Practical session for blockchain

(3) Escrow

Dongkyu Kwak[1] Yeongbong Jin[1]

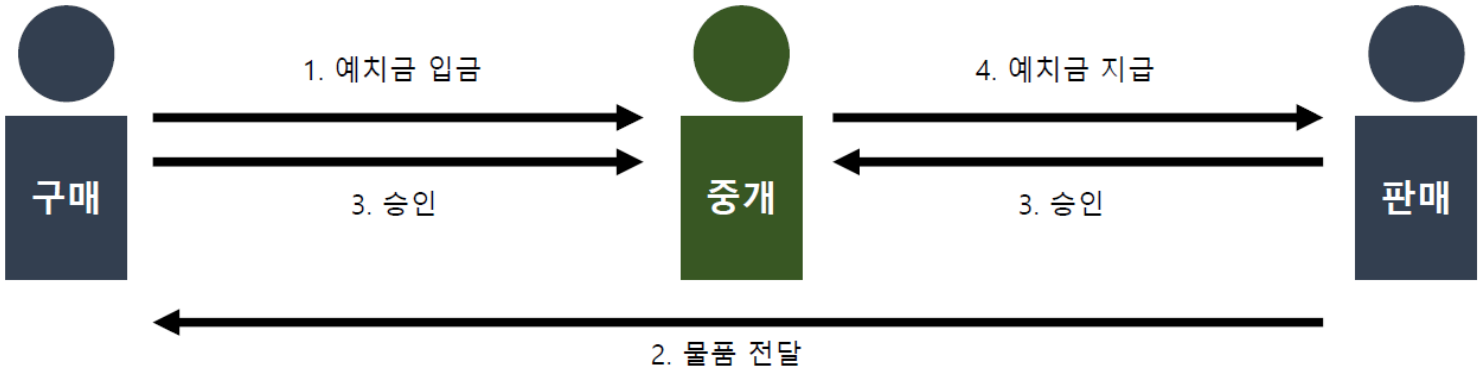
[1]Financial Risk Engineering Lab.
Seoul National University

April 20, 2022

Escrow

- **Escrow** is one of the methods for safe transaction, also called brokerage transaction.
- Both send their final determination to approve the transaction through a third-party intermediary, and then the transaction amount is paid.
- The escrow contract is only valid for a certain period from deployment; after that, the unconcluded contract would be automatically destroyed.

Escrow (cont.)



State Variables for Escrow

1. Addresses of buyer, seller, and intermediary (escrow).
2. Timestamp of contract creation.
3. Whether the transaction is approved from buyer and seller.

State Variables, Constructor

Modifier

- *modifier* statements **modifier** {_;} can be used repeatedly when a function is defined with a specific restriction.

State Variables and Constructor Function

```
pragma solidity >=0.7.0 <0.8.0;
contract Escrow {
        address public buyer; // address of buyer's wallet
        address public seller; // address of seller's wallet
        address private escrow; // address of intermediary's wallet (private)
        uint private start; // timestamp of contract creation (private)

        bool buyerOk;
        bool sellerOk;

        modifier onlyBuyer {
            require(msg.sender == buyer, "Only buyer can call this function");
        _;}

        constructor(address buyer_address, address seller_address) {
                buyer = buyer_address;
                seller = seller_address;
                escrow = msg.sender;
                start = block.timestamp;
        }
```

Payable Keyword

- *payable* keyword can be used to both **function, address** and **modifier**.
- *address payable* can be paid ETH from the contract through **transfer** or **send**.
- *payable address* (payable(address)) can be used to make a non-payable address as a payable one.
- *payable modifier* allows those who calls the function to send ETH to the contract from their wallet.

Payable Keyword (cont.)

[Ex. 1] Payable Keywords

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.8.0;
contract payableExample {

        // 1. address payable
        address payable owner;
        address subOwner;

        constructor(address _subOwner) {
                owner = msg.sender;
        subOwner = _subOwner;
        }

        // 2. payable modifier
        function deposit() public payable {

        }

        function withdrawOwner(uint _eth) public {
                require(_eth * 1 ether <= address(this).balance);
                owner.transfer(_eth * 1 ether);
                // subOwner.transfer(_eth * 1 ether);
        }

        // 3. payable(<address>)
        function withdrawSubowner(uint _eth) public {
                payable(subOwner).transfer(_eth * 1 ether);
        }

}
```

Function - payBalance, deposit

payBalance, deposit

```
function deposit() public payable onlyBuyer{

}

function payBalance() private {
        payable(escrow).transfer(address(this).balance / 100); // pay a fee for escrow
        payable(seller).transfer(address(this).balance);
}
```

Function - Accept

Selfdestruct

- *selfdestruct* function deletes the deployed bytecode from the blockchain and carries the Ether in the contract to a specific target.

accept

```
function accept() public {

        if (msg.sender == buyer){
                buyerOk = true;
        } else if (msg.sender == seller){
                sellerOk = true;
        }

        if (buyerOk && sellerOk){
                payBalance();
        } else if (buyerOk && !sellerOk && block.timestamp > start + 30 days) {
                selfdestruct(payable(buyer));
        }
}
```

Function - cancel, kill

cancel, kill

```
        function cancel() public {
                if (msg.sender == buyer){
                        buyerOk = false;
                } else if (msg.sender == seller){
                        sellerOk = false;
                }

                if (!buyerOk && !sellerOk){
                        selfdestruct(payable(buyer));
                        // destruct when the contract is not approved by calling the cancel function
                }
        }

        function kill() public  {
                if (msg.sender == escrow) {
                        selfdestruct(payable(buyer)); // destruct if the escrow kill this contract
                }
        }
        function view_balance() public view returns (uint){return address(this).balance;}


}
```

Function - vote

vote

```
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(sender.weight != 0, "Has no right to vote");
    require(!sender.voted, "Already voted.");
    sender.voted = true;
    sender.vote = proposal;

    // If 'proposal' is out of the range of the array,
    // this will throw automatically and revert all
    // changes.
    proposals[proposal].voteCount += sender.weight;
}
```

Escrow

1. Create a Escrow contract
2. Try 'deposit' function with the various addresses
3. After both buyer and seller call 'accept', check the Ether transferred to seller.
4. After buyer or seller call 'cancel', check the Ether transferred to buyer.
5. After the escrow call 'kill', check the Ether transferred to buyer.

References

- Solidity official documentation, https://solidity-kr.readthedocs.io/ko/latest/introduction-to-smart-contracts.html