ver 1.0

OOPT STAGE

-Demonstration

Project Team

Team T4

Date

2018 / 06 / 10

Team Members

1. 201611269 신문기
2. 201610401 손하영
3. 201510283 임진웅

1. 시스템 테스트

1. Step1. Specification Review

1.1 Stage 1000 Planning

-Activity 1001. Define draft plan

3. Functional Requirements
    - 4. Check Criminal Account
        - Functional Requirements에는 'Check Criminal Account'로 명시되어 있지만, 이후 문서에는 'Check Criminal History'로 명시되어 있다.

->

3. Functional Requirements

1. Deposit
2. Withdraw
3. Transfer
4. Check Transaction History
5. Check Criminal History

4. Non-functional Requirements
    - 1. Safe Transaction
        - Safe에 대한 기준이 제시되어 있지 않다.
    - 2. High Performance
        - Performance의 기준이 제시되어 있지 않다.
    - 3. OS-Independent
        - OS 독립적으로 실행하는 기준이 제시되어 있지 않다.

->

4. Non-functional Requirements
    1. Safe Transaction
       -범죄 이력이 있는 계좌를 조회할 수 있어 범죄에 피해를 입기 전에 예방할 수 있다.
    2. High Performance
       -반응속도가 1 초 이내로 작동한다.

-Activity 1003. Define Requirements

3. Operating Environments
    - Operating System- / Development Environments
        - Mac OS에 대한 버전이 명시되어있지 않다.
        - Java 버전이 명시되어있지 않다.

->

3. Operating Environments
    1. **OS** : OS-Independent - Windows 7 / 10, MacOS 10.13

4. Development Environments
    1. **OS** : OS-Independent - Windows 7 / 10, MacOS 10.13
    2. **Language** : Java 8
    3. **IDE** : IntelliJ IDEA Ultimate / Community 2018.1

5. Other Requirements
    - 1. Simple한 UI/UX구조를 가진다.
        - Simple 한 UI/UX에 대한 Design Definition 혹은 Guideline이 제시되어 있지 않다.

->

5. Other Requirements
    1. 5 가지 이내의 색 조합을 가지고 같은 화면 내에 5 개 이하의 버튼을 가져 Simple 한 UI/UX 구조를 가진다.

-Activity 1005. Implement Prototype

- Activity 1005. Implement Prototype
  - 그림 번호 및 상황에 대한 정보가 제시되어 있지 않다.

-> 2050-2060 단계에서 자세히 설명되어 있기 때문에 생략해도 된다고 판단했다.

-Activity 1006. Define Business Use Case

- Activity 1006. Define Business Use Case

  8. Describe Use-Cases
    - 3. Deposit
      - 1003에 언급되었던 타행입금에 관한 수수료가 누락되었다.
    - 7. CheckTransactionHistory
      - CheckCriminalHistory로 수정이 필요하다.

->

Use Case	3. Deposit
Actor	User
Description	1. 입금할 금액을 입력한다. 2. 계좌 번호에 해당되는 비밀번호를 입력한다. 입력한 비밀번호가 유효하지 않을 경우 알림을 받고 다시 입력한다. 3. 계좌에 돈을 입금하고 화면에 실행된 작업과 입금한 금액, 계좌의 잔액이 출력된 것을 확인한다. 이때 타행 입금시 수수료를 제외하고 입금한다. 4. 끝내기 버튼을 눌러 메인 화면으로 돌아간다.

Use Case	7. CheckCriminalHistory

1.3 Stage 2030 Analysis

-2031. Define Essential Use Case

- 전체적인 사항
    - Typical Course of Events의 시나리오에 번호를 입력하여 순서를 명확히 할 필요가 있다.
- 2. MediumCheck
    - Exceptional Course of Events
        - 5. “유효”의 정의가 모호하여 어떤 상황에서 유효한 것인지 알 수 없다. 카드번호와 계좌번호를 모두 입력받는데 계좌번호 오류에 대한 알림만 명시되어 있다.

->

Typical Courses of Events	(A): Actor , (S):System 1. (S) 화면에 매체 선택 버튼(카드 / 통장)을 띄운다. 2. (A) 둘 중에 이용할 매체에 해당하는 버튼을 입력한다. 3. (S) 번호 입력창을 띄운다. 4. (A) 선택에 따라 카드 번호나 계좌 번호를 입력한다. 5. (S) 입력된 계좌 번호나 입력된 카드번호에 연결된 계좌 번호가 DB에 존재하는 계좌인지 확인한다. .
Alternative Courses of Events	N/A
Exceptional Courses of Events	5. DB에 계좌가 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력받는다.(1번부터 다시 진행)

- 3. Deposit
    - Typical Courses of Events
        - 수수료를 정확히 어느정도 차감하는지 비율이 나와있지 않다.
        - 1006에는 수수료관련 언급이 없었는데 해당 단계에서 당행/타행을 비교하여 수수료를 계산하는 과정이 추가되었다.
        - 1006에는 끝내기 버튼을 눌러 메인으로 돌아간다고 명시되어 있으나 누락되었다.
        - Error에 대한 처리가 명시되지 않아 단순히 표시한다는 것인지 프로그램을 종료하는 것인지 알 수 없다.

->

Typical Courses of Events	(A): Actor , (S):System 1. (S) 화면에 금액 입력 창을 띄운다. 2. (A) 입금할 금액을 입력한다. 3. (S) 비밀번호 입력 창을 띄운다. 4. (A) 비밀번호를 입력한다. 5. (S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. . 6. (S) 비밀번호가 일치할 시 ATM의 소속은행과 계좌의 소속은행을 비교하여 수수료를 계산한다. 7. (S) 수수료를 제외한 금액만큼 계좌의 보유현금을 증가시킨다. 8. (S) 계좌에 돈을 입금한 후 실행한 작업(입금)과 입금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다. 9. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	6. 타행 계좌의 경우 1%의 수수료를 부과하고, 당행 계좌일 경우 수수료를 부과하지 않는다.
Exceptional Courses of Events	2. 입금할 금액이 음수일 경우 양수를 입력해달라고 알림을 보낸다. 5. 비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸 후 다시 3번 부터 진행한다.

- 4. Withdraw
    - Typical Courses of Events
        - 1006에는 수수료관련 사항이 없었는데 해당 단계에서 당행/타행을 비교하여 수수료를 계산하는 과정이 추가되었다.
        - 1006에는 끝내기 버튼을 눌러 메인으로 돌아간다고 명시되어 있으나 누락되었다.
        - Error에 대한 처리가 명시되지 않아 단순히 표시한다는 것인지 프로그램을 종료하는 것인지 알 수 없다.

