uint _amount) public onlyOwner {
        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) {
        allowance[_who] -= _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);
        }
        _to.transfer(_amount);
    }
    receive() external payable {
    }
}
```

7. Selanjutnya, kita akan meningkatkan Smart Contract yang sudah kita buat supaya menjadi lebih mudah untuk dibaca dan dimengerti. Maka kita akan mengubah sedikit code yang kita buat menjadi seperti berikut



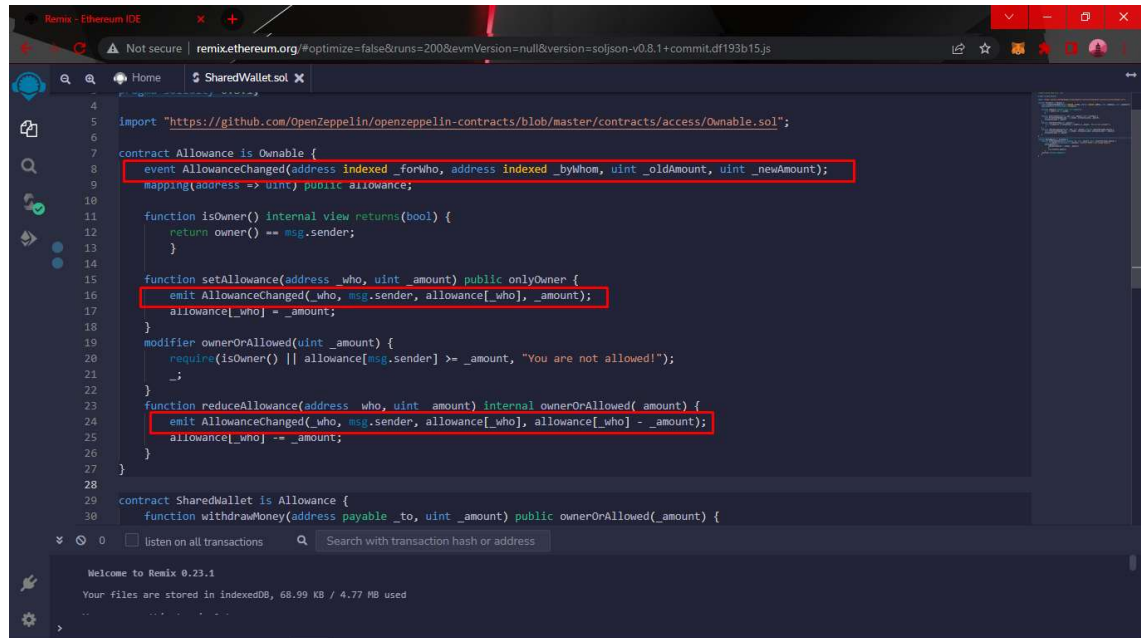
```
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1; (ctrl + click)

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract Allowance is Ownable {
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }
    mapping(address => uint) public allowance;
    function addAllowance(address _who, uint _amount) public onlyOwner {
        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) {
        allowance[_who] -= _amount;
    }
}

contract SharedWallet is Allowance {
    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);
        }
        _to.transfer(_amount);
    }
    receive() external payable {
    }
}
```

8. Selanjutnya kita akan menambahkan *Event* pada struktur code milik Allowance



