

*Be as proud of Sogang as Sogang is proud of you*

# 블록체인 스마트컨트랙트



서강대학교  
SOGANG UNIVERSITY

- **Solidity**

- 라이브러리
- send/call/transfer
- receive/fallback
- delegatecall

- **스마트컨트랙트 예제**

- Counter 스마트컨트랙트
- Bank 스마트컨트랙트

- **프로젝트 설명**

### ■ 매핑(mapping)

- 매핑 데이터 조회: 값으로 키를 얻는 방법이 필요하다면?

```
contract SearchMapping {  
  
    mapping(address => uint) public balances;  
  
    function findAddressByBalance  
  
}
```


## ■ overflow와 underflow

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.7.6;

contract SimpleStorage {
    uint min = 0;
    uint max = 115792089237316195423570985008687907853269984665640564039457584007913129639935;

    function under() public view returns( uint) {
        return min - 1;
    }

    function over() public view returns( uint) {
        return max + 1;
    }
}
```

▼ SIMPLESTORAGE AT 0XF8E...9FBE8 (M) 

Balance: 0 ETH

**over**

0: uint256: 0

**under**

0: uint256: 115792089237316195423570985008687907853269984665640564039457584007913129639935

## ■ 라이브러리

- 다른 스마트 컨트랙트에서 **재사용**할 수 있는 코드 블록
  - 공통 기능을 다른 스마트컨트랙트에서 중복 구현하지 않고도 코드를 간결하게 유지

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.5.0 < 0.8.0;
```

```
library Math{
```

```
    function add(uint8 a, uint8 b) internal pure returns (uint8) {  
        require(a+b >= a , "Error: addition overflow");  
        return a+b;  
    }  
}
```

```
contract Ex6_11{  
    using Math for uint8;
```

```
    function overflow(uint8 _num1,uint8 _num2) public pure returns(uint8) {  
        return _num1+_num2;  
    }  
    function noOverflow1(uint8 _num1,uint8 _num2) public pure returns(uint8) {  
        return Math.add(_num1 ,_num2);  
    }  
    function noOverflow2(uint8 _num1,uint8 _num2) public pure returns(uint8) {  
        return _num1.add(_num2);  
    }  
}
```

## ■ 라이브러리

### ■ 결과

**overflow**

\_num1: 255

\_num2: 1

Calldata

Parameters

call

0: uint8: 0

**noOverflow1**

\_num1: 255

\_num2: 1

Calldata

Parameters

call

**noOverflow2**

\_num1: 255

\_num2: 1

Calldata

Parameters

call

call to Ex6\_11.noOverflow1

**CALL** [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4  
to: Ex6\_11.noOverflow1(uint8,uint8) data: 0xb71...00001

call to Ex6\_11.noOverflow1 errored: Error occurred: revert.

revert

The transaction has been reverted to the initial state.

Reason provided by the contract: "Error: addition overflow".

You may want to cautiously increase the gas limit if the transaction went out of gas.

call to Ex6\_11.noOverflow2

**CALL** [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4  
to: Ex6\_11.noOverflow2(uint8,uint8) data: 0x508...00001

call to Ex6\_11.noOverflow2 errored: Error occurred: revert.

revert

The transaction has been reverted to the initial state.

Reason provided by the contract: "Error: addition overflow".

You may want to cautiously increase the gas limit if the transaction went out of gas.

## ■ send/call/transfer

- address 자료형의 이더를 전송하는 내장 함수
- (payable 적용된 address).send(송금할 wei)
- (payable 적용된 address).transfer(송금할 wei)
- (address).call{value: 송금할 wei}("")

	payable	가스 소비량	실패시
send	필요	2300 gas	false 반환, require 함께 이용 권장
call		가변	false 반환, require 함께 이용 권장
transfer	필요	2300 gas	트랜잭션 실패

- **send/transfer**

```
contract Ex6_14 {

    function getBalance(address _address) public view returns(uint) {
        return _address.balance;
    }

    function etherUnits() public pure returns(uint,uint,uint) {
        return(1 ether, 1 gwei, 1 wei);
    }

    function ethDelivery1(address payable _address) public payable {
        bool result = _address.send(10 ether);
        require(result, "Failed");
    }

    function ethDelivery2(address _address) public payable {
        payable(_address).transfer(msg.value);
    }

}
```

