**Design Project A**

Project #1

\* 은행 거래 관리 프로그램을 작성하시오. 자세한 요구사항은 아래와 같다.

**1. System Requirements**

- 은행 거래 관리 프로그램을 구현하기 위한 객체는 App, User, transaction, loan 이다.

- App는 User DB, transaction DB, loan DB를 가지고 있다.

- User는 아이디, 비밀번호, 사용자 이름, 계좌번호, 잔액, 신용등급, 거래내역 캐시를 가지고 있다.

- 사용자가 로그인 할 때, 로그인 한 사용자의 거래내역을 User 클래스의 거래내역 캐시에 저장한다.

- App는 User와 transaction, loan들을 총괄하여 관리하고 필요한 함수를 호출한다.

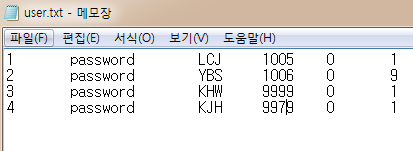
- 프로그램 실행 시에 추가된 거래내역들은 프로그램 종료 시에 일괄적으로 처리하여 사용자의 잔액을 갱신한다.

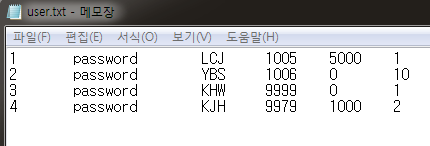
- 프로그램 종료 시에 갱신된 정보들을 파일에 다시 쓴다.

- 메인에서는 오직 App의 run()만을 호출한다.

**2. 공통 기능**

* 데이터 적재 기능 : 프로그램 시작 시, user.txt, transaction.txt, loan.txt로부터 데이터를 불러와 각각 User와 transaction, loan 객체 배열에 저장시킨다.
  + User DB

하나의 User에 대한 정보는 사용자 아이디, 사용자 비밀번호, 사용자 이름, 계좌번호, 잔액, 신용등급과 같은 여섯 가지 필드로 구성되며, 탭을 이용하여 필드를 구분한다. 그리고 개행문자를 통해 각각의 User를 구분한다. 

프로그램 종료 시에 사용자의 거래내역을 계산하여 잔액과 신용등급을 갱신하고, User DB를 저장한다.   


유저의 아이디는 1부터 시작한다.

신용등급은 1~10 이다.

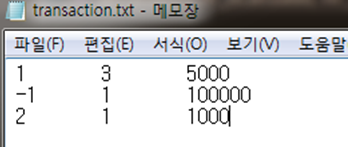
사용자 회원 가입 기능은 구현하지 않아도 되며, User파일에 10명 정도의 사용자 정보를 임의로 생성한다.

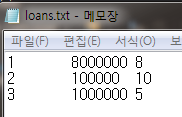
대출을 상환에 한번 성공할 때마다 신용등급이 1씩 올라가고, 잔액부족으로 실패할 때 1씩 떨어진다.

* + Transaction DB

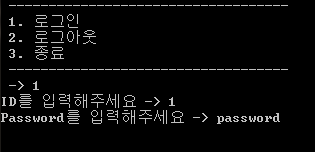
보낸 사람 ID, 받는 사람 ID, 금액과 같은 네 가지 필드로 구성되며, 탭을 이용하여 필드를 구분한다. 그리고 개행문자를 통해 각각의 transaction을 구분한다.

입출금의 경우는 보낸 사람 ID, 받는 사람 ID가 -1이다.



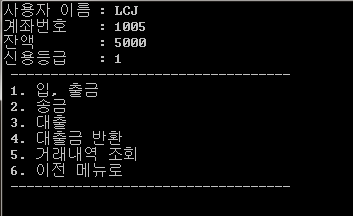
* + loan DB
    - loan DB는 모든 사용자의 대출 내역을 저장한다.
    - 저장된 정보는 대출한 사용자의 아이디와, 대출 금액, 남은 상환 기간을 의미한다. 탭을 이용하여 필드를 구분하고, 개행문자를 통해 각각의 loan을 구분한다.  
      

로그인 : 프로그램은 사용자로부터 아이디와 비밀번호를 입력 받는다. 전체 User의 정보와 입력 값을 비교하여 일치하는 정보를 찾는다. 일치하는 경우 해당 정보를 현재 로그인한 사용자로 가지고 있는다. 현재 로그인한 사용자는 User\* 형태의 변수로 선언된다.

