

Se-Chain Code Description

서강대학교
서강미래기술원
글로벌 핀테크 연구소

고덕윤
maniara.k@gmail.com

01 교육용 블록체인 소개

- 블록체인의 가장 기본적인 형태 구현
 - P2P broadcast (transaction, block)
 - PKI algorithm
 - Simple smart contract
- Named Se-Chain
- Python code
- Simple UI
- Simple Smart contract
- <https://github.com/maniara/SeChain>

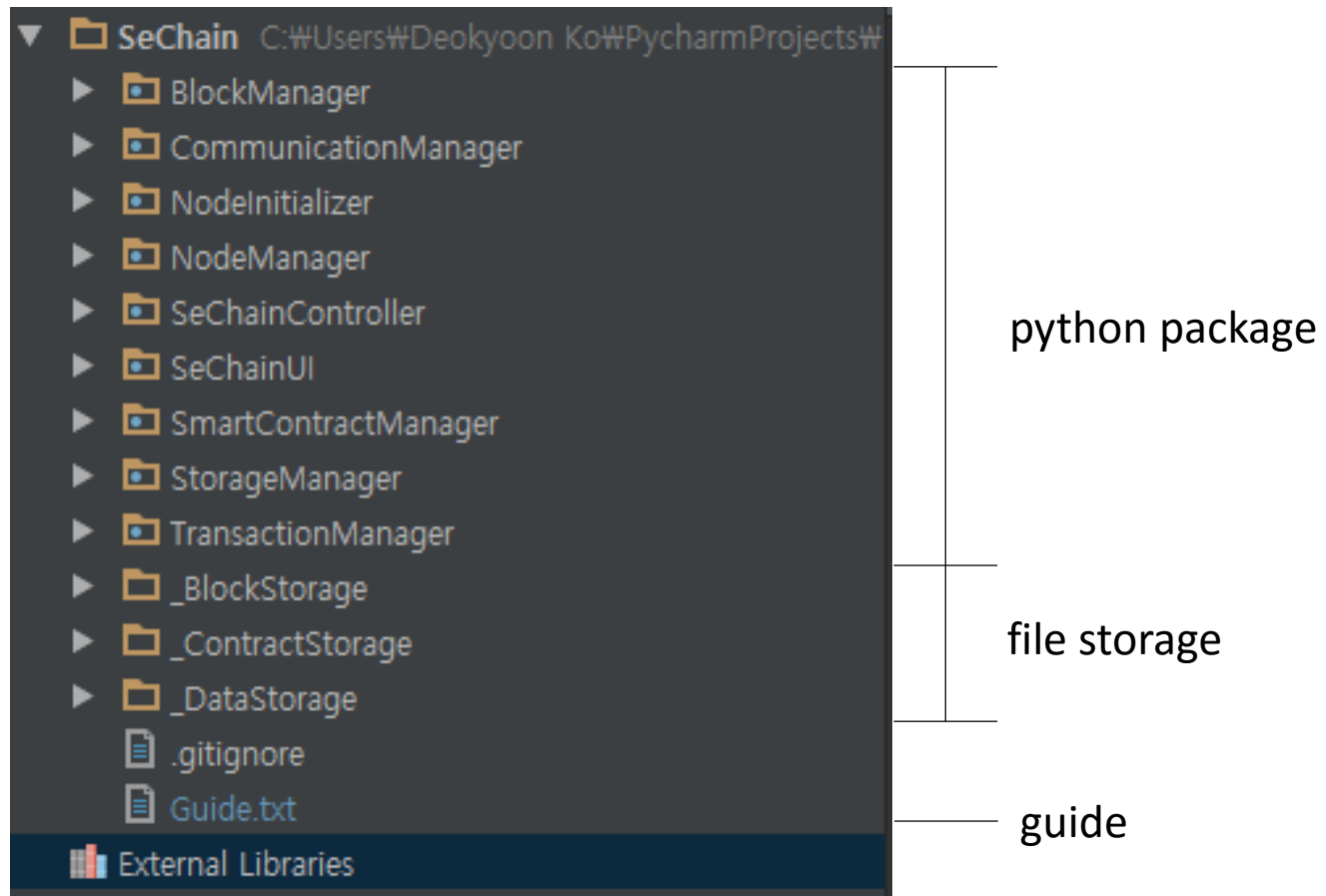
02 시작하기

- 기본적으로 두 대 이상의 PC가 필요하며,
- 모두 동일한 작업을 필요로 함

- Python2.7 설치 : www.python.org
- PyCharm 설치 : www.jetbrains.com/pycharm/
 - Community edition : Free