EX6\_14 AT OXD91...39138 (MEMORY)

Balance: 0 ETH

ethDelivery1

address\_address

ethDelivery2

address\_address

etherUnits

0: uint256: 10000000000000000000

1: uint256: 10000000000

2: uint256: 1

getBalance

address\_address



## ■ send/transfer





- **receive:** 스마트컨트랙트가 이더를 전송받는 함수
  - 매개변수, 반환값 없음

```
contract Ex6_16 {

    event Obtain(address from, uint amount);

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

    function getBalance() public view returns(uint) {
        return address(this).balance;
    }

    function sendEther() public payable {
        payable(address(this)).transfer(msg.value);
    }

}
```

```
{
  "event": "Obtain",
  "args": {
    "0": "0xd9145CCE52D386f254917e481eB44e9943F39138",
    "1": "10",
    "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
    "amount": "10"
  }
}
```

▼ EX6\_16 AT 0XD91...39138 (MEMORY)

Balance: 0 ETH

sendEther

getBalance

0: uint256: 0



VALUE

10

Wei

sendEther



▼ EX6\_16 AT 0XD91...39138 (MEMORY)

Balance: 0.000000000000000001 ETH

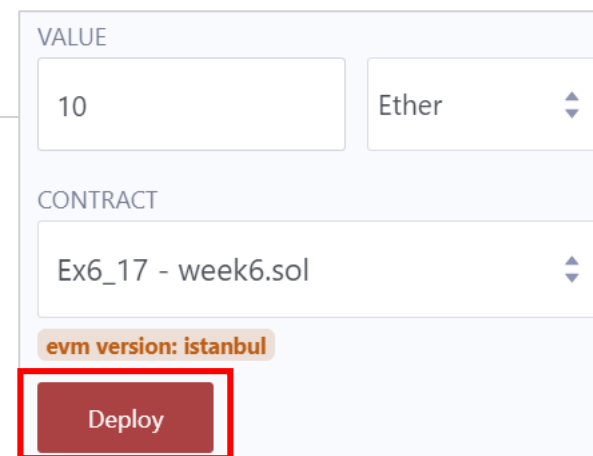
sendEther

getBalance

0: uint256: 10

- payable을 적용한 생성자
  - 배포와 동시에 스마트컨트랙트로 이더 전송 가능

```
contract Ex6_17 {  
  
    constructor() payable {  
  
    }  
  
    function getBalance() public view returns(uint) {  
        return address(this).balance;  
    }  
}
```



▼ EX6\_17 AT 0X358...D5EE3 (MEMORY)

Balance: 10 ETH

getBalance

0: uint256: 10000000000000000000

## ■ call: 외부 컨트랙트의 함수 호출

```
contract Math {
    uint result = 0;
    function add(uint256 _num1, uint256 _num2) public {
        result = _num1 + _num2;
    }
    function returnResult() public view returns(uint) {
        return result;
    }
}
```

```
contract Buttons {
    function addButtons(address _address, uint _num1, uint _num2) public {
        (bool success, ) = _address.call(
            abi.encodeWithSignature("add(uint256,uint256)",_num1 ,_num2)
        );
        require(success, "Failed");
    }
    function showResult(address _address) public returns(bytes memory) {
        (bool success, bytes memory result) = _address.call(
            abi.encodeWithSignature("returnResult()")
        );
        require(success, "Failed");
        return result;
    }
}
```

- **call: 외부 컨트랙트의 함수 호출**
  - Math 컨트랙트 주소의 함수 호출

Deployed Contracts

▶ MATH AT 0XD91...39138 (MEMORY)

▼ BUTTONS AT 0XD8B...33FA8 (MEMORY)

Balance: 0 ETH

**addButtons**

\_address:

0xd9145CCE52D386f254917e481eB4

\_num1:

3

\_num2:

2

Calldata

Parameters

transact

▼ MATH AT 0XD91...39138 (MEMORY)

Balance: 0 ETH

add

uint256 \_num1, uint256 \_num2

returnResult

0: uint256: 5

showResult

0xd9145CCE52D386f254917e

decoded output


{

"0": "bytes: 0x0000000000000000000000000000000000000000000000000000000000000000"