->

Typical Courses of Events	(A): Actor , (S):System 1.(S) 화면에 금액 입력창을 띄운다. 2.(A) 출금할 금액을 입력한다. 3.(S) ATM의 소속은행과 계좌의 소속은행을 비교하여 수수료를 계산한다. 4.(S) 입력받은 금액과 수수료를 더한 만큼의 금액이 계좌의 잔액보다 크지 않은지 확인한다. 5.(S) 비밀번호 입력 창을 띄운다. 6.(A) 비밀번호를 입력한다. 7.(S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. 8.(S) 출금 금액과 수수료만큼 계좌의 보유현금을 감소시킨다. 9.(S) 계좌에 돈을 출금한 후 실행한 작업(출금)과 출금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다. 10. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	3. 타행 계좌의 경우 1%의 수수료를 부과하고, 당행 계좌일 경우 수수료를 부과하지 않는다.
Exceptional Courses of Events	2. 출금할 금액이 음수이면 양수를 입력해달라고 알림을 보낸다. 4. 출금하려는 금액과 수수료보다 잔액이 적을 시 잔액이 부족하다고 알림을 보낸다.

- 5. Transfer
  - Typical Courses of Events

    - 1006에는 수수료관련 사항이 없었는데 해당 단계에서
      당행/타행을 비교하여 수수료를 계산하는 과정이 추가되었다.
    - Error에 대한 처리가 명시되지 않아 단순히 표시한다는 것인지
      프로그램을 종료하는 것인지 알 수 없다.

->

	11. (S) 송금 금액과 수수료를 더한 만큼 송금할 계좌의 보유현금을 감소시킨다. 12. (S) 송금 금액만큼 송금받은 계좌의 보유현금을 증가시킨다. 13. (S) 계좌에 돈을 송금한 후 실행한 작업(송금)과 송금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다. 14. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	6. 타행 계좌의 경우 1%의 수수료를 부과하고, 당행 계좌일 경우 수수료를 부과하지 않는다.
Exceptional Courses of Events	2. 송금할 금액이 음수일 경우 양수를 입력해달라고 알림을 보낸다. 5. DB에 계좌가 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력을 받는다.(3번부터 다시 진행한다.) 7. 송금하려는 금액과 수수료보다 잔액이 적을 시 잔액이 부족하다고 알림을 보낸다. 10. 비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸 후 다시 8번 부터 진행한다.

- 6. CheckTransactionHistory
    - 1006에는 끝내기 버튼을 눌러 메인으로 돌아간다고 명시되어 있으나 누락되었다.

->

Typical Courses of Events	(A): Actor , (S):System 1. (S) 비밀번호 입력창을 띄운다. 2. (A) 비밀번호를 입력한다. 3. (S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. 4. (S) MediumCheck를 통해 입력받은 계좌의 거래내역 log파일을 불러와 화면에 출력한다. 5. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	3. 비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸 후 1번 부터 다시 진행한다.

- 7. CheckCriminalHistory
  - 1006에는 끝내기 버튼을 눌러 메인으로 돌아간다고 명시되어 있으나 누락되었다.

->

	3. (S) 입력된 계좌번호가 DB에 존재하는 계좌인지 확인한다. 4. (S) 입력된 계좌에 대한 범죄 이력 log를 가져와 출력한다. 5. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	3. DB에 계좌가 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력받는다(1번부터 다시 진행)

-2033. Define Domain Model

- 1004에서 제시하였던 User 객체가 보이지 않는다.

->1004에서 User에 대한 정의가 오해의 소지가 존재했으며, 1004의 User의 정의를 바로 잡았다.

-2034. Refine Glossary

- **Class Diagram**에선 **Bankbook**으로 표시되어 있으나 **Glossary**에는 **Passbook**으로 표시되었다. **Bankbook**과 **Passbook**을 혼용하여 혼란을 준다.

->

Account	Class	카드나 통장에 연결된 계좌
Bank	Class	ATM과 연결된 은행 데이터
Card	Class	카드
Bankbook	Class	통장
ATM	Class	ATM
User	Class	ATM을 사용하는 Actor
Account.accountNum	Attribute	계좌 번호
Account.bank	Attribute	계좌의 은행

-2036. Define Operation Contracts

- **inputMedium()**
  - **Exception**
    유효하지 않을때의 **Exception**이 없음.
  - **Post Condition**

- "입력된 계좌번호가 DB에 존재하는지 확인한다"라고 했지만 확인하는 단계에 해당하는 System Operation에 해당되는 사항이 없다.

->

Name	inputMedium()
Responsibilities	카드 번호를 입력할지 계좌 번호를 입력할지 선택할 수 있으며, 선택에 따라 카드번호나 계좌번호를 입력받는다.
Type	System
Cross Reference	System functions: R2 Use Case: "MediumCheck"
Notes	
Exceptions	DB에 계좌가 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력받는다.
Output	입력받은 카드 번호, 계좌 번호

Pre-conditions	입금, 출금, 송금, 거래내역확인 중 하나를 선택해야만 한다.
Post-conditions	입금일 경우 inputMoney(), 출금, 송금일 경우 inputAmount(), 거래내역확인일 경우 inputPassword()를 호출한다.

- inputMoney()
  - Exceptions
    - 2031의 Essential Use Cases에 음수를 입력할 시 에러 처리를 한다고 명시되어 있으나 해당 단계에선 제시되지 않고있다.

->

Name	inputMoney()
Responsibilities	입금할 돈을 입력받는다.
Type	System
Cross Reference	System functions: R3 Use Case: “Deposit”
Notes	
Exceptions	입력한 금액이 음수일 경우 양수를 입력해달라고 알림을 보낸다.
Output	입금할 돈의 양
Pre-conditions	계좌가 유효한지 확인되어 있는 상태여야 한다.
Post-conditions	비밀번호를 입력받는다.

- inputPassword()
    - Exception
      유효하지 않을때의 Exception이 없음.
    - Post Condition
        - “비밀번호가 계좌에 해당하는 비밀번호인지 확인하다”라고 명시되어 있으나 비밀번호를 확인하는 System Operation이 없다.

->

Name	inputPassword()
Responsibilities	비밀번호를 입력받는다.
Type	System
Cross Reference	System functions: R3, R4, R5, R6 Use Case: “Deposit”, “Withdraw”, “Transfer”, “CheckTransactionHistory”
Notes	
Exceptions	비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸 후 다시 입력받는다.
Output	입력받은 비밀번호
Pre-conditions	입금, 출금, 송금, 거래내역확인 중 하나를 선택한 상태이다.
Post-conditions	입금일 경우 통장에 돈이 입금되고, 출금일 경우 통장에서 돈이 출금되며,

	송금일 경우 통장에서 돈이 송금된다. 거래내역확인일 경우 입력된 계좌의 거래내역이 보여진다.

- inputAmount()
    - Exceptions
        - 2031의 **Essential Use Cases**에 음수를 입력할 시 에러 처리를 한다고 명시되어 있으나 해당 단계에선 제시되지 않고있다.