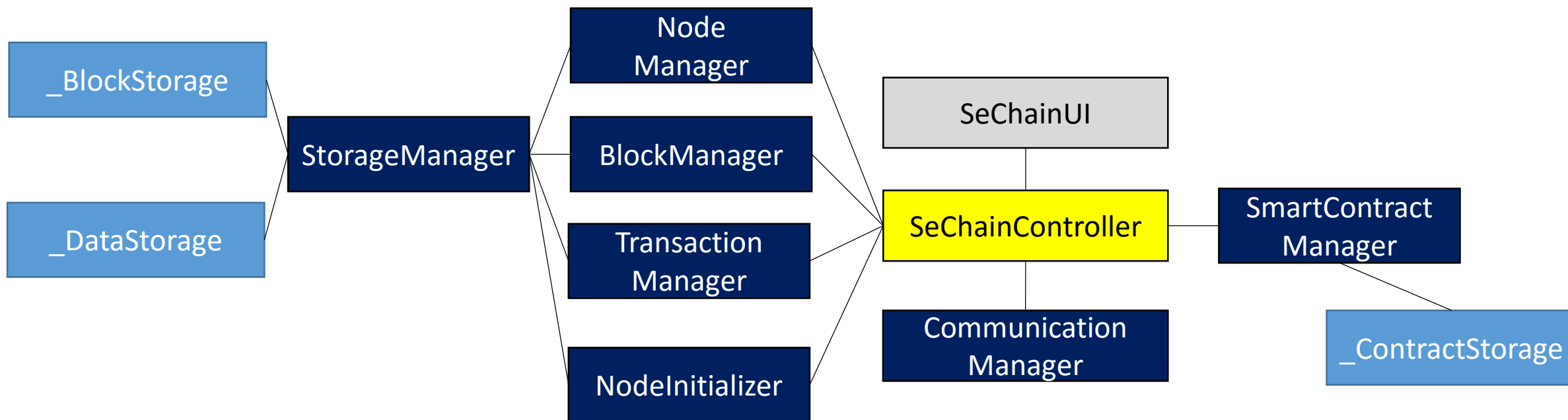
- 라이브러리 설치
 - Guide.txt 참조

03 폴더 구조



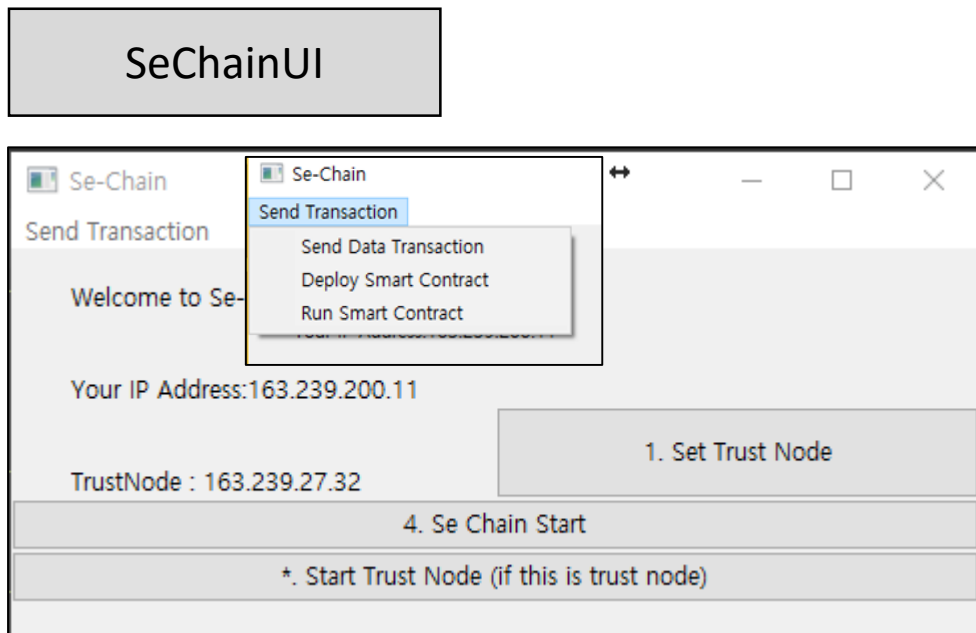
04

Brief architecture



- 모듈
- 메인 컨트롤러
- 사용자 뷰
- 파일 데이터 베이스

05 SeChainUI



- SeChain UI를 그림
 - SeChainLauncher.py : UI를 시작
 - MainUI : UI 구성 파일
- TrustNode : 블록체인의 신뢰 노드
 - 모든 블록을 저장하고 있음
 - 모든 노드 리스트를 저장하고 있음
 - 다른 노드 시작 시 동기화를 수행함
 - 항상 On 상태여야 함
 - *.Start Trust Node 버튼으로 시작
- 트랜잭션 송신 : 노드를 시작하여야 가능함
 - 데이터 트랜잭션 송신
 - 스마트 계약 배포
 - 스마트 계약 실행