## ■ fallback(1/2)

- 존재하지 않는 함수 호출시 실행되는 함수

```
contract Receiver {  
  
    string lastFunctionCalled; // 마지막으로 호출된 함수명을 저장  
  
    fallback() external {  
        lastFunctionCalled = "fallback()"; // 함수명 저장  
    }  
    function outPut() public payable {  
        lastFunctionCalled = "outPut()"; // 함수명 저장  
    }  
    function getBalance() public view returns(uint) {  
        return address(this).balance;  
    }  
    function getLastFunctionCalled() public view returns (string memory) {  
        return lastFunctionCalled; // 마지막으로 호출된 함수명을 반환  
    }  
}
```

▼ RECEIVER AT 0X296...C35D3 (M) 

Balance: 0 ETH

outPut



getBalance



0: uint256: 0

getLastFunction...

0: string:

Deployed/Unpinned Contracts

> RECEIVER AT 0X296...C35D3 (MI)  

▼ CALLER AT 0X9CF...E70FA (MEM)  

Balance: 0 ETH

expectFallback	0x2961b3e121ebC1d889fFc
outPutWithEther	0x2961b3e121ebC1d889fFc

## ■ fallback (2/2)

contract Caller {

```
function expectFallback(address _address) public {
    (bool success, ) = _address.call(
        abi.encodeWithSignature("outPut2()")
    );
    require(success, "Failed");
}
```

// 존재하지 않는 함수 호출

expectFallback 0x2961b3e121ebC1d889fFc

```
function outPutWithEther(address _address) public payable {
    (bool success, ) = _address.call{value:msg.value}(
        abi.encodeWithSignature("outPut()")
    );
    require(success, "Failed");
}
```

VALUE

10

Ether

outPutWithEther

0x2961b3e121ebC1d889fFc

RECEIVER AT 0X296...

Balance: 0 ETH

outPut

getBalance

0: uint256: 0

getLastFuncio...

0: string: fallback()

RECEIVER AT 0X296...C35D3 (M)

Balance: 10 ETH

outPut

getBalance

0: uint256: 10000000000000000000

getLastFuncio...

0: string: outPut()



## ■ delegatecall (1/2)


- 주소형의 내장함수, 외부 스마트 컨트랙트의 함수 실행
- 호출된 함수의 로직만 수행
- 결과는 delegatecall 이 명시된 스마트컨트랙트에 저장
  - 로직을 변경하여 스마트컨트랙트 배포 가능


```
contract Points {  
  
    uint public total;  
    event From(address from);  
  
    function addPoints(uint _point) public {  
        total += _point;  
        emit From(msg.sender);  
    }  
}
```

## ■ delegatecall (2/2) : call 과 delegatecall 비교

```
contract UserInfo {  
  
    uint public total;  
    function pointUpWithCall(address _address, uint _point) public {  
        (bool success, ) = _address.call(  
            abi.encodeWithSignature("addPoints(uint256)",_point)  
        );  
        require(success, "Failed");  
    }  
  
    function pointUpWithDelegateCall(address _address, uint _point) public {  
        (bool success, ) = _address.delegatecall(  
            abi.encodeWithSignature("addPoints(uint256)",_point)  
        );  
        require(success, "Failed");  
    }  
}
```

## ■ delegatecall

POINTS AT 0X5FD...9D88D (MEMORY) 



USERINFO AT 0X7B9...B6ACE (MEMOF) 

Balance: 0 ETH

**pointUpWithCall**

\_address: 0x5FD6eB55D12E759a21C09eF703fe


\_point: 3

 Calldata  Parameters **transact**

**pointUpWithD...** address \_address, uint256 \_px

**total**

0: uint256: 0

POINTS AT 0X5FD...9D88D (MEMORY) 

Balance: 0 ETH


**addPoints** uint256 \_point

**total**

0: uint256: 3

```
"event": "From",
"args": {
  "0": "0x7b96aF9Bd211cBf6BA5b0dd53aa61Dc5806b6AcE",
  "from": "0x7b96aF9Bd211cBf6BA5b0dd53aa61Dc5806b6AcE"
```

From: UserInfo의 주소

USERINFO AT 0X7B9...B6ACE (MEMOF) 