->

Name	inputAmount()
Responsibilities	출금하거나 송금할 돈의 양을 입력한다.
Type	System
Cross Reference	System functions: R4, R5 Use Case: “Withdraw”, “Transfer”
Notes	
Exceptions	입력한 금액이 음수일 경우 양수를 입력해달라고 알림을 보낸다.
Output	출금일 경우 출금할 돈, 송금일 경우 송금할 돈
Pre-conditions	계좌가 유효한 상태인지 확인되어 있어야한다.
Post-conditions	출금일 경우 출금이 되고 송금일 경우 송금할 계좌를 입력받는다.

- inputAccount()
    - Post Condition
        - “계좌번호가 유효한지 확인한다.”라고 명시되어 있으나 계좌가 유효한지 확인하는 System Operation이 없다.
        - 계좌의 유효성에 대한 기준이 없어 어떤 계좌가 유효한 것인지 알 수 없다.

->

Name	inputAccount()
Responsibilities	돈을 보낼 계좌나 범죄 이력을 조회할 계좌 번호를 입력한다.
Type	System
Cross Reference	System functions: R5, R7 Use Case: "Transfer", "CheckCriminalHistory"
Notes	
Exceptions	
Output	돈을 보낼 계좌 번호나 범죄 이력을 조회할 계좌 번호
Pre-conditions	계좌번호가 DB에 존재하는 계좌인지 확인한다.
Post-conditions	송금일 경우 비밀번호를 입력받고, 범죄이력조회일 경우 범죄 이력이 보여진다.

-2038. Refine System Test Case

- ATM_STC_003/004/005
  - _001: "잘"이라는 기준이 모호하다. 2031에 음수를 입력 받는 경우 Error처리한다고 명시되어 있으나 이를 확인 하는 것이 누락되어 있다.

->

ATM_STC_003 _001	Deposit	입금할 금액을 정확히 입력받을 수 있는지 확인한다.

1.4 Stage 2040 Design

-2041. Design Real Use Cases

- 2031과 동일한 이슈

->

Typical Courses of Events	(A): Actor , (S):System 1. (S) 화면에 매체 선택 버튼(카드 / 통장)을 띄운다. 2. (A) 둘 중에 이용할 매체에 해당하는 버튼을 입력한다. 3. (S) 번호 입력창을 띄운다. 4. (A) 선택에 따라 카드 번호나 계좌 번호를 입력한다. 5. (S) 입력된 계좌 번호나 입력된 카드번호에 연결된 계좌 번호가 DB에 존재하는 계좌인지 확인한다. .
Alternative Courses of Events	N/A
Exceptional Courses of Events	5. DB에 계좌가 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력받는다.(1번부터 다시 진행)

Typical Courses of Events	(A): Actor , (S):System 1. (S) 화면에 금액 입력 창을 띄운다. 2. (A) 입금할 금액을 입력한다. 3. (S) 비밀번호 입력 창을 띄운다. 4. (A) 비밀번호를 입력한다. 5. (S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. 6. (S) 비밀번호가 일치할 시 ATM의 소속은행과 계좌의 소속은행을 비교하여 수수료를 계산한다. 7. (S) 수수료를 제외한 금액만큼 계좌의 보유현금을 증가시킨다. 8. (S) 계좌에 돈을 입금한 후 실행한 작업(입금)과 입금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다. 9. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	6. 타행 계좌의 경우 1%의 수수료를 부과하고, 당행 계좌일 경우 수수료를 부과하지 않는다.
Exceptional Courses of Events	2. 입금할 금액이 음수일 경우 양수를 입력해달라고 알림을 보낸다. 5. 비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸후 다시 3번 부터 진행한다.

	4. (S) 입력받은 금액과 수수료를 더한 만큼의 금액이 계좌의 잔액보다 크지 않은지 확인한다. 5. (S) 비밀번호 입력 창을 띄운다. 6. (A) 비밀번호를 입력한다. 7. (S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. 8. (S) 출금 금액과 수수료만큼 계좌의 보유현금을 감소시킨다. 9. (S) 계좌에 돈을 출금한 후 실행한 작업(출금)과 출금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다. 10. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	3. 타행 계좌의 경우 1%의 수수료를 부과하고, 당행 계좌일 경우 수수료를 부과하지 않는다.
Exceptional Courses of Events	2. 출금할 금액이 음수이면 양수를 입력해달라고 알림을 보낸다. 4. 출금하려는 금액과 수수료보다 잔액이 적을 시 잔액이 부족하다고 알림을 보낸다. 7. 비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸후 다시 5번 부터 진행한다.

	시킨다. 12. (S) 송금 금액만큼 송금받은 계좌의 보유현금을 증가시킨다. 13. (S) 계좌에 돈을 송금한 후 실행한 작업(송금)과 송금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다. 14. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses	6. 타행 계좌의 경우 1%의 수수료를 부과하고, 당행 계좌일 경우 수수

of Events	료를 부과하지 않는다.
Exceptional Courses of Events	2. 송금할 금액이 음수일 경우 양수를 입력해달라고 알림을 보낸다. 5. DB에 계좌에 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력을 받는다.(3번부터 다시 진행한다.) 7. 송금하려는 금액과 수수료보다 잔액이 적을 시 잔액이 부족하다고 알림을 보낸다. 10. 비밀번호가 일치하지 않을 시 비밀번호가 일치하지 않다고 알림을 보낸후 다시 8번 부터 진행한다.

Typical Courses of Events	(A): Actor , (S):System 1. (S) 비밀번호 입력창을 띄운다. 2. (A) 비밀번호를 입력한다. 3. (S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. 4. (S) MediumCheck를 통해 입력받은 계좌의 거래내역 log파일을 불러와 화면에 출력한다. 5. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	N/A

Typical Courses of Events	(A): Actor , (S):System 1. (S) 계좌번호 입력창을 띄운다. 2. (A) 계좌번호를 입력한다. 3. (S) 입력된 계좌번호가 DB에 존재하는 계좌인지 확인한다. 4. (S) 입력된 계좌에 대한 범죄 이력 log를 가져와 출력한다. 5. (A) 끝내기 버튼을 눌러 메인 화면으로 돌아간다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	3. DB에 계좌가 존재하지 않을 경우 계좌가 유효하지 않다고 알림을 보낸 후 다시 입력받는다(1번부터 다시 진행)

-2042. Define Reports, UI and Storyboards

- 이미지 제목 및 설명 누락

->

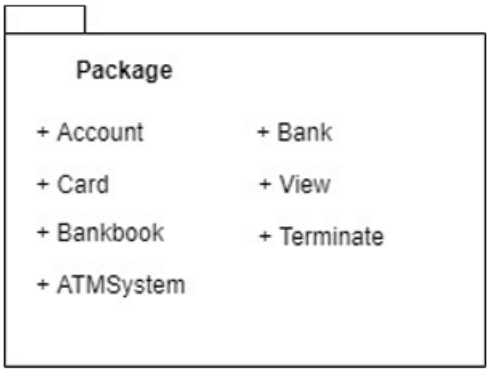
2. 계좌를 입력하는 화면

카드나 통장 중 입력할 매체를 고른 후 계좌 번호를 입력할 수 있다.