06 SeChain Controller

SeChainController

- SeChain 전체 컨트롤을 담당함
 - Perperty.py : 블록체인 세팅 값 저장소 (전역 변수 저장소)
 - FunctionAPI.py : 블록체인 운영 시 UI(SeChainUI.MainUI)가 호출할 수 있는 API 내역
 - `send_transaction(...message...)` : 트랜잭션 송신
 - `deploy_contract(...contract_source[source_path, initial_arguments]...)` : 스마트 계약 배포
 - `run_contract (...contract_data[contract_address, function_name, arguments]...)` : 배포된 스마트 계약 실행
 - 현재 특정 수 이상의 트랜잭션이 쌓이면(BlockManager.BlockGenerator.
`check_block_generation_condition`), 블록 생성
 - MainController.py : UI(SeChainUI.MainUI)가 노드를 시작하면 호출하는 컨트롤러
 - `initiate_node` : 노드 실행 전 `trust_node` 와 블록과 노드 리스트 동기화
 - `node_start` : 노드를 실행함
 - IP 주소, 2PKs 를 생성하고 이를 전체 노드에 알림
 - `CommunicationManager.Receiver` 스레드를 시작함

07 Communication Manager Module

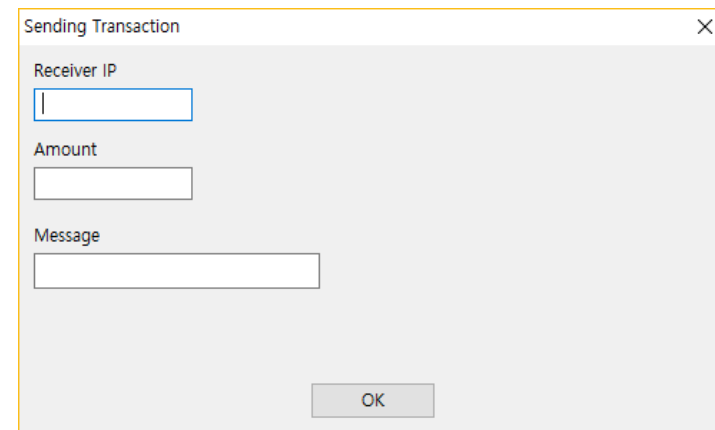
Communication Manager

- P2P 브로드 캐스팅 통신을 담당함
 - Sender.py : _DataStorage에 저장된 노드 리스트에게 특정 메시지를 보냄
 - send(ip_address, message, port)
 - send_to_all_node(message)
 - Receiver.py : 별도의 스레드로 실행하며 각종 메시지를 처리함
 - 들어오는 메시지의 타입을 판단하여, 각 모듈의 일 처리를 요청함
 - B : 블록, BlockManager.BlockVerifier 에게 검증을 요청함
 - C : 블록 동기화를 요청 받은 경우, 받지 못한 블록을 보내주고, 마지막엔 타입 Q를 보냄
 - N : 새로운 노드가 들어오는 경우, NodeManager를 통해 노드리스트에 추가함
 - t, ct, rt ; 트랜잭션(스마트 계약 배포, 스마트 계약 실행) 의 경우 트랜잭션 리스트에 추가함
 - RN : 새로 접속한 노드에서 노드 리스트 동기화를 요청한 경우, 마지막엔 QN을 보냄

08 Transaction Manager

Transaction Manager

- 트랜잭션을 관리함
 - Transaction.py : 트랜잭션 entity 클래스
 - 불변 영역 : type, time_stamp, ip_address, sender_public_key, contract_datas
 - 가변 영역 : ip_address, data[ValueData(ip_address, receiver, amount, msg)]
 - 관리 영역 : message(메시지의 json)
 - CryptoController : 암호/복호화
 - dataEncode(msg, privateKey) / dataDecode(bcipher, publicKey)
 - TransactionController : 트랜잭션 관리
 - create_transaction : 트랜잭션 엔티티 생성
 - ValueData : 별도 데이터 저장을 위한 entity 클래스
 - 비즈니스에 따라 원하는 데이터 엔티티 형태로 변경 가능
 - But 변경 시, MainUI의 변경이 불가피함



A screenshot of a 'Sending Transaction' dialog box. It has a title bar with a close button (X). The dialog contains three input fields: 'Receiver IP' (a small text box), 'Amount' (a small text box), and 'Message' (a larger text box). At the bottom right, there is an 'OK' button.