```
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;
    }

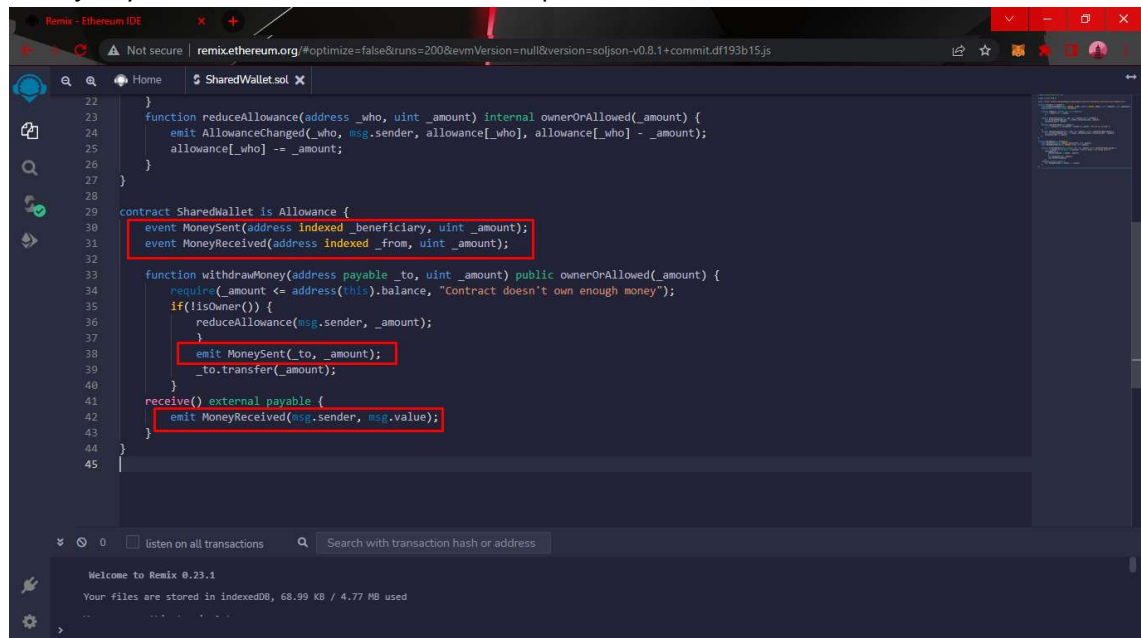
    function setAllowance(address _who, uint _amount) public onlyOwner {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
        allowance[_who] = _amount;
    }

    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;
    }
}

contract SharedWallet is Allowance {
    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
```

9. Selanjutnya kita akan menambahkan *Event* pada struktur code milik SharedWallet



```
    }

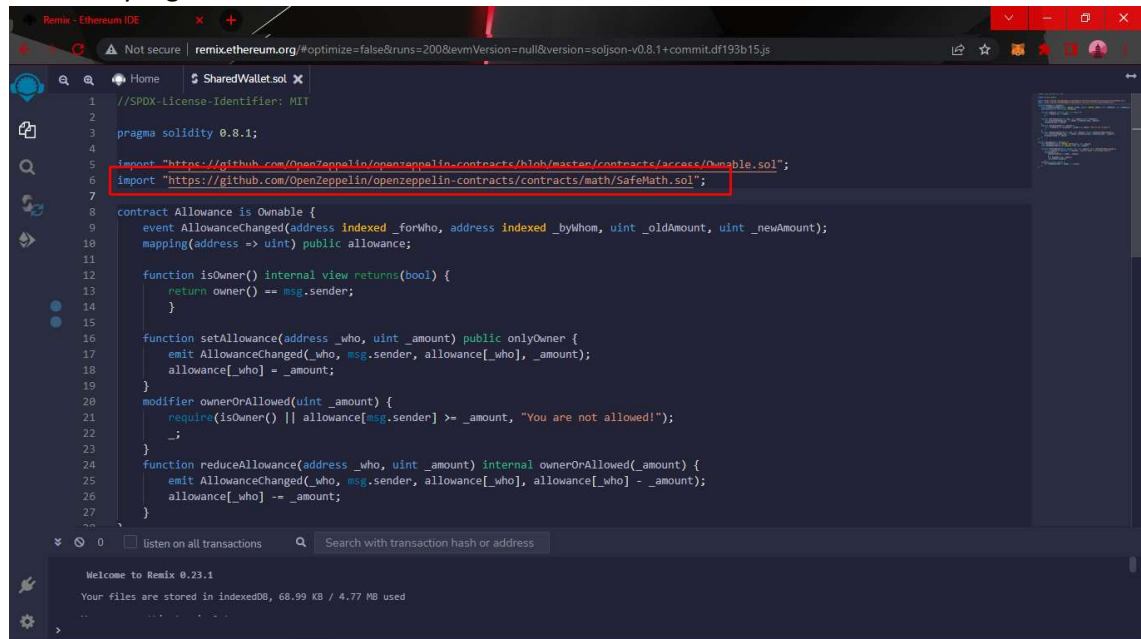
    function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
        allowance[_who] -= _amount;
    }
}