Balance: 0 ETH

**pointUpWithCall** address \_address, uint256 \_px

**pointUpWithDelegateCall**

\_address: 0x5FD6eB55D12E759a21C09eF703fe

\_point: 99

 Calldata  Parameters **transact**

**total**

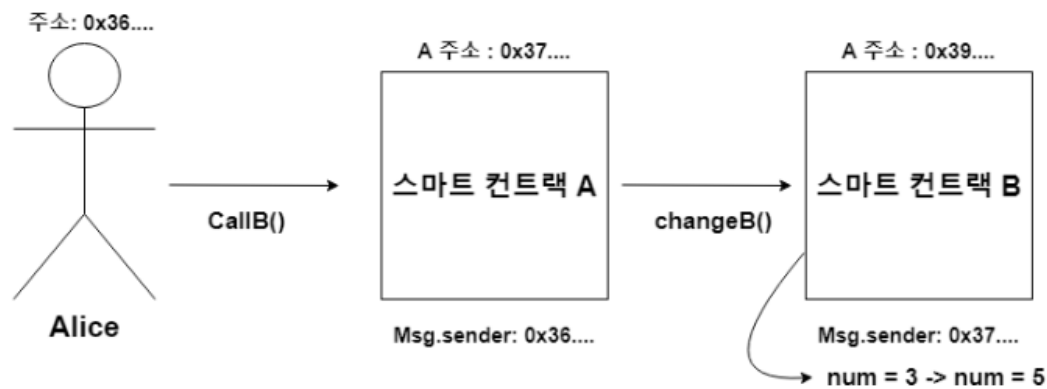
0: uint256: 99

```
"event": "From",
"args": {
  "0": "0x5B38Da6a701c568545dCf cB03FcB875f 56beddC4",
  "from": "0x5B38Da6a701c568545dCf cB03FcB875f 56beddC4"
```

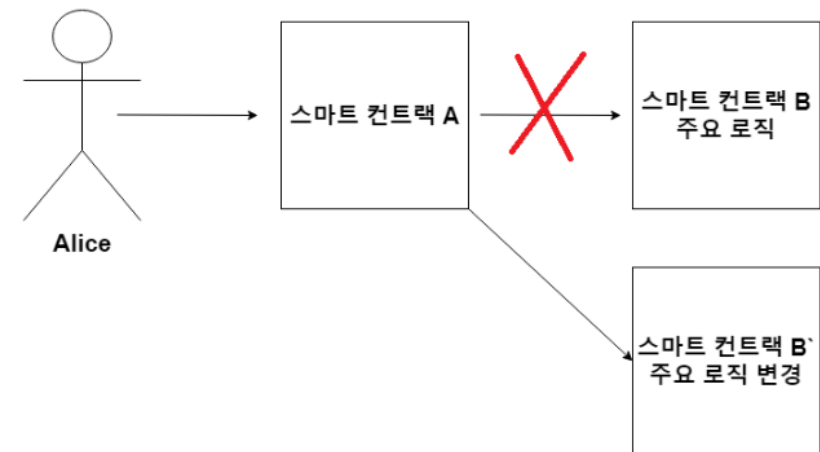
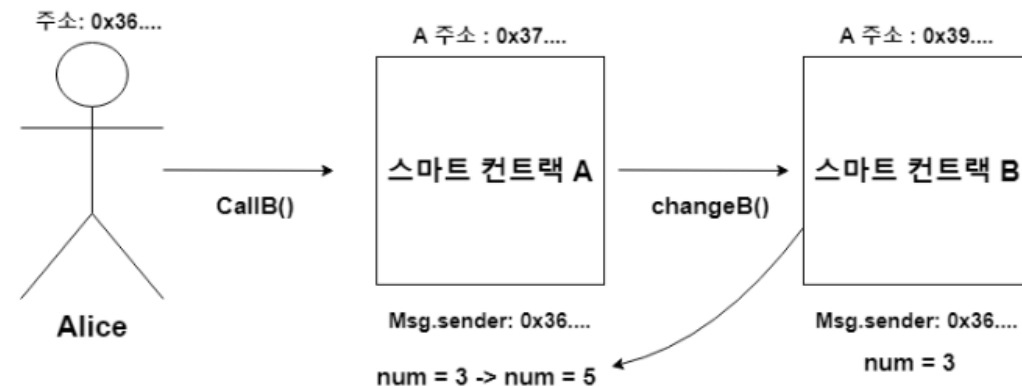
From: pointUpWithDelegateCall을 호출한 외부소유계정의 주소