09 Block Manager

BlockManager

- 블록을 관리함
 - Block.py : 블록 entity 클래스
 - type, block_id('B' +time_stamp), time_stamp, previous_block_hash(id), transactions[]
 - BlockGenerator.py : 블록 생성을 담당
 - check_block_generation_condition
 - 블록 생성 조건을 검사
 - **비즈니스에 맞게 변경 필요**
 - 현재는 트랜잭션 수를 검사하여 일정 수 이상이면 블록 생성
 - generate_block : 블록 객체 생성 담당
 - 이전 블록 해시 코드 추출 및 블록 로컬 저장
 - BlockVerifier.py : 다른 노드로 부터 블록을 받으면, 이를 검증함
 - 현재는 미구현 상태
 - **비즈니스에 맞게 변경 필요**

10 Node Manager

Node Manager

- 노드 정보 관리를 담당함
- node.py : 노드 정보 저장을 위한 entity
 - ip_address : 모든 노드는 ip 주소를 id로 함
 - public_key, private_key
- KeyGenerator.py : 2PKs 를 생성
- NodeController.py : 노드 생성을 관리함
 - get_node : node 객체를 생성하고 이를 리턴함
 - send_my_node : 나의 노드 객체를 다른 노드에게 알림
 - add_new_node : 나의 로컬 데이터 베이스에 새로운 노드를 추가함

11 Node Initializer

NodeInitializer

- 노드가 블록체인에 참여할 때, 그 동안의 장부 내역과 노드 리스트를 신뢰노드(`trust_node`)와 동기화함
 - 신뢰 노드의 응답 처리는 `CommunicationManager.Receiver` 에서 처리함
 - `BlockSynchronizer.py` : 블록 동기화 요청 및 응답 처리
 - `NodeListSynchronizer.py` : 노드 리스트 요청 및 응답 처리