-2043. Refine System Architecture

- 실제 소스상에 없는 User, Passbook, ATM 존재
- 소스상에는 Bankbook, Terminate, View, AtmSystem 존재

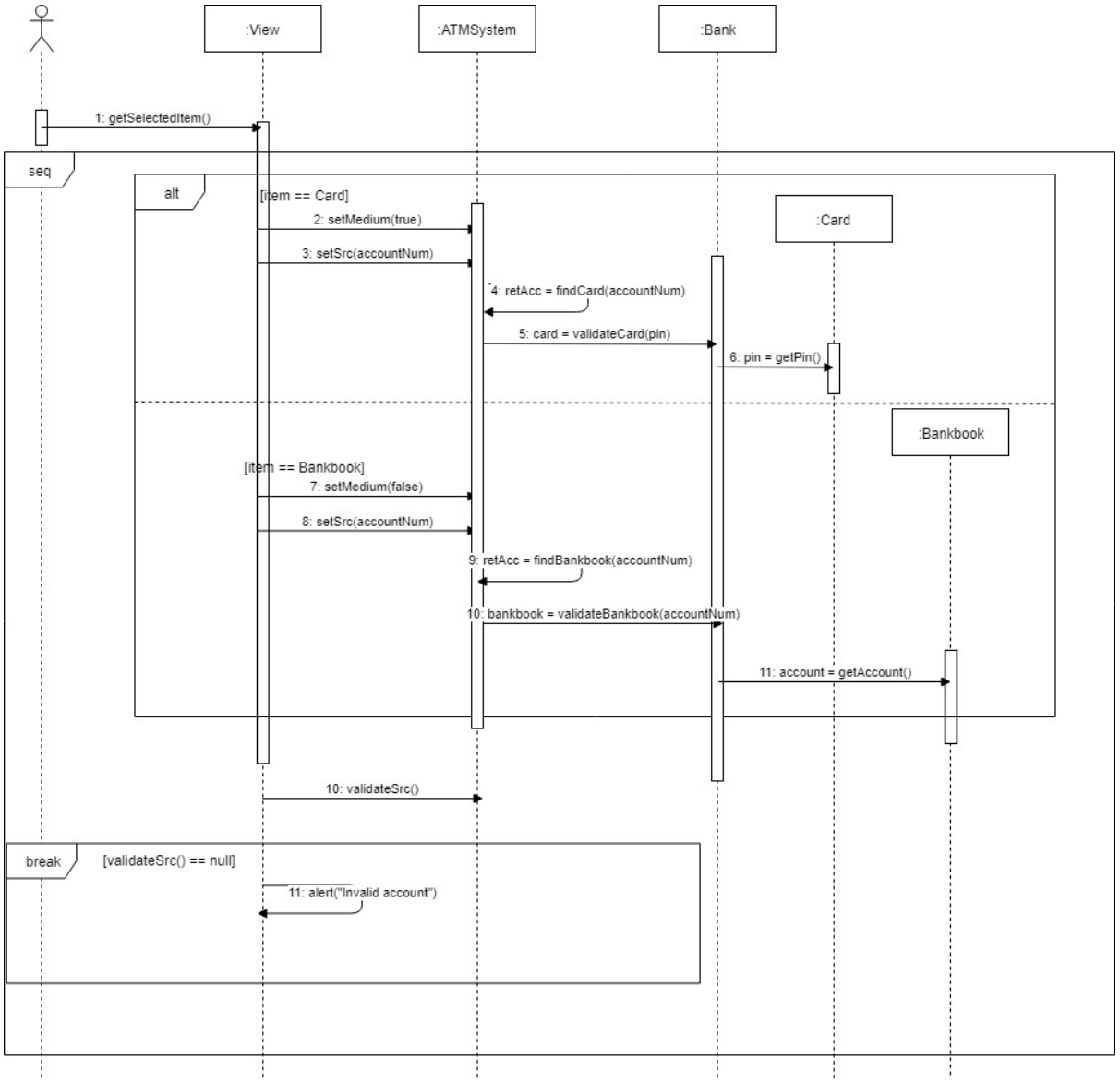
->



-2044. Define Interaction Diagrams

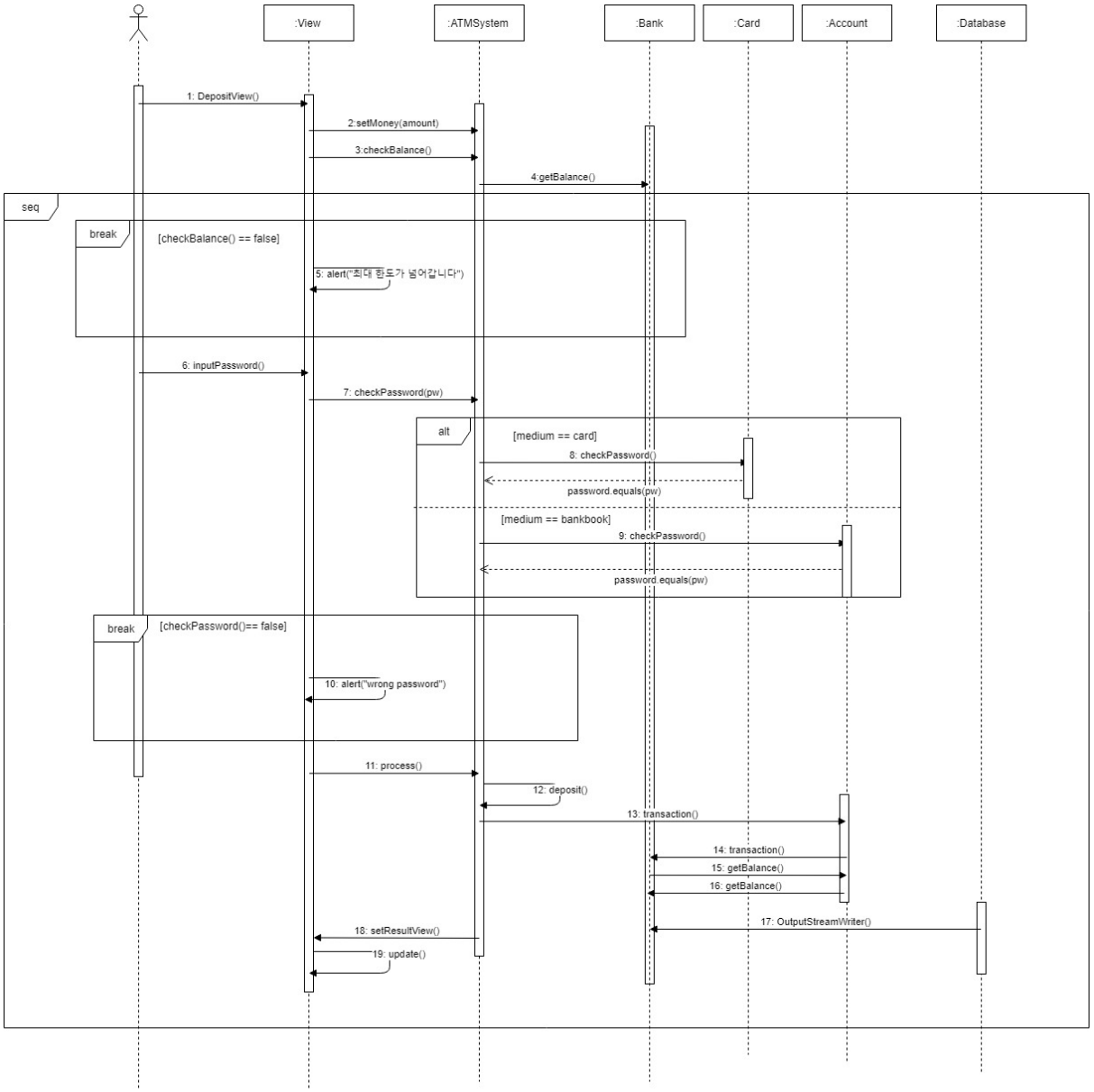
- Alert : 실제 소스상에서, ATMSystem 안에서 alert를 호출하지 않는다.
- 2. MediumCheck
  - 3. findCard:실제 소스상에서 atm.setSrc를 이용해 AtmSystem 내부에서 findCard를 자체적으로 호출한다.
  - 4. validateCard : return이 명시되어있지 않다.
  - 5. getPin : 실제 소스상에서 getPin은 card를 return 하지 않고 validateCard가 card를 return한다.
  - 7. findBankbook : 실제 소스상에서 atm.setSrc를 이용해 AtmSystem 내부에서 findBankbook을 자체적으로 호출한다.
  - 8. validateBankbook : return이 명시되어있지 않다.

