

Be as proud of Sogang as Sogang is proud of you

블록체인 스마트컨트랙트



서강대학교
SOGANG UNIVERSITY

- 스마트컨트랙트 예제
 - 크라우드펀딩 스마트컨트랙트
 - 스마트컨트랙트의 이름과 주소를 관리하는 계약

■ 크라우드펀딩

- 모금만을 목적으로 이더를 모금
- 투자자는 계약에 이더를 보내는 거래를 통해 투자
- 계약 소유자
 - 모금 활동의 마감일과 목표액 설정
 - 마감일 시점에 목표액이 달성되면 계약의 소유자에게 모금된 이더 송금
 - 마감일 시점에 목표액이 달성되지 않으면 투자자에게 이더를 돌려줌

■ 데이터

■ 투자자 구조체와 매핑

```
struct Investor {  
    address addr;    // 투자자 주소  
    uint amount;    // 투자액  
}
```

```
mapping (uint => Investor) public investors;    // 투자자 추가할 때 key 증가
```

■ 그외

```
address public owner;    // 컨트랙트 소유자  
uint public numInvestors;    // 투자자 수  
uint public deadline;    // 마감일  
string public status;    // 모금활동 상태(Funding, Campaign Succeeded, Campaign Failed)  
bool public ended;    // 모금 종료여부  
uint public goalAmount;    // 목표액  
uint public totalAmount;    // 총 투자액
```

■ 소유자만 실행하게 하는 modifier

```
modifier onlyOwner () {  
  
}
```

■ 생성자

- 매개변수: 모금기간(초단위), 목표액(이더 단위)
- 마감일 설정: 현재(block.timestamp) + 모금기간(_duration)

```
constructor(uint _duration, uint _goalAmount) {  
    owner = msg.sender;  
  
    deadline = block.timestamp + _duration;  
    goalAmount = _goalAmount * 1 ether;  
    status = "Funding";  
    ended = false;  
  
    numInvestors = 0;  
    totalAmount = 0;  
}
```



- 투자자가 투자할 때 호출하는 함수
 - 투자금 송금과 함께 호출
 - 모금이 끝났다면 처리 중단
 - 투자자 정보를 매핑에 저장
 - 투자자수와 투자 총액 업데이트

```
function fund() public payable {
```

```
}
```

- 소유자가 모금을 종료할 때 호출하는 함수
 - 소유자만 호출 가능
 - 모금이 끝나거나 마감이 지나지 않았다면 처리 중단
 - 목표액 달성 여부 확인 즉 모금 성공/실패에 따라 송금
 - 성공이면 소유자에게 모든 이더를 송금
 - 실패면 각 투자자에게 투자금을 돌려줌
 - 상태 정보 변경
 - status = "Campaign Succeeded" or "Campaign Failed"
 - ended = true

- 소유자가 모금을 종료할 때 호출하는 함수

```
function checkGoalReached () public onlyOwner {
```

```
}
```


■ 테스트

■ 모금 성공(10이더 목표)

- ACCOUNT2 7이더 투자: fund()
- ACCOUNT3 3이더 투자: fund()
- 마감시간 전에 checkGoalReached 호출
- 마감시간 후 checkGoalReached 호출

■ 모금 실패(10이더 목표)

- ACCOUNT2 3이더 투자: fund()
- ACCOUNT3 4이더 투자: fund()
- 마감시간 전에 checkGoalReached 호출
- 마감시간 후 checkGoalReached 호출

■ 배포자가 아닌 계정이 checkGoalReached 호출

크라우드펀딩 스마트컨트랙트

checkGoalReac...

fund

deadline

0: uint256: 1699188480

ended

0: bool: true

goalAmount

0: uint256: 1000000000000000000

investors

1

0: address: addr 0x5c6B0f7Bf3E7ce046039Bd8FABdFD3f9F5021678

1: uint256: amount 3000000000000000000

numInvestors

0: uint256: 2

owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4



status

0: string: Campaign Succeeded

totalAmount

0: uint256: 1000000000000000000

ACCOUNT +



0x5B3...eddC4 (109.99999999)  