12 Storage Manager & File Storage

StorageManager

- 로컬 파일 스토리지를 관리하는 모듈
 - FileController.py : 파일 스토리지에서 파일을 읽고 쓰는 기능을 수행함

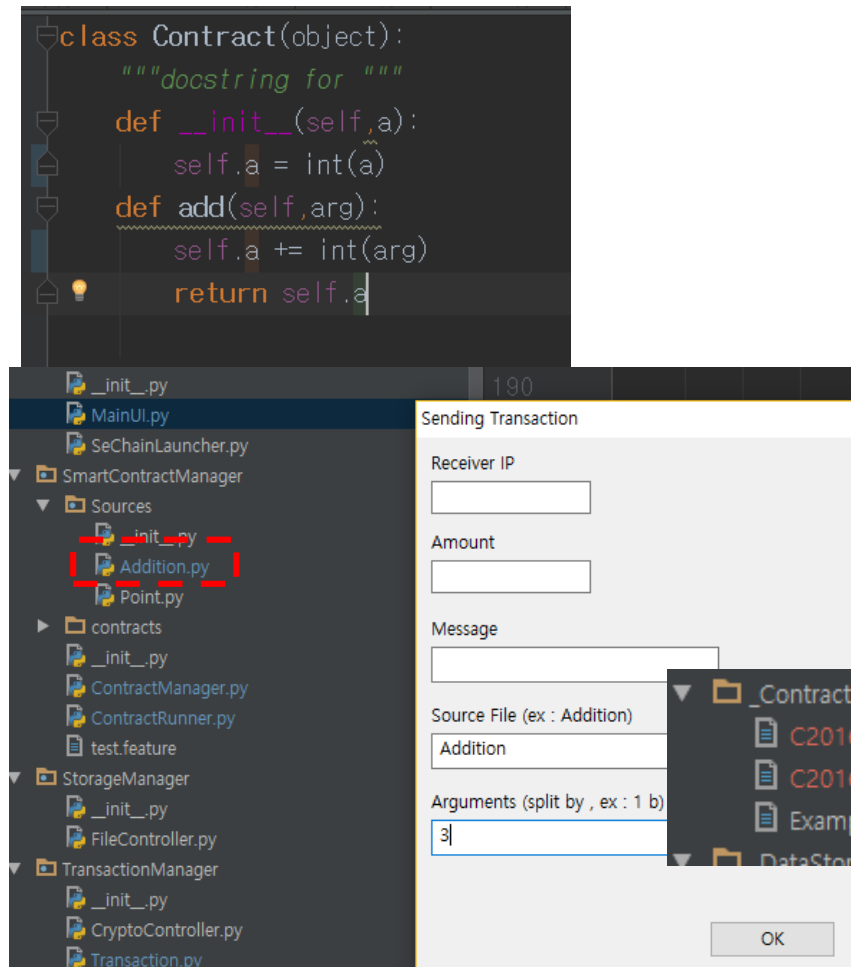
_BlockStorage

- 블록을 저장하는 폴더
- 블록 ID를 파일 명으로 함

_DataStorage

- NodeInfo.txt : 노드 리스트 엔티티의 json을 저장함
 - *to do : 다른 노드의 private key도 저장하므로 이를 처리할 수 있는 방안이 필요함*
- Transactions.txt : 아직 블록화가 되지 않은 트랜잭션의 json을 저장함
 - 블록 생성과 동시에 초기화됨

13 Smart contract

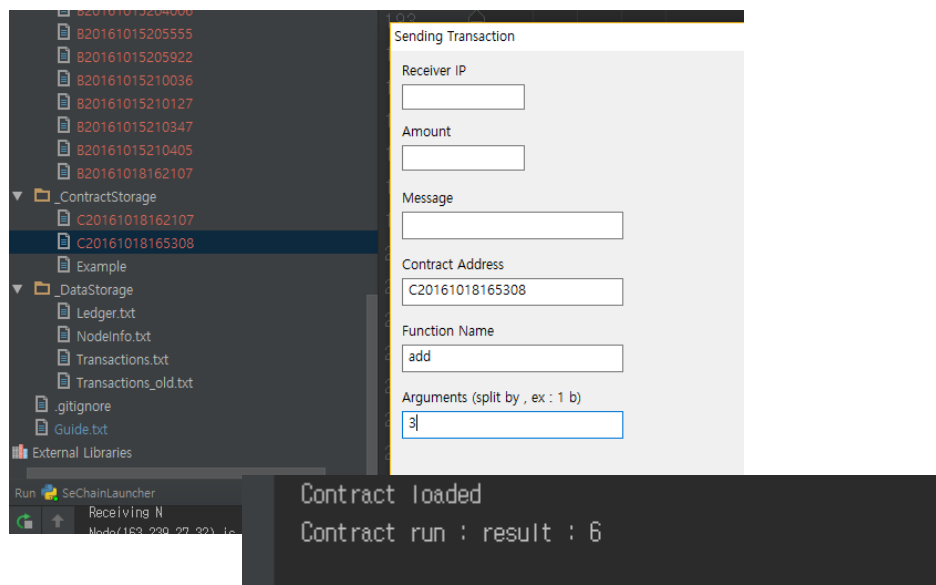


- 스마트 계약 배포 방법
 - SmartContractManager/Sources 에 py 파일을 작성함.
 - 모든 노드의 동일한 위치에 소스코드를 복사함
 - to do : 소스코드 복사 메커니즘 구현*
 - Deploy Smart Contract 메뉴 선택 후
 - 파일 명을 적음 (.py는 뺀)
 - 생성자에 필요한 인자를 공백으로 구분하여 넣음
- 스마트 계약 소스 트랜잭션 배포
 - 블록 검증 시 소스코드를 직렬화(serialize)함
 - 블록 검증 후 _ContractStorage 에 저장

14 Smart contract

```
class Contract(object):  
    """docstring for """  
    def __init__(self,a):  
        self.a = int(a)  
    def add(self,arg):  
        self.a += int(arg)  
        return self.a
```

- 스마트 계약 실행 방법
 - Run Smart Contract 선택 후,
 - 계약 주소, 함수명, 인자를 넣고 실행
 - 콘솔창에 리턴 값 출력됨



15 Smart Contract Manager

SmartContract
Manager

_ContractStorage

- 스마트 계약 배포 및 실행을 관리함
 - ContractManager.py
 - CT : 배포일 경우 ContractRunner.makeContract() 실행
 - RT : 배포일 경우 ContractRunner.run() 실행
 - ContractRunner.py
 - makeContract : 계약 코드를 직렬화 후 저장소(_ContractStorage)에 기록
 - runContract : 특정 코드를 원격 import 후 실행

16 실행 방법

1. 신뢰 노드 (Trust Node)에서 SeChainLauncher.py를 실행
2. Run Trust Node 버튼 클릭
3. 일반 노드에서 SeChainLauncher.py를 실행
4. Se Chain Start 버튼 클릭
5. Send Transaction 메뉴에서 원하는 기능을 수행