## ■ call & delegatecall

### Call



### Delegate call



## ■ delegatecall 응용: 새로운 로직을 갖는 스마트컨트랙트 배포 사용 가능



```
contract Points {                                // 기존 스마트 컨트랙트
    uint public total;
    function addPoints(uint _point) public {
        total += _point;
    }
}



contract Points2 {                               // 로직 변경하여 배포
    uint public total;
    function addPoints(uint _point) public {
        total += _point*2;
    }
}



contract UserInfo {
    uint public total;

    function pointsUpWithDelegateCall(address _address, uint _point) public {
        (bool success, ) = _address.delegatecall(
            abi.encodeWithSignature("addPoints(uint256)",_point)
        );
        require(success, "Failed");
    }
}
```

- **delegatecall** 응용: 새로운 로직을 갖는 스마트컨트랙트 배포 사용 가능

> POINTS AT 0X9DE...38584 (MEM)  

> POINTS2 AT 0X77D...A7E8D (ME)  

▼ USERINFO AT 0X391...5ACCD (!)  




Balance: 0 ETH




pointsUpWithD...




address \_address, uint256 \_

total


0: uint256: 0

> POINTS AT 0X9DE...38584 (MEM)   

> POINTS2 AT 0X77D...A7E8D (ME)   



▼ USERINFO AT 0X391...5ACCD (!)   

Balance: 0 ETH

pointsUpWithDelegateCall 

\_address: 0x9De0a845D9c7558B0B8E1a68303




\_point: 3




 Calldata  Parameters 




transact

total


0: uint256: 3

> POINTS AT 0X9DE...38584 (MEM)   

> POINTS2 AT 0X77D...A7E8D (ME)   



▼ USERINFO AT 0X391...5ACCD (!)   

Balance: 0 ETH

pointsUpWithDelegateCall 

\_address: 0x77dE3535Cb32c0cBDf1ab92b373

\_point: 5

 Calldata  Parameters 

transact

total

0: uint256: 13

- 카운트를 가져오고, 증가시키고, 감소시키는 간단한 계약
- 데이터
  - count: 외부에 노출시키지 않음
- 함수
  - get(): 현재 카운트 값을 반환하는 함수
  - inc(): 카운트를 1 증가시키는 함수
  - dec(): 카운트를 1 감소시키는 함수

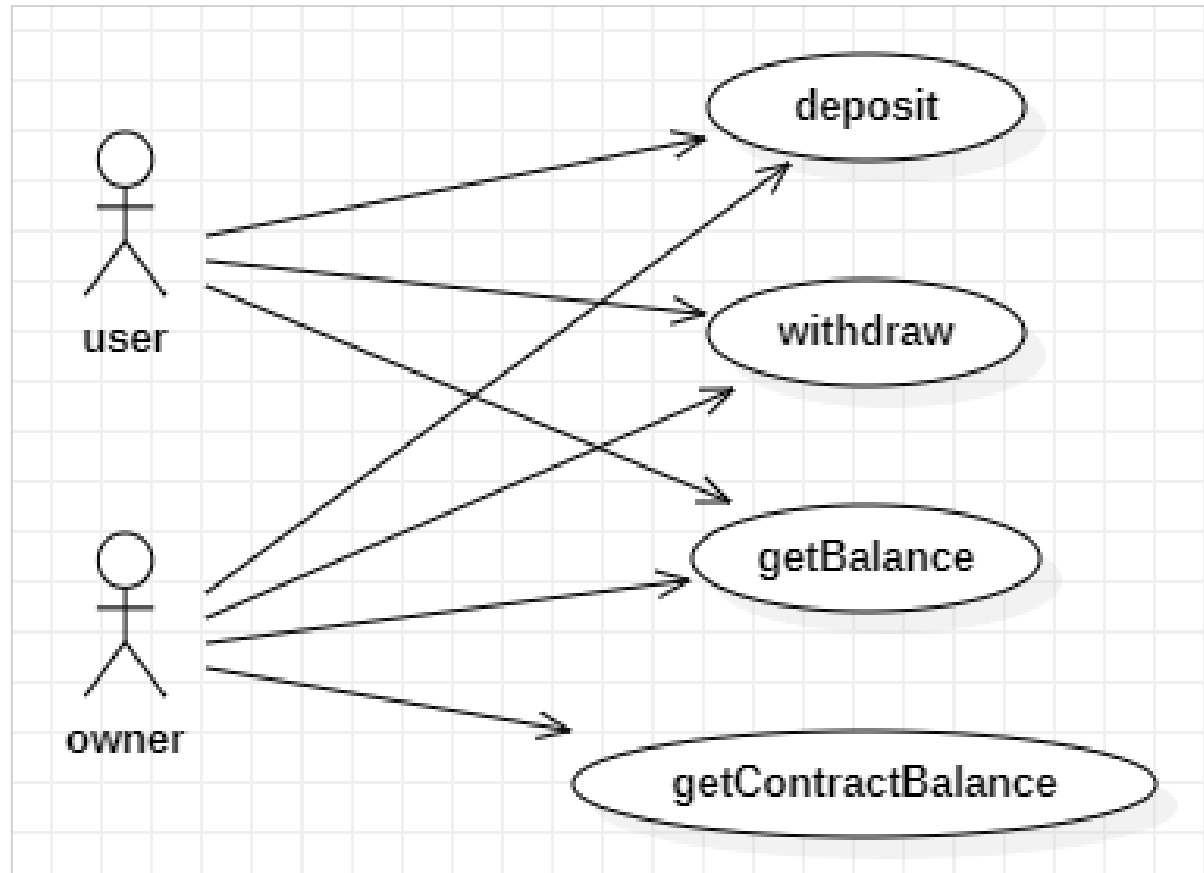
```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.20;  
  
contract Counter {  
  