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);
    }

    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

10. Selanjutnya kita akan menambahkan *SafeMath Library Safeguard* kedalam Smart Contract yang kita buat. Maka kita akan menambahkan code berikut



The screenshot shows the Remix IDE interface with a Solidity file named 'SharedWallet.sol'. The code defines an 'Allowance' contract. A red box highlights the import statement for the SafeMath library. The code is as follows:

```
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/contracts/math/SafeMath.sol";

contract Allowance is Ownable {
    event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmount);
    mapping(address => uint) public allowance;

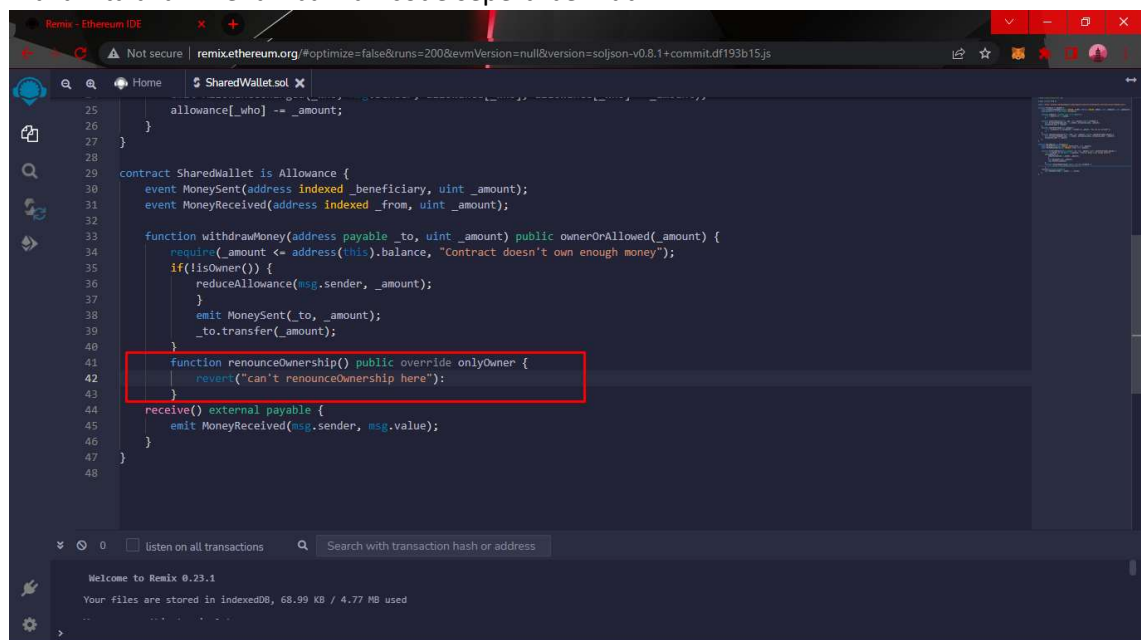
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    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;
    }
}
```

11. Selanjutnya kita akan menambahkan fungsi untuk menghapus fungsi kepemilikan, maka kita akan menambahkan code seperti berikut



