은행업무 프로그램

사람	희현 은빈 hyoeun jeong (백엔드)배근호ⓖ GY K

💻 SmartBankSystem 팀 프로젝트 가이드

💁‍♀️ 팀 구성: 총 6명!

각자 하나의 기능을 맡아 **클래스 단위**로 개발하며, `main.py`에서 호출해 통합합니다.

.py 파일을 공유 할 때 파일 명은 '날짜_이름_담당번호.py'로 공유해주세요!

ex. 1번 로그인 관리를 맡았다면) 250518_배주완_1.py

🧩 1️⃣ LoginManager – 로그인 & 공인인증서 관리

- `issue_certificate()`

  → 공인인증서 발급 (ID + 비밀번호 등록)

- `verify_password_format(password)`

  → 비밀번호 규칙 검사

  ✅ 조건: 영문, 숫자, 특수문자 포함 / 8자 이상 / 5회 실패 시 차단

- `login_user(user_id, password)`

  → 로그인 실행, 인증 상태 등록

- `logout_user()`

  → 로그아웃 (세션 초기화)

🛠️ 구현 힌트:

`re` 모듈(정규표현식), 로그인 실패 횟수 저장용 딕셔너리

🧾 2️⃣ AccountManager – 계좌 개설 및 해지

- `create_account()`

  → 4자리 고유 계좌번호 생성, 중복 방지

- `close_account()`

→ 잔액이 0일 경우 계좌 해지

- `verify_account()`

  → 계좌번호 중복 및 주민번호 형식 확인 (`######-#######`)

🛠️ 구현 힌트:

`random`, `set`, `str.split()` 활용

💰 3️⃣ TransactionManager – 입금 & 출금

- `deposit(account_no, amount)`

  → 해당 계좌에 금액 추가

- `withdraw(account_no, amount)`

  → 잔액 확인 후 출금, 부족할 경우 실패

- `get_balance(account_no)`

  → 현재 잔액 확인

🛠️ 구현 힌트:

계좌 딕셔너리에서 `balance` 키 사용

🔁 4️⃣ TransferManager – 계좌 이체

- `transfer(from_account, to_account, amount)`

  → 출금 후 입금 처리

- `verify_recipient(account_no)`

  → 수신자 계좌 확인 및 예금주 이름 일치 여부 확인

- `record_transfer_history()`

  → 이체 내역 저장 (보낼 계좌, 받을 계좌, 금액, 시간 등)

🛠️ 구현 힌트:

`datetime.now()`, 거래내역 `list` 기록

📄 5️⃣ HistoryManager – 거래 내역 조회

- `get_all_history(account_no)`

  → 해당 계좌의 모든 거래 내역 보기

- `search_history_by_date(account_no, start, end)`

→ 날짜 범위로 검색 (최대 1년)

- `get_recent_history(account_no)`

  → 최근 5건 출력

🛠️ 구현 힌트:

`datetime`, 슬라이싱 (`[-5:]`), `csv` 저장은 `csv.writer` 활용 가능

👤6️⃣ UserManager – 사용자 정보 관리

- `register_user()`

  → 이름, 아이디, 비밀번호 등록

- `change_password()`

  → 기존 비밀번호 확인 후 새 비밀번호 설정

- `update_user_info()`

  → 이메일, 휴대폰 번호, 이름 수정 (이름 변경은 1회 제한)

🛠️ 구현 힌트:

변경 횟수 카운트 변수 사용, `email`, `phone` 형식 검사

🧠 공통 규칙 (협업 전 통일!)

항목	예시
클래스명	`PascalCase` → `LoginManager`
함수/변수명	`snake_case` → `change_password`
파일명	`login.py`, `account.py` 등 역할별 명확히

🧪 main.py 예시 구조

```
python
복사편집
def main():
    login = LoginManager()
    account = AccountManager()
    trans = TransactionManager()
    transfer = TransferManager()
```

```
    history = HistoryManager()
    user = UserManager()

    # 예시 호출
    # login.login_user("user01", "pw123!")
    # account.create_account()

if __name__ == "__main__":
    main()
```

📋 테스트용 계좌 정보 (5명)

```
python
복사편집
accounts = {
    "1234": {"name": "김지훈", "balance": 150000, "history": []},
    "2345": {"name": "이서연", "balance": 82000, "history": []},
    "3456": {"name": "박민수", "balance": 230000, "history": []},
    "4567": {"name": "최하은", "balance": 56000, "history": []},
    "5678": {"name": "정우진", "balance": 99000, "history": []}
}
```

- 계좌번호는 문자열 `"1234"` 형식 (숫자처럼 생긴 문자)
- 각 계좌는 `name`, `balance`, `history`로 구성
- 거래 내역은 `history` 리스트에 딕셔너리로 저장