```

```
}
```



## ■ Use Case Diagram



- 데이터

- 계정별 잔고

- 이벤트

- event Deposit: 주소와 금액
  - event Withdrawal: 주소와 금액

- 함수

- deposit(): 계좌에 이더를 입금, Deposit 이벤트
  - withdraw(): 계좌에서 이더를 출금, Withdrawal 이벤트
  - getBalance(): 호출자 본인의 계좌 잔고 확인
  - getContractBalance(): 컨트랙트의 잔고를 확인, 이 함수는 소유자에게만 허용

## ■ 데이터와 이벤트

```
contract Bank {  
  
    event Deposit  
    event Withdrawal  
  
    constructor() {  
  
    }  
  
    modifier onlyOwner() {  
  
    }  
}
```

## ■ 함수

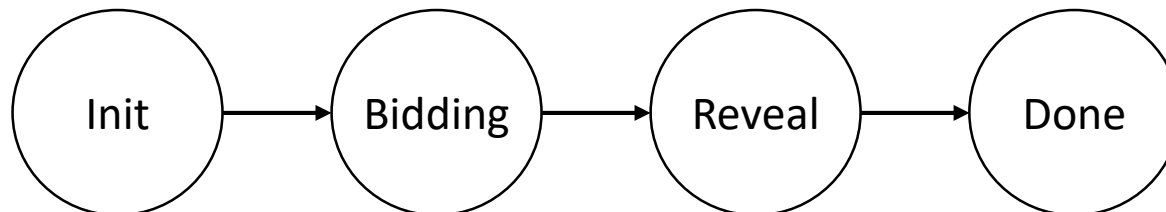
```
function deposit() public payable {  
  
}  
  
function withdraw(uint256 amount) public {  
  
}  
  
function getBalance() public view returns (uint256) {  
  
}  
  
function getContractBalance() public view onlyOwner returns (uint256) {  
  
}  
}
```

## ■ 테스트

- deposit(): 몇 개의 계정을 이용한 예금
  - 이벤트 확인
- getBalance(): 현재 자신의 예금액 확인
- withdraw()
  - 예금보다 적은 금액 출금
    - 이벤트 확인
  - 예금보다 큰 금액 출금
- getContractBalance()
  - 배포자가 확인
  - 배포자가 아닌 다른 계정이 확인

