NAMA : SATYA YUDA PURNAMA

NIM : 1103184137

KELAS : TK-42-PIL

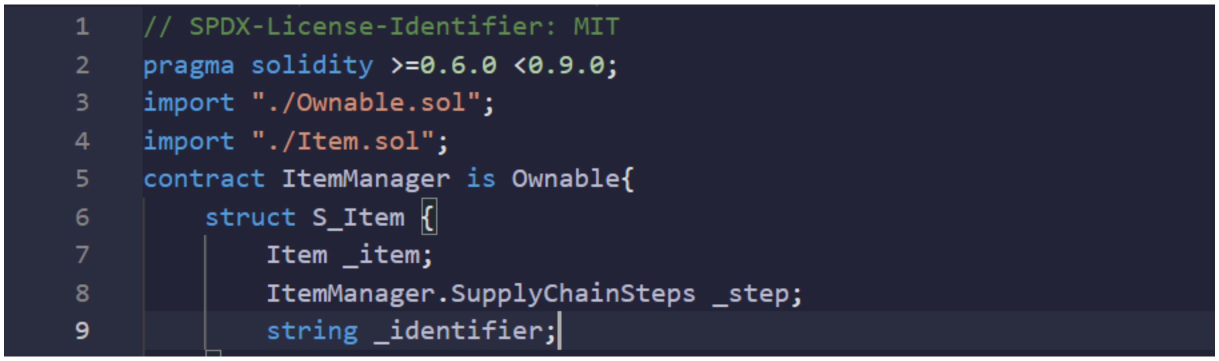
LAB 3 : Supply Chain :

ItemManager.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
import "./Ownable.sol";
import "./Item.sol";
contract ItemManager is Ownable{
    struct S_Item {
        Item _item;
        ItemManager.SupplyChainSteps _step;
        string _identifier;
    }
    mapping(uint => S_Item) public items;
        uint index;
        enum SupplyChainSteps {Created, Paid, Delivered}
        event SupplyChainStep(uint _itemIndex, uint _step, address _address);
        function createItem(string memory _identifier, uint _priceInWei) public
onlyOwner {
        Item item = new Item(this, _priceInWei, index);
        items[index]._item = item;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step), address(item));
        index++;
    }
    function triggerPayment(uint _index) public payable {
        Item item = items[_index]._item;
        require(address(item) == msg.sender, "Only items are allowed to update
themselves");
        require(item.priceInWei() == msg.value, "Not fully paid yet");
        require(items[_index]._step == SupplyChainSteps.Created, "Item is further
in the supply chain");
        items[_index]._step = SupplyChainSteps.Paid;
        emit SupplyChainStep(_index, uint(items[_index]._step), address(item));
    }
    function triggerDelivery(uint _index) public onlyOwner {
        require(items[_index]._step == SupplyChainSteps.Paid, "Item is further in
the supply chain");
        items[_index]._step = SupplyChainSteps.Delivered;
        emit SupplyChainStep(_index, uint(items[_index]._step),
address(items[_index]._item));
    }
}
```

-Smart Contract ItemManager

Pertama kita membutuhkan Smart Contract ItemManager



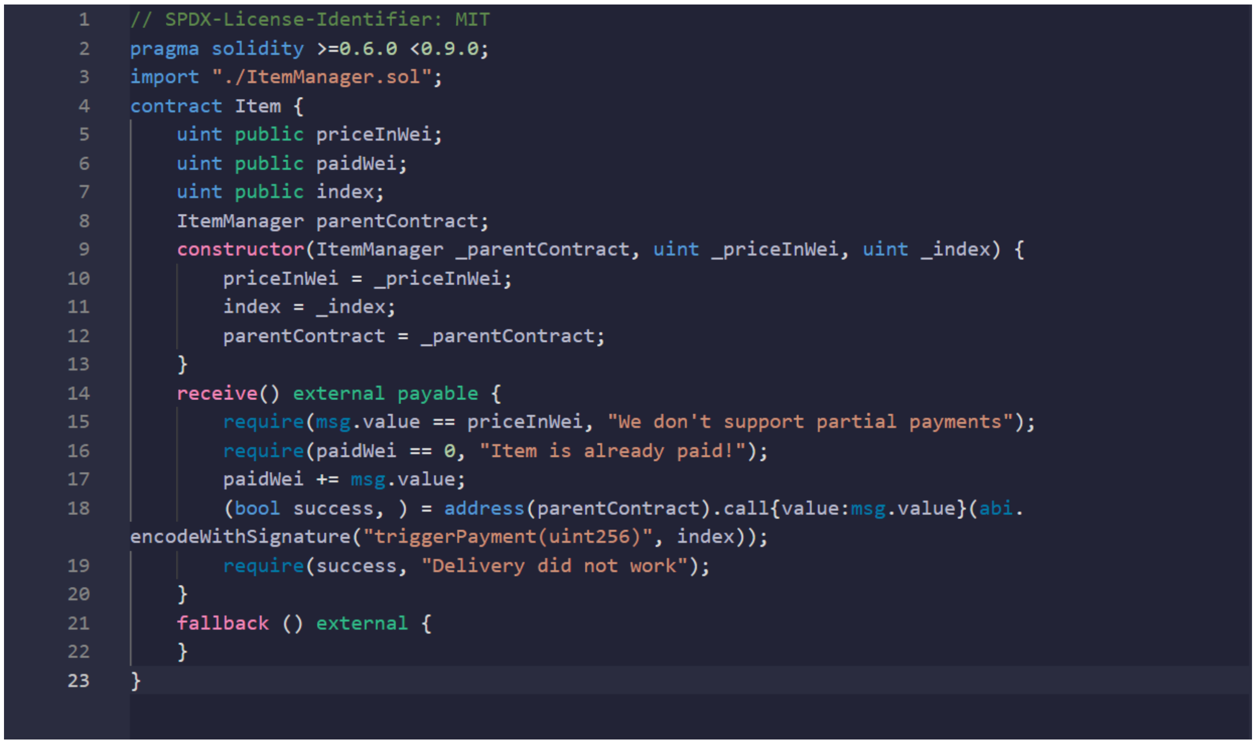
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
import "./Ownable.sol";
import "./Item.sol";
contract ItemManager is Ownable{
    struct S_Item {
        Item _item;
        ItemManager.SupplyChainSteps _step;
        string _identifier;
```