0x5B3...eddC4 (109.9999999999999970394 ether)

0xAb8...35cb2 (92.99999999999999887369 ether)

0x4B2...C02db (96.999999999999998982 ether)

ACCOUNT +

0x5B3...eddC4 (99.99999999)  

0x5B3...eddC4 (99.999999999999998945554 ether)

0xAb8...35cb2 (99.99999999999999887369 ether)

0x4B2...C02db (99.999999999999998982 ether)

checkGoalReac...

fund

deadline

0: uint256: 1699189157

ended

0: bool: true

goalAmount

0: uint256: 1000000000000000000

investors

uint256

numInvestors

0: uint256: 2

owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

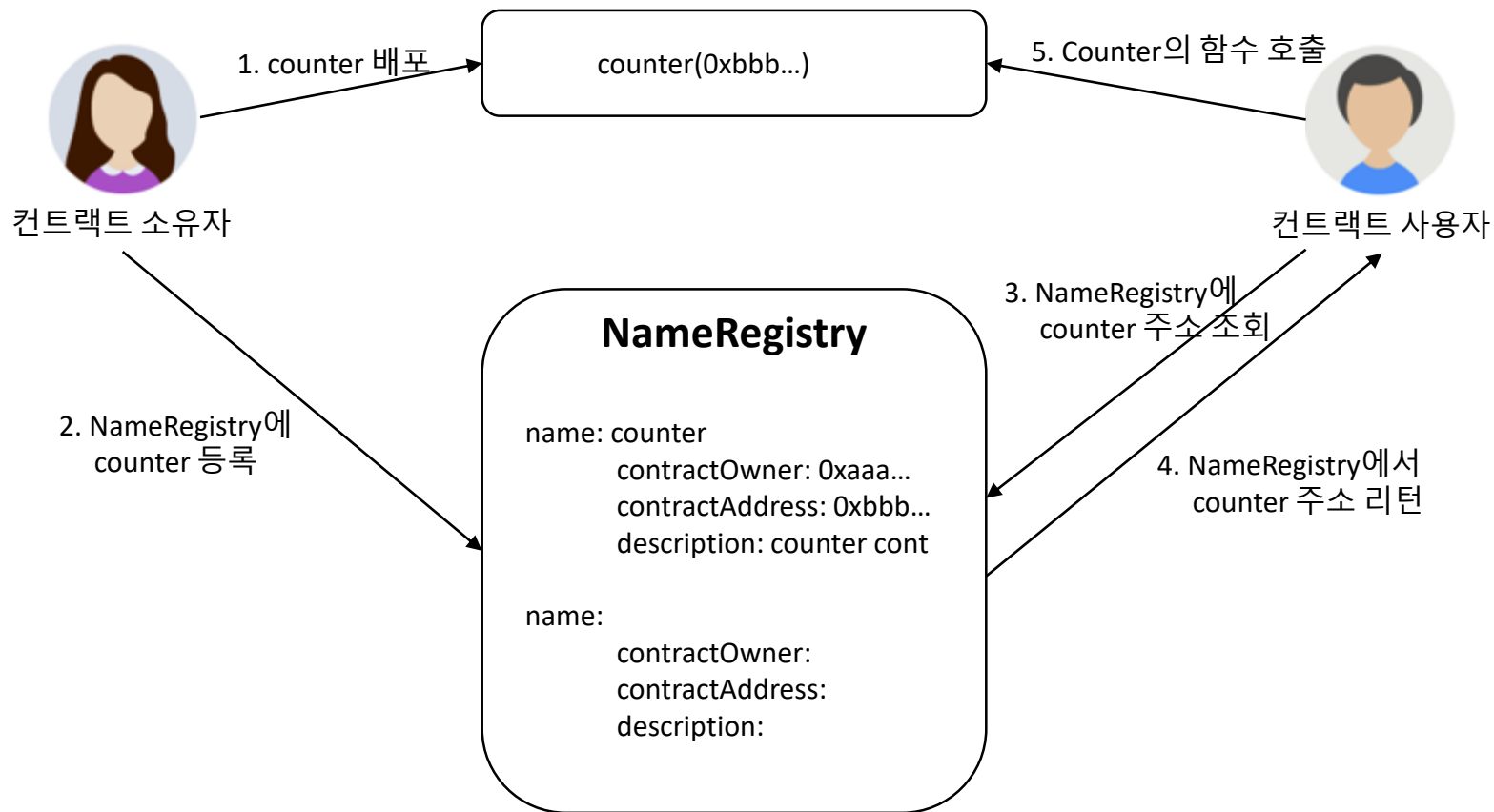
status

0: string: Campaign Failed

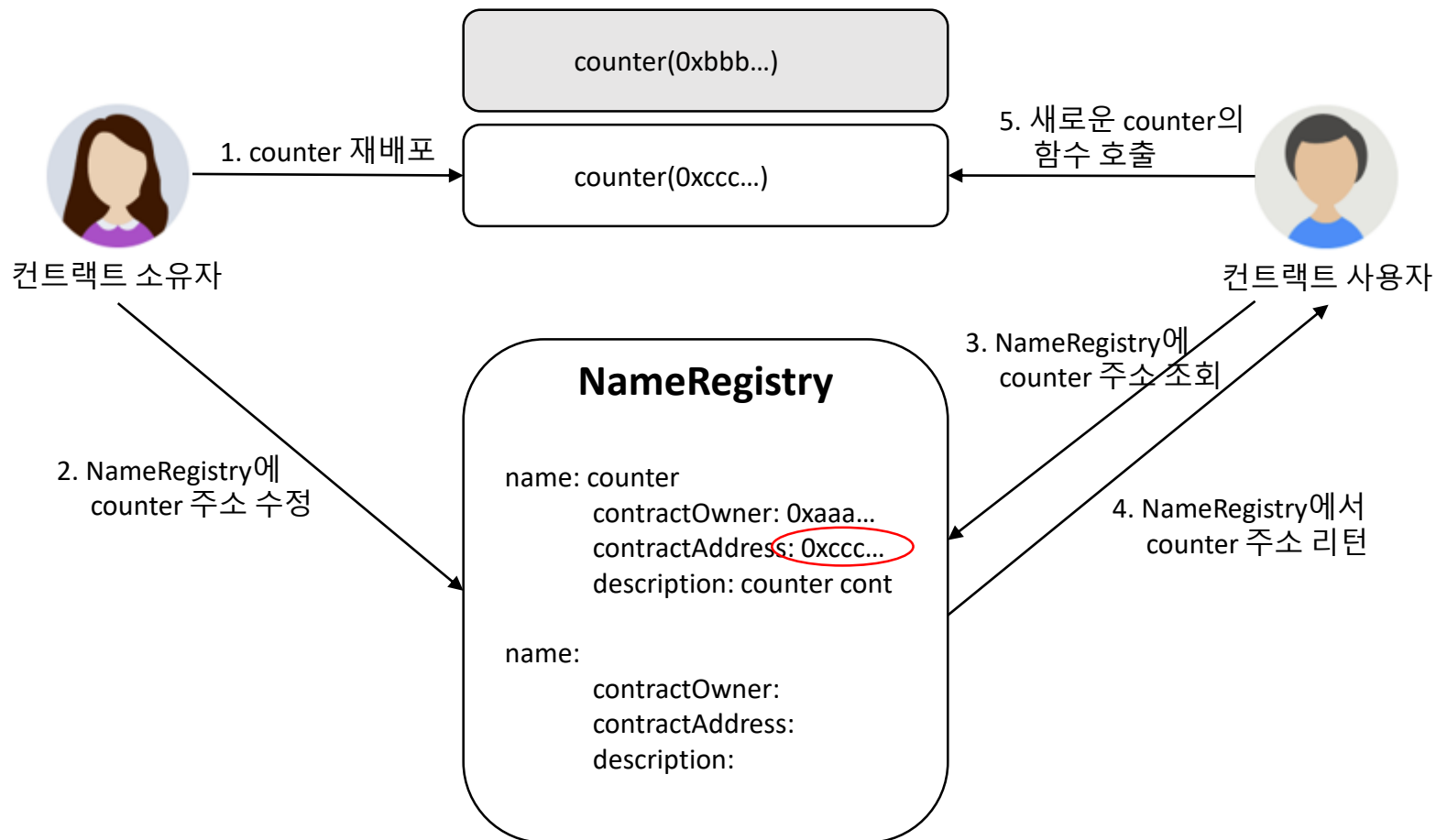
totalAmount

0: uint256: 7000000000000000000

■ 스마트컨트랙트의 이름과 주소를 관리



■ 스마트컨트랙트의 이름과 주소를 관리



■ 스마트컨트랙트의 이름과 주소를 관리

■ 스마트컨트랙트 등록정보

- 소유자(배포자)
- 주소
- 설명

- 등록된 스마트컨트랙트를 사용하기 위해 사용자가 주소를 조회하면 주소 전달
- 스마트컨트랙트 등록자는 스마트컨트랙트 변경시 새롭게 배포하고 새로운 주소로 업데이트