->



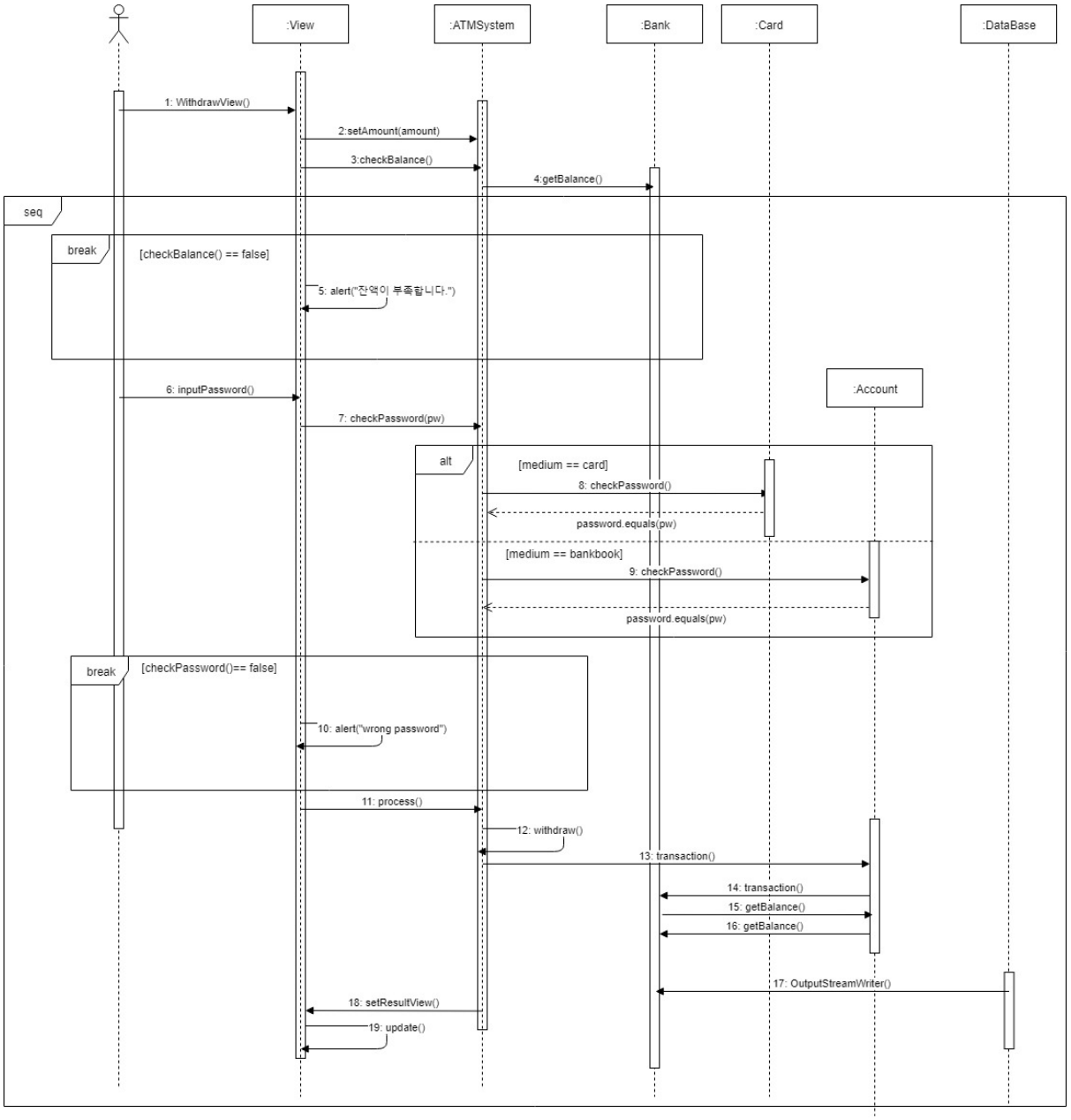
- 3. Deposit
  - 12. transaction : process -> transfer로 가는 부분이 명시되어 있지 않다.
  - 14. getBalance : 호출하고 return되는 과정이 생략되어 있다.
  - 15. fileReader : 실제 소스에서 FileWriter를 호출한다.

->



- 4. Withdraw
  - 12. transaction : process -> transfer로 가는 부분이 명시되어 있지 않다.
  - 14. getBalance : 호출하고 return되는 과정이 생략되어 있다.
  - 15. fileReader : 실제 소스에서 FileWriter를 호출한다.