Item.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
import "./ItemManager.sol";
contract Item {
    uint public priceInWei;
    uint public paidWei;
    uint public index;
    ItemManager parentContract;
    constructor(ItemManager _parentContract, uint _priceInWei, uint _index) {
        priceInWei = _priceInWei;
        index = _index;
        parentContract = _parentContract;
    }
    receive() external payable {
        require(msg.value == priceInWei, "We don't support partial payments");
        require(paidWei == 0, "Item is already paid!");
        paidWei += msg.value;
        (bool success, ) =
address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPay
ment(uint256)", index));
        require(success, "Delivery did not work");
    }
    fallback () external {
    }
}
```

-Smart Contract Item

Kita akan membuat satu Smart Contract lagi yang bernama Item

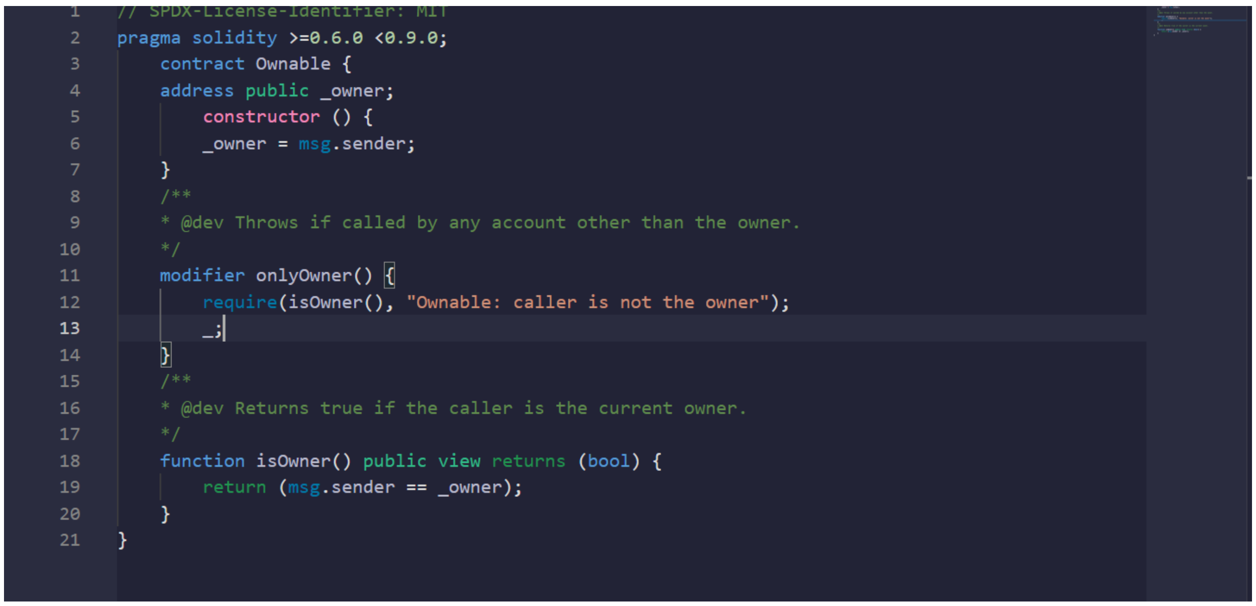