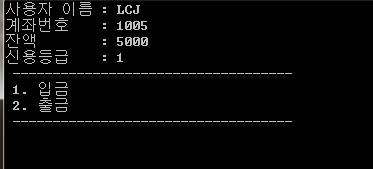


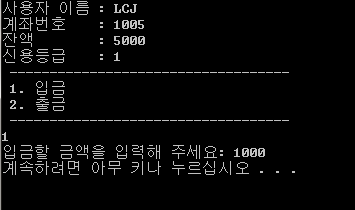
**[사용자 기능]**

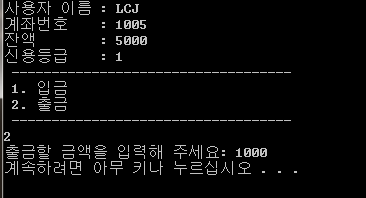
* 다음 아래의 기능들을 로그인한 사용자에게 제공한다.



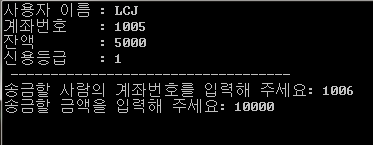
* 입, 출금 : 현재 로그인 된 사용자의 계좌에 입, 출금을 실행한다.



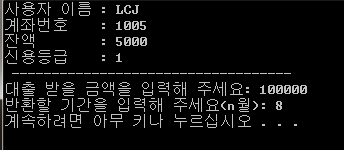


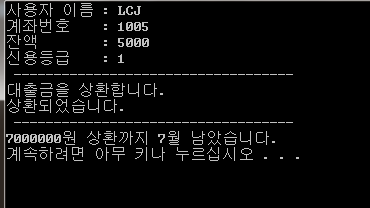
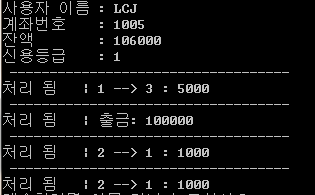


* 송금: 다른 사용자에게 현재 로그인 된 사용자가 돈을 송금한다. 자신이 가지고 있는   
   잔액을 넘어서 송금할 수 없다.



* 대출 : 은행으로부터 대출을 받는다.  
   대출 가능한 금액은 신용등급에 따라 상한선이 정해진다.  
   이미 대출한 내역이 있으면, 모두 상환하기 전까지 대출이 불가능하다.

신용등급이 1이면 1000만원까지 가능하며, 1씩 하락할 때마다 100만원씩 한도가 줄어들며, 신용등급이 10이면 대출이 불가하다.  


* 대출금 상환 : 대출한 금액을 자동으로 반환한다.  
   한번 반환할 때 마다 한 달이 지난 것으로 가정하고,   
   남은 상환금액을 남은 상환 기간.만금 나눈 몫을 상환한다.  
   상환 마지막 달은 남은 금액을 모두 반환한다.  
   상환한 내용은 거래내역에서 출금 형식으로 확인할 수 있다.  
    
  
* 거래내역 조회: 사용자의 모든 거래내역(transaction)을 조회한다.  
  

**[테스트]**

* 최종 테스트 시에 사용자는 10명 이상으로 한다.   
  거래 내역은 사용자당 20개 이상으로 한다.  
  적어도 1명 이상의 사용자는 대출 내역을 가지고 있어야 한다.

**[특수 상황 조건]**

* 실제 프로그램과 약간의 차이로 발생하는 특수한 상황의 결정은 본인이 직접 규칙을 정하고, 발표 시 설명한다.

**3. 예외 처리**

- 잔액이 부족하여 대출금 상환이 불가능한 상황에는 상환하지 않고 신용등급을 강등한다.

- 잔액이 부족하여 송금 및 출금이 불가한 경우

- 잘못된 금액을 넣었을 경우(음수, 문자 등)

- 메뉴 선택 시 메뉴의 리스트 이외의 값을 넣었을 경우

- 잘못된 아이디 혹은 비밀번호로 로그인 시도 시

**4. Program Requirements**

- App , User, transaction, loan 클래스로 구현

- 배열만 사용가능(다른 자료구조 사용 못함, STL 사용금지)

- 파일 입출력을 이용해서 사용자, transaction, loan DB 갱신.

- 소스코드에 함수, 중요 변수, 중요 기능, 중요 알고리즘 등은 반드시 주석을 단다.

**5. Interface Example**

- 자신이 편리한 인터페이스를 자유롭게 디자인한다.