- 스마트컨트랙트가 변경되어도 새롭게 등록된 스마트컨트랙트 주소를 전달하여 사용자 이용이 가능하게 함
- 등록한 스마트컨트랙트 정보는 등록한 소유자만 변경 가능

■ 데이터

- 컨트랙트 정보를 나타낼 구조체
- 등록된 컨트랙트들을 저장할 매핑
- 등록된 컨트랙트 수

```
// 컨트랙트 정보를 나타낼 구조체
struct ContractInfo {
    address contractOwner;
    address contractAddress;
    string description;
}

// 등록된 컨트랙트 수
uint public numContracts;

// 등록된 컨트랙트들을 저장할 매핑(이름->컨트랙트 정보 구조체)
mapping(string => ContractInfo) public registeredContracts;
```

■ modifier

- 등록된 스마트컨트랙트 정보는 **등록한 소유자**만 변경 가능

```
// 함수를 호출 전 먼저 처리되는 modifier를 정의  
modifier
```

■ 생성자

```
constructor() {  
    numContracts = 0;  
}
```


■ 컨트랙트 소유자 정보 변경과 확인

```
// 컨트랙트 소유자 변경  
function changeOwner(string memory _name, address _newOwner)
```

```
// 컨트랙트 소유자 정보 확인  
function getOwner(string memory _name)
```

■ 컨트랙트 어드레스 변경과 확인

```
// 컨트랙트 어드레스 변경  
function setAddr(string memory _name, address _addr)
```

```
// 컨트랙트 어드레스 확인  
function getAddr(string memory _name)
```

■ 컨트랙트 설명 변경과 확인

```
// 컨트랙트 설명 변경  
function setDescription(string memory _name, string memory _description)
```

```
// 컨트랙트 설명 확인  
function getDescription(string memory _name)
```

■ 테스트

- Account1이 NameRegistry 배포
- Account2가 가상의 컨트랙트(bank) 배포
- Account2가 컨트랙트 NameRegistry에 등록
 - Account1이 bank 컨트랙트 설명 변경/확인
 - Account2가 bank 컨트랙트 소유자 정보 변경/확인
 - Account2가 bank 컨트랙트 어드레스 정보 변경/확인(재배포 후)
 - Account2가 bank 컨트랙트 설명 변경/확인
- Account1이 Account2가 등록한 bank 컨트랙트 삭제
- Account2가 등록한 bank 컨트랙트 삭제

registerContract

_name: bank

_contractAddress: 0xa131AD247055FD2e2aA8b156A11

_description: bank contract

Calldata Parameters **transact**

getAddr bank

0: address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

getDescription bank

0: string: bank contract

getOwner bank

0: address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2