```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
import "./ItemManager.sol";
contract Item {
    uint public priceInWei;
    uint public paidWei;
    uint public index;
    ItemManager parentContract;
    constructor(ItemManager _parentContract, uint _priceInWei, uint _index) {
        priceInWei = _priceInWei;
        index = _index;
        parentContract = _parentContract;
    }
    receive() external payable {
        require(msg.value == priceInWei, "We don't support partial payments");
        require(paidWei == 0, "Item is already paid!");
        paidWei += msg.value;
        (bool success, ) = address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index));
        require(success, "Delivery did not work");
    }
    fallback () external {
    }
}
```

Owanable.sol

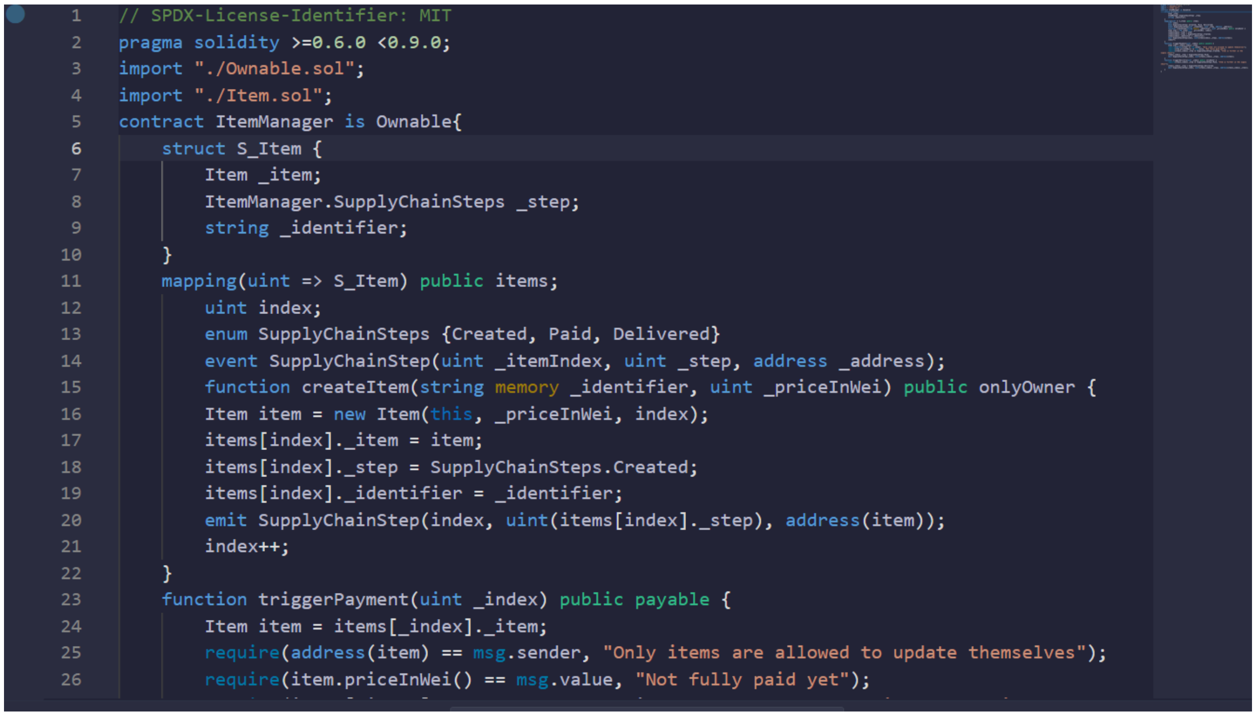
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
    contract Ownable {
    address public _owner;
        constructor () {
        _owner = msg.sender;
    }
    /**
    * @dev Throws if called by any account other than the owner.
    */
    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner");
        _;
    }
    /**
    * @dev Returns true if the caller is the current owner.
    */
    function isOwner() public view returns (bool) {
        return (msg.sender == _owner);
    }
}
```

-Fungsi kepemilikan



```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
    contract Ownable {
    address public _owner;
        constructor () {
        _owner = msg.sender;
    }
    /**
    * @dev Throws if called by any account other than the owner.
    */
    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner");
        _;
    }
    /**
    * @dev Returns true if the caller is the current owner.
    */
    function isOwner() public view returns (bool) {
        return (msg.sender == _owner);
    }
}
```