The screenshot shows the Remix IDE interface with the 'SharedWallet.sol' file. The code defines a 'SharedWallet' contract that inherits from 'Allowance'. A red box highlights the 'renounceOwnership' function. The code is as follows:

```

    allowance[_who] -= _amount;
}

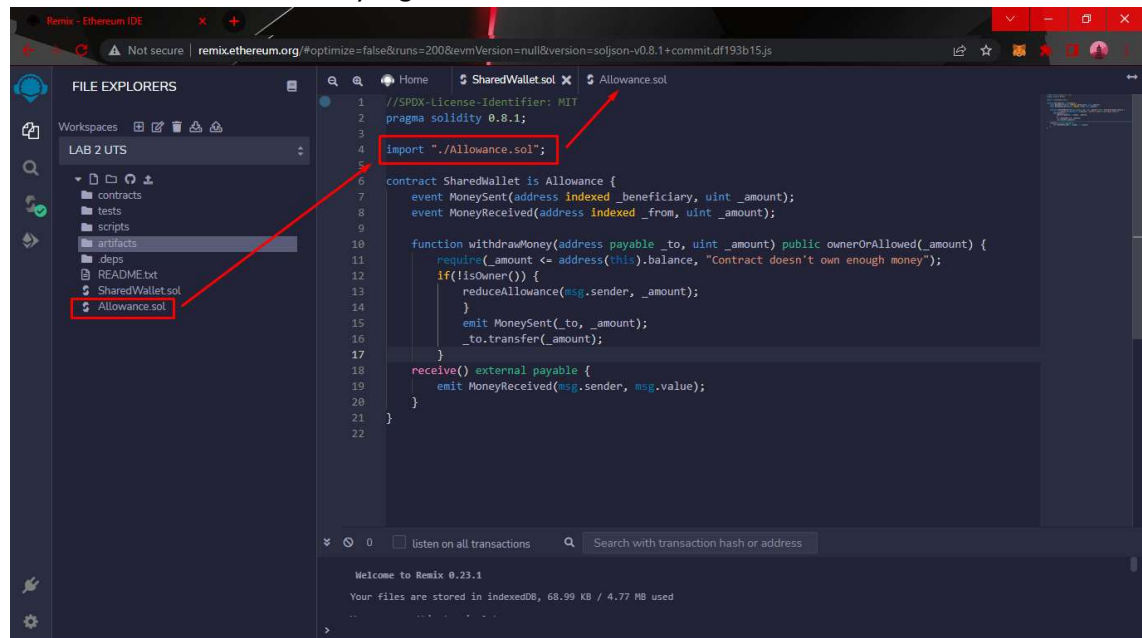
contract SharedWallet is Allowance {
    event MoneySent(address indexed _beneficiary, uint _amount);
    event MoneyReceived(address indexed _from, uint _amount);

    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");
    }

    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

12. Yang terakhir kita akan menghubungkan Smart Contract dari file yang terpisah. Dengan menambahkan code berikut maka Smart Contract akan menyambungkan Smart Contract dari dua file yang berbeda



The screenshot displays the Remix Ethereum IDE interface. On the left, the 'FILE EXPLORERS' panel shows a project structure with files 'SharedWallet.sol' and 'Allowance.sol'. The 'Allowance.sol' file is highlighted with a red box. In the main editor, the 'SharedWallet.sol' file is open, showing Solidity code. A red box highlights the line `import './Allowance.sol';` at line 4. Another red box highlights the `contract SharedWallet is Allowance {` line at line 6. The code defines a contract that inherits from 'Allowance' and includes events for 'MoneySent' and 'MoneyReceived', a 'withdrawMoney' function, and a 'receive' function. The bottom status bar indicates 'Welcome to Remix 0.23.1' and 'Your files are stored in indexedDB, 68.99 KB / 4.77 MB used'.

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity 0.8.1;
3
4 import './Allowance.sol';
5
6 contract SharedWallet is Allowance {
7     event MoneySent(address indexed _beneficiary, uint _amount);
8     event MoneyReceived(address indexed _from, uint _amount);
9
10
11     function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
12         require(_amount <= address(this).balance, "Contract doesn't own enough money");
13         if(!isOwner()) {
14             reduceAllowance(msg.sender, _amount);
15         }
16         emit MoneySent(_to, _amount);
17         _to.transfer(_amount);
18     }
19
20     receive() external payable {
21         emit MoneyReceived(msg.sender, msg.value);
22     }
23 }
```