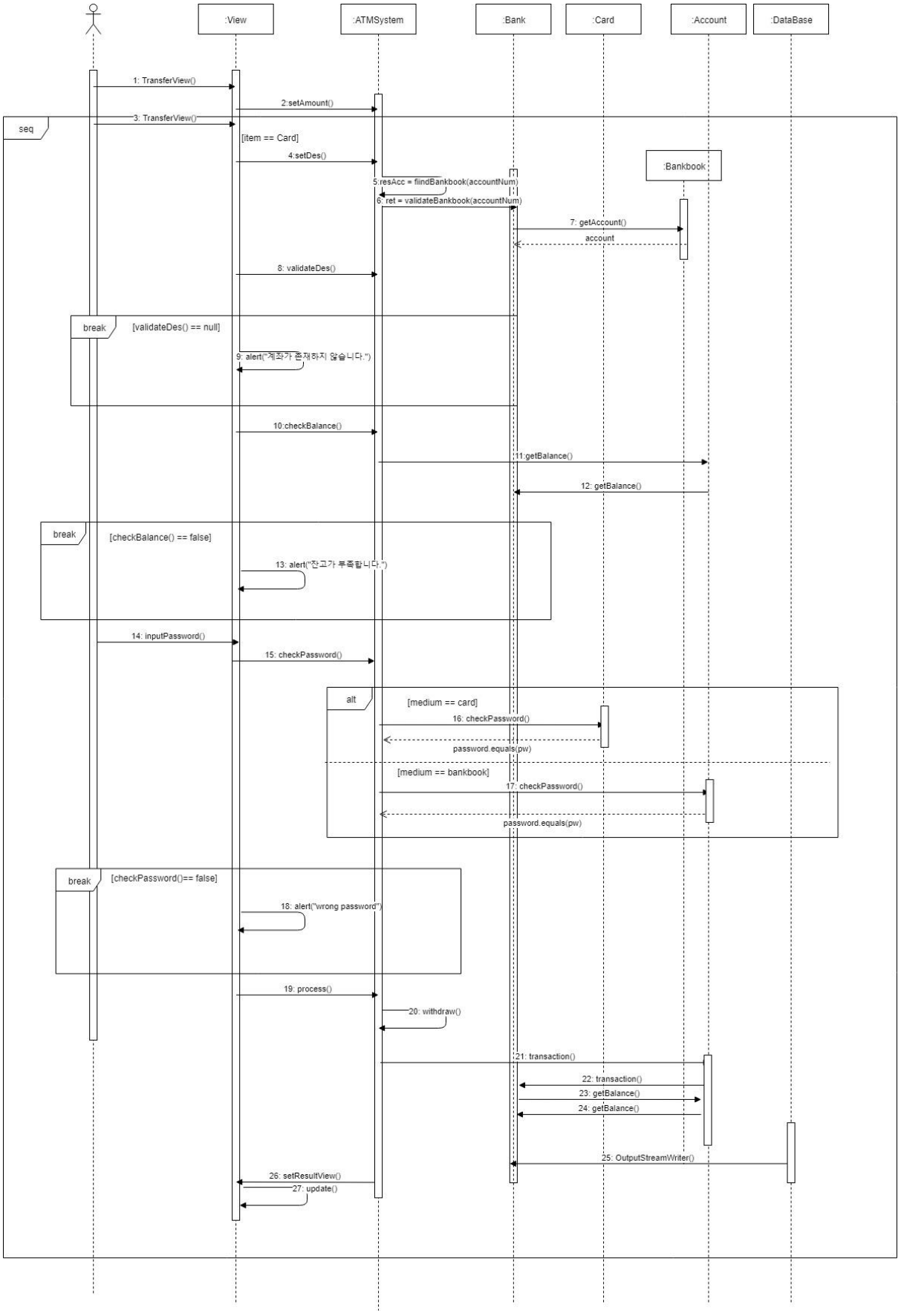
->

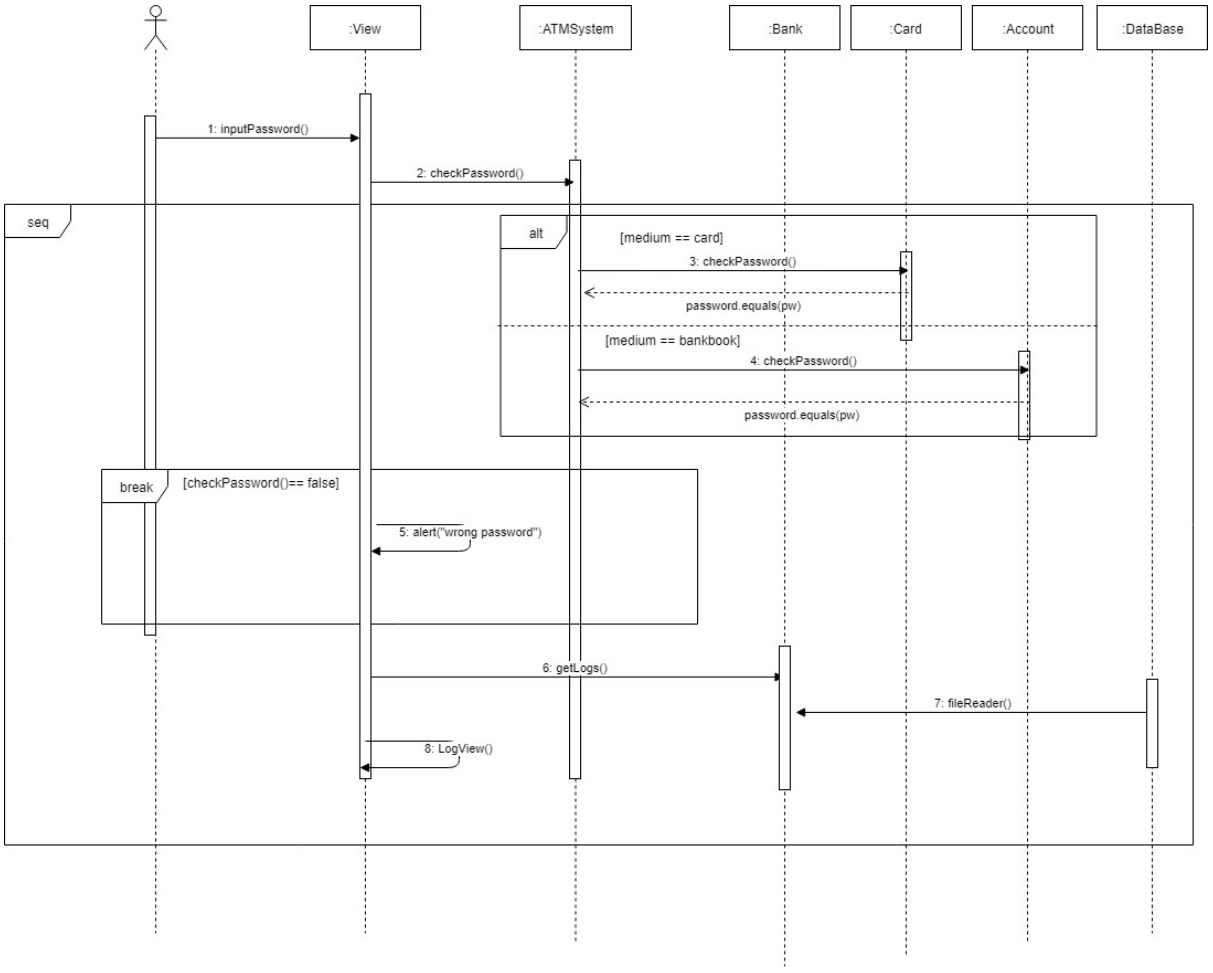


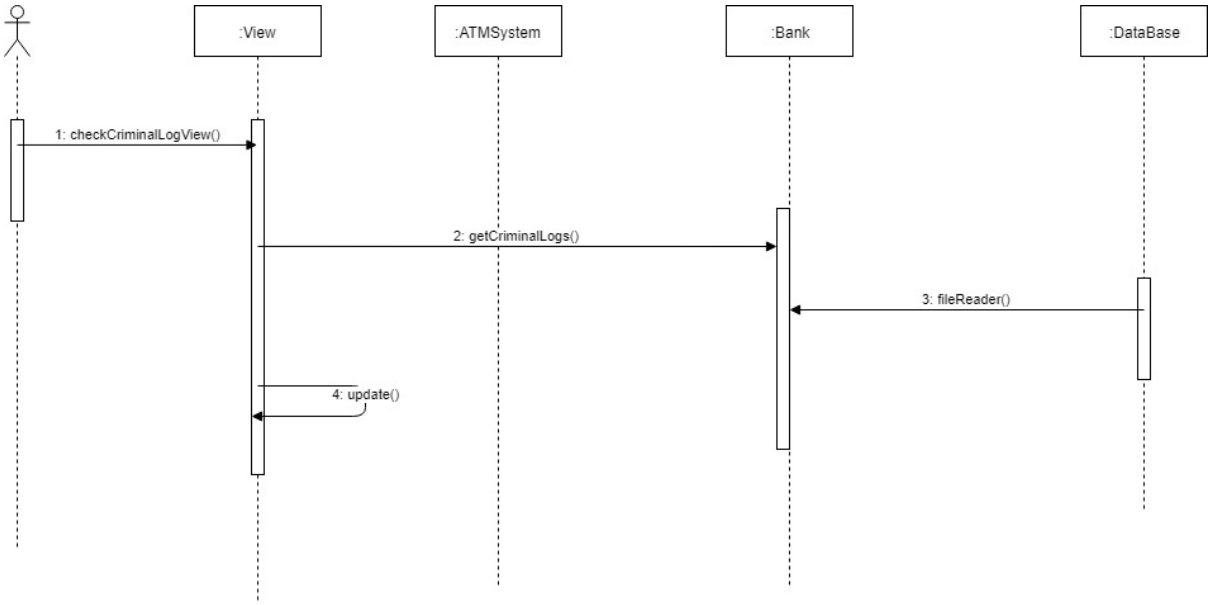
- 5. Transfer
  - 5. findCard : 실제 소스상에서 atm.setSrc를 이용해 AtmSystem 내부에서 findCard를 자체적으로 호출한다.
  - 6. validateCard : return이 명시되어있지 않다.

- 7. getPin : 실제 소스상에서 getPin은 card를 return 하지 않고 validateCard가 card를 return한다.
- 9. findBankbook : 실제 소스상에서 atm.setSrc를 이용해 AtmSystem 내부에서 findBankbook을 자체적으로 호출한다.
- 10. validateBankbook : return이 명시되어있지 않다.
- 23. transaction : process -> transfer로 가는 부분이 명시되어 있지 않다.
- 25. getBalance : 호출하고 return되는 과정이 생략되어 있다.
- 26. fileReader : 실제 소스에서 FileWriter를 호출한다.