## ■ 문제 정의(1/2)

- 경매 Beneficiary(수혜자) 설정: 컨트랙트 배포자로 설정
- 경매 단계 제어: Beneficiary가 담당
  - 경매 단계: Init, Bidding(입찰), Reveal(입찰가 공개), Done(종료)
- 경매 시작 (Bidding 단계): Beneficiary가 경매를 시작(Bidding 단계로 변경)
  - 입찰자는 한번씩 입찰할 수 있음
  - 블라인드 입찰: 보안이 보장되도록 입찰가와 비밀번호를 이용한 해시값 제출
  - 입찰자는 입찰시 입찰가 이상의 예치금을 전송해야 함




## ■ 문제 정의(2/2)

- 입찰가 공개 (Reveal 단계): Beneficiary가 Reveal단계로 변경
  - 각 입찰자는 자신의 입찰가를 공개한다(입찰가와 비밀번호 전송)
  - 컨트랙트는 입찰가와 비밀번호의 해시값이 제출된 해시값과 일치하는지 확인
    - 해시값이 다르면 예치금을 되돌려준다
    - 해시값이 일치하면 예치금에서 입찰가를 뺀 나머지는 되돌려준다
  - 최고 입찰가와 비교
    - 최고 입찰가보다 작으면 입찰 탈락자의 입찰금 반환을 위한 매핑에 추가
- 경매 종료 (Done 단계): Beneficiary가 Done 단계로 변경하고 경매를 종료
  - 최고 입찰가를 Beneficiary에게 전송
  - 입찰 탈락자는 자신의 입찰금을 출금 (withdraw 함수)

## ■ 웹 UI

### ETH - Blind Auction

May the highest bidder win!



#### *Bid on this antique trophy!*

Enter blinded bid:

Enter deposit amount:

Bid

Reveal your bid:

Enter the OTP:

Reveal

Current phase: Auction Ended

Move to next phase

Withdraw

Show winning bid

단위(이더)

단위(이더)

비밀번호



## ■ 함수

- advancePhase(): [Move to next phase] 버튼
- bid(bytes32 blindBid): [Bid] 버튼
- reveal(uint value, bytes32 secret): [Reveal] 버튼
- withdraw(): [Withdraw] 버튼
- auctionEnd(): [Show winning bid] 버튼

## ■ 개발 환경

- **Node.js**: 클라이언트 인터페이스를 위한 웹서버
  - <https://nodejs.org/en>
- **npm**: 패키지 매니저
- **트러플**: 이더리움 스마트 컨트랙트 개발 및 배포를 위한 프레임워크
  - <https://trufflesuite.com/truffle/>
- **가나쉬**: 개발 및 테스트 목적으로 사용할 수 있는 개발용 이더리움 블록체인
  - <https://trufflesuite.com/ganache/>
- 브라우저/웹클라이언트: 크롬과 **메타마스크**
  - 메타마스크는 특정한 블록체인에 연결해 계정을 관리(디지털서명 등)하도록 지원
  - <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbepgpgknn>

## ■ BlindAuction-Dapp.zip 압축 풀기

### ■ blindauction-app(UI부분)

- src
  - css
  - fonts
  - img
  - js
    - [app.js](#) (스마트컨트랙트와 웹UI를 연결)
  - [index.html](#) (웹UI)
- [index.js](#) (서버 시작을 위한 스크립트)
- package.json

### ■ blindauction-contract(스마트컨트랙트)

- build
- contracts
  - [BlindAuction.sol](#)
- migrations
- truffle-config.js

- 블라인드경매를 위한 해시값

- 입찰가와 비밀번호를 이용한 해시값 구하기 → 이 값으로 입찰

```
contract Khash {  
    bytes32 public hashedValue;  
  
    function hashMe(uint value, bytes32 password) public {  
        hashedValue = keccak256(abi.encodePacked(value, password));  
    }  
}
```

- Reveal 단계에서는 입찰가와 비밀번호 입력 → 컨트랙트에서 keccak256 함수를 이용하여 맞는지 확인

```
(예)  
value: 20  
password: 0x2000000000000000000000000000000000000000000000000000000000000000  
  
hashedValue: 0x60ba2a76256b7bcc6d87ecd1daa6c86d3f9f523dabc2b62fce23a855827d3b34
```