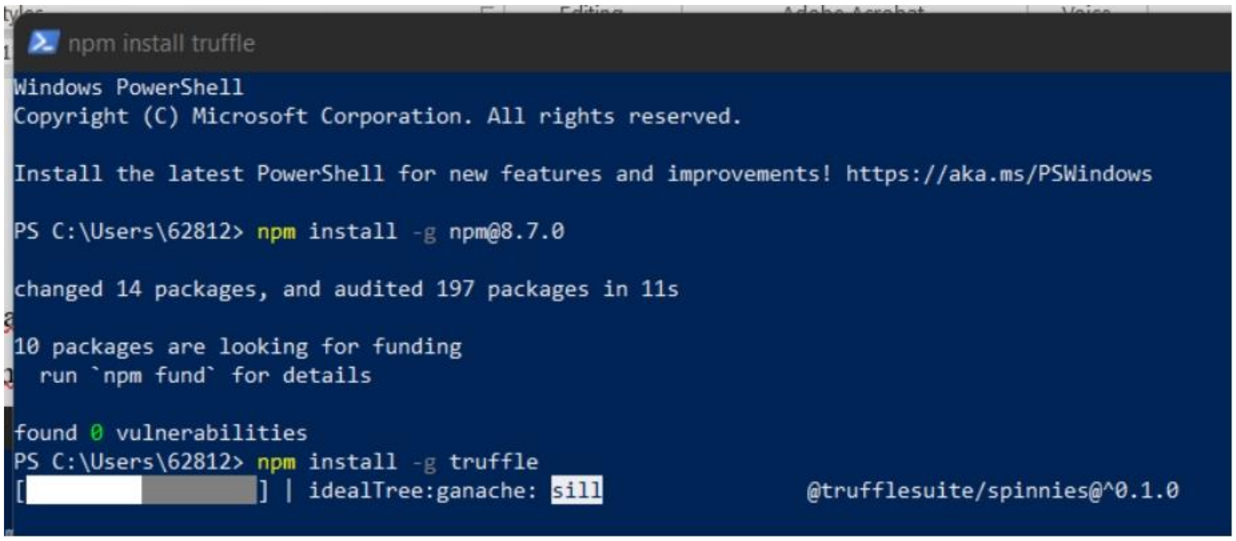
Lalu kita rubah sedikit pada smartcontract “ItemManager” kita dan kita set untuk dapat di eksekusi oleh pemilik saja



```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.9.0;
import "./Ownable.sol";
import "./Item.sol";
contract ItemManager is Ownable{
    struct S_Item {
        Item _item;
        ItemManager.SupplyChainSteps _step;
        string _identifier;
    }
    mapping(uint => S_Item) public items;
        uint index;
        enum SupplyChainSteps {Created, Paid, Delivered}
        event SupplyChainStep(uint _itemIndex, uint _step, address _address);
        function createItem(string memory _identifier, uint _priceInWei) public onlyOwner {
        Item item = new Item(this, _priceInWei, index);
        items[index]._item = item;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step), address(item));
        index++;
    }
    function triggerPayment(uint _index) public payable {
        Item item = items[_index]._item;
        require(address(item) == msg.sender, "Only items are allowed to update themselves");
        require(item.priceInWei() == msg.value, "Not fully paid yet");
```

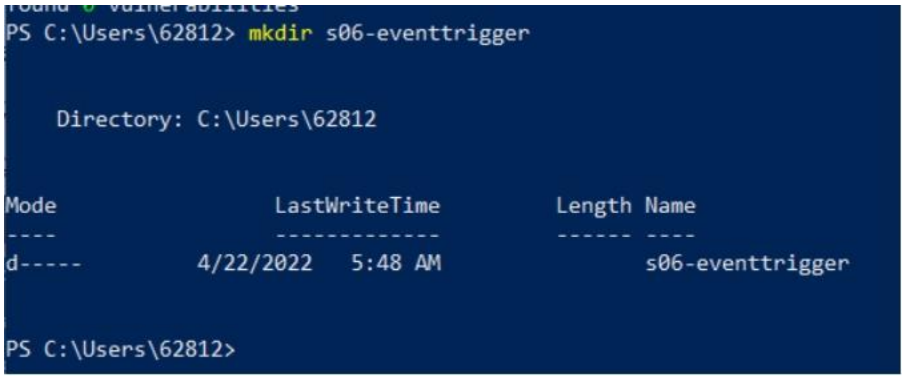
- Install Truffle

Untuk meninstall Truffle pada windows, kita menginstal berbasiskan CLI bisa menggunakan Windows Powershell dengan mengetikkan “ npm install -g npm@8.7.0 “



```
npm install truffle
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\62812> npm install -g npm@8.7.0

changed 14 packages, and audited 197 packages in 11s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\62812> npm install -g truffle
[                  ] | idealTree:ganache: sill              @trufflesuite/spinnies@^0.1.0
```

Lalu buat folder disini saya menggunakan penamaan “s06-eventtrigger”



```
found 0 vulnerabilities
PS C:\Users\62812> mkdir s06-eventtrigger


    Directory: C:\Users\62812


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/22/2022   5:48 AM                s06-eventtrigger


PS C:\Users\62812>
```

Lalu unbox react boxnya