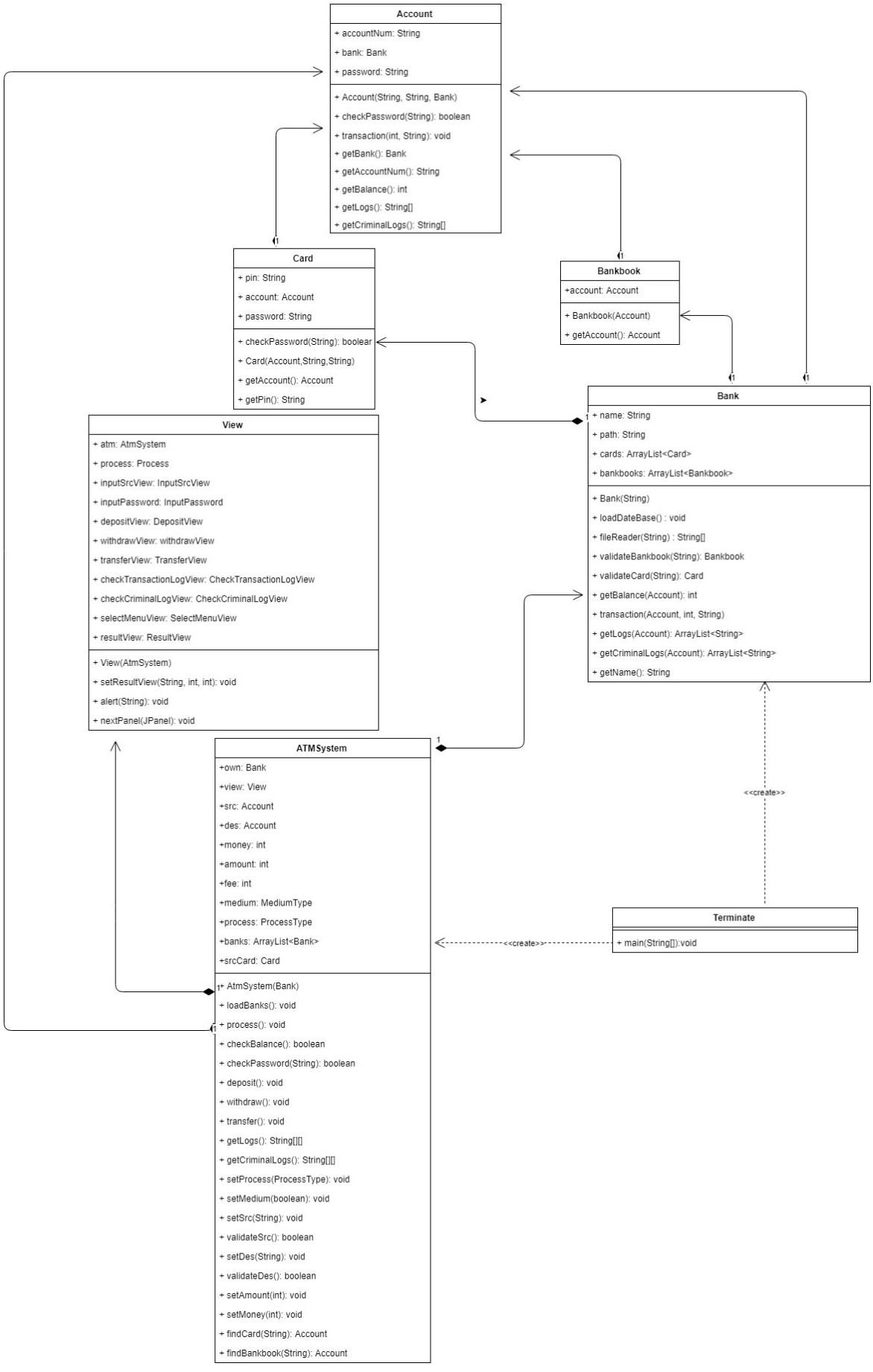
->


:View
:ATMSystem
:Bank
:Card
:Account
:DataBase
:Bankbook
1: TransferView()
2:setAmount()
3. TransferView()
seq
[item == Card]
4:setDes()
5:resAcc = fiindBankbook(accountNum)
6: ret = validateBankbook(accountNum)
7: getAccount()
account
8: validateDes()
break
[validateDes() == null]
9: alert("계좌가 존재하지 않습니다.")
10:checkBalance()
11:getBalance()
12: getBalance()
break
[checkBalance() == false]
13: alert("잔고가 부족합니다.")
14: inputPassword()
15: checkPassword()
alt
[medium == card]
16: checkPassword()
password.equals(pw)
[medium == bankbook]
17: checkPassword()
password.equals(pw)
break
[checkPassword()== false]
18: alert("wrong password")
19: process()
20: withdraw()
21: transaction()
22: transaction()
23: getBalance()
24: getBalance()
25: OutputStreamWriter()
26: setResultView()
27: update()



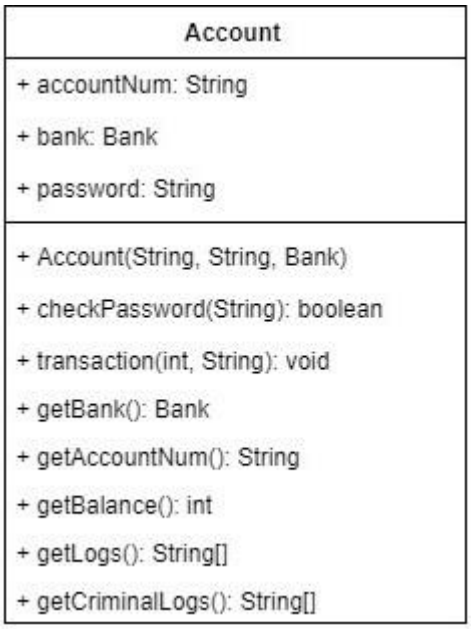


-2045. Define Design Class Diagrams


Account
+ accountNum: String
+ bank: Bank
+ password: String
+ Account(String, String, Bank)
+ checkPassword(String): boolean
+ transaction(int, String): void
+ getBank(): Bank
+ getAccountNum(): String
+ getBalance(): int
+ getLogs(): String[]
+ getCriminalLogs(): String[]
Card
+ pin: String
+ account: Account
+ password: String
+ checkPassword(String): boolear
+ Card(Account,String,String)
+ getAccount(): Account
+ getPin(): String
Bankbook
+account: Account
+ Bankbook(Account)
+ getAccount(): Account
Bank
+ name: String
+ path: String
+ cards: ArrayList<Card>
+ bankbooks: ArrayList<Bankbook>
+ Bank(String)
+ loadDateBase() : void
+ fileReader(String) : String[]
+ validateBankbook(String): Bankbook
+ validateCard(String): Card
+ getBalance(Account): int
+ transaction(Account, int, String)
+ getLogs(Account): ArrayList<String>
+ getCriminalLogs(Account): ArrayList<String>
+ getName(): String
View
+ atm: AtmSystem
+ process: Process
+ inputSrcView: InputSrcView
+ inputPassword: InputPassword
+ depositView: DepositView
+ withdrawView: withdrawView
+ transferView: TransferView
+ checkTransactionLogView: CheckTransactionLogView
+ checkCriminalLogView: CheckCriminalLogView
+ selectMenuView: SelectMenuView
+ resultView: ResultView
+ View(AtmSystem)
+ setResultView(String, int, int): void
+ alert(String): void
+ nextPanel(JPanel): void
ATMSystem
+own: Bank
+view: View
+src: Account
+des: Account
+money: int
+amount: int
+fee: int
+medium: MediumType
+process: ProcessType
+banks: ArrayList<Bank>
+srcCard: Card
+ AtmSystem(Bank)
+ loadBanks(): void
+ process(): void
+ checkBalance(): boolean
+ checkPassword(String): boolean
+ deposit(): void
+ withdraw(): void
+ transfer(): void
+ getLogs(): String[][]
+ getCriminalLogs(): String[][]
+ setProcess(ProcessType): void
+ setMedium(boolean): void
+ setSrc(String): void
+ validateSrc(): boolean
+ setDes(String): void
+ validateDes(): boolean
+ setAmount(int): void
+ setMoney(int): void
+ findCard(String): Account
+ findBankbook(String): Account
Terminate
+ main(String[]):void
<<create>>
<<create>>
1
1
1
1
1
1
1
1

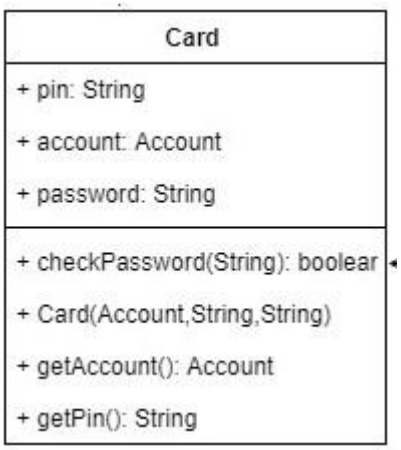
- Account
    - 소스 : **accountNum**, **bank**, password, **Account**, getBank, getAccountNum, getBalance, **checkPassword**,**transaction**, getLogs, getCriminalLogs
    - 문서 : **accountNum**, **bank**, balance, logs, criminalLogs, **Account**, **checkPassword**, **transaction**

->



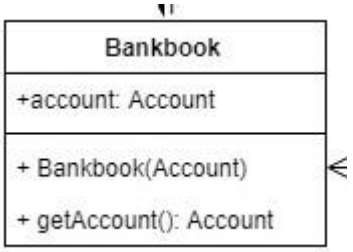
- Card
    - 소스 : **pin**, **account**, password, **Card**, getAccount, getPin, **checkPassