pragma solidity 0.6.4;

contract Ownable {

address public \_owner;

constructor () public {

\_owner = msg.sender;

}

/\*\*

\* Throws if called by any account other than the owner. \*/

modifier onlyOwner() {

require(isOwner(), "Ownable: YOU ARE NOT THE OWNER!!"); \_;

}

/\*\*

\* @dev Returns true if the caller is the current owner. \*/

function isOwner() public view returns (bool) { return (msg.sender == \_owner);

} }

contract Item {

uint public priceInWei;

uint public pricePaid;

uint public index;

ItemManager parentContract;

constructor(ItemManager \_parentContract, uint \_priceInWei, uint \_index) public {

priceInWei = \_priceInWei;

index = \_index;

parentContract = \_parentContract;

}

receive() external payable {

require(priceInWei == msg.value, "We don't support partial payments");

require(pricePaid == 0, "Item is already paid!");

pricePaid += msg.value;

(bool success, ) = address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index));

require(success, "Delivery did not work");

}

fallback () external {

}

}

contract ItemManager is Ownable {

struct S\_Item {

Item \_item;

string \_identifier;

uint \_itemPrice;

ItemManager.SupplyChainSteps \_state;

}

mapping(uint => S\_Item) public items;

uint itemIndex;

enum SupplyChainSteps {Created, Paid, Delivered}

event SupplyChainStep(uint \_itemIndex, uint \_step, address \_itemAddress);

function createItem(string memory \_identifier, uint \_itemPrice) public onlyOwner {

Item item = new Item(this, \_itemPrice, itemIndex);

items[itemIndex].\_item = item;

items[itemIndex].\_identifier = \_identifier;

items[itemIndex].\_state = SupplyChainSteps.Created;

items[itemIndex].\_itemPrice = \_itemPrice;

emit SupplyChainStep(itemIndex, uint(items[itemIndex].\_state), address(item));

itemIndex++;

}

function triggerPayment(uint \_itemIndex) public payable {

require(items[\_itemIndex].\_itemPrice == msg.value, "Only full payments accepted");

require(items[\_itemIndex].\_state == SupplyChainSteps.Created, "İtem in further in the chain");

items[\_itemIndex].\_state = SupplyChainSteps.Paid;

emit SupplyChainStep(\_itemIndex, uint(items[\_itemIndex].\_state), address(items[\_itemIndex].\_item));

}

function triggerDelivery(uint \_itemIndex) public onlyOwner{

require(items[\_itemIndex].\_state == SupplyChainSteps.Paid, "Item is further in the supply chain");

items[\_itemIndex].\_state= SupplyChainSteps.Delivered;

emit SupplyChainStep(\_itemIndex, uint(items[\_itemIndex].\_state), address(items[\_itemIndex].\_item));

}

}