## ■ BlindAuction 스마트컨트랙트

```
contract BlindAuction {
    struct Bid {
        bytes32 blindedBid;
        uint deposit;
    }

    // Init - 0; Bidding - 1; Reveal - 2; Done - 3
    enum Phase {Init, Bidding, Reveal, Done}

    // 소유자
    address payable public beneficiary;

    // 최고 입찰가, 입찰자
    address public highestBidder;
    uint public highestBid = 0;

    // 입찰자의 입찰 해시값과 예치금에 대한 매핑
    mapping(address => Bid) public bids;

    // 입찰금 반환을 위한 매핑
    mapping(address => uint) pendingReturns;

    Phase public currentPhase = Phase.Init;
```

## ■ BlindAuction 스마트컨트랙트

```
// Events
event AuctionEnded(address winner, uint highestBid);
event BiddingStarted();
event RevealStarted();
event AuctionInit();

// Modifiers
// 경매 단계별로 실행 가능한 함수 제한하는 Modifier
// 소유자(beneficiary)만 실행 가능하게 제어하는 Modifier

constructor() {
}

// 경매 단계 변경, 각 단계에 맞게 이벤트 발생(BiddingStarted, RevealStarted, AuctionInit)
// AuctionEnded 이벤트는 auctionEnd() 함수에서 발생
function advancePhase() public {
}

// 입찰 정보 저장
function bid(bytes32 blindBid) public {
}
```

## ■ BlindAuction 스마트컨트랙트

```
// 입찰가와 비밀번호 확인
// 예치금에서 입찰가를 뺀 나머지는 되돌려준다
// 최고 입찰가를 비교
// 최고 입찰가보다 작으면 입찰 탈락자의 입찰금 반환을 위한 매핑을 추가
function reveal(uint value, bytes32 secret) public {
}

// 낙찰되지 않은 입찰금 반환
function withdraw() public {
}

// 소유자(수혜자)에게 가장 높은 입찰가를 보내고 경매를 종료
// AuctionEnded 이벤트 발생
function auctionEnd() public {
}
}
```

## ■ 실행 과정

### 1. 스마트 컨트랙트 작성

### 2. 가나쉬 실행: 테스트 블록체인 네트워크

### 3. 스마트 컨트랙트 컴파일&배포(truffle-config.js: 가나쉬에 배포 설정)

- `cd blindauction-contract`
- `truffle migrate`

### 4. 웹서버(node.js) 실행

- `cd blindauction-app`
- `npm install`
- `npm start`



## ■ 실행 과정

### 5. 웹브라우저(메타마스크) 실행

- 브라우저에서 localhost:3000 접속
- 메타마스크를 가나쉬 네트워크에 연결
- 가나쉬 계정 가져오기(3개): 편의상 Account2, Account3, Account4 라고 함
  - Account2: Beneficiary
  - Account3: 입찰자1
  - Account4: 입찰자2
- 가나쉬 계정(3개) localhost:3000 에 연결

## ■ 실행 과정

### 6. 테스트

- Account2가 [Move to next phase] 버튼 클릭하여 Bidding 단계로 변경
- Account3가 입찰(예치금: 50이더)
  - 입찰가 20(이더), 비밀번호로 만든 해시값 입력
- Account4가 입찰(예치금: 50이더)
  - 입찰가 30(이더), 비밀번호로 만든 해시값 입력
- Account2가 [Move to next phase] 버튼 클릭하여 Reveal 단계로 변경
- Account3가 Reveal
- Account4가 Reveal
- Account2가 [Move to next phase] 버튼 클릭하여 경매 종료 단계로 변경
- Account2가 [Show winning bid] 버튼 클릭 경매 결과 확인 및 수혜자에게 낙찰금 전송
- Account3가 [Withdraw] 버튼 클릭하여 자신의 입찰금 반환

## ■ 테스트용 값

### ■ Account3

- Bid

- 블라인드 입찰값: 0x60ba2a76256b7bcc6d87ecd1daa6c86d3f9f523dabc2b62fce23a855827d3b34
- 예치금: 50

- Reveal

- 입찰값: 20
- OTP: 0x2000000000000000000000000000000000000000000000000000000000000000

### ■ Account4

- Bid

- 블라인드 입찰값: 0x4a4ed630c9ae66b1bafcce494b7148750fc65e97ea91602b10e4ffa108ad1ce8
- 예치금: 50

- Reveal

- 입찰값: 30
- OTP: 0x3000000000000000000000000000000000000000000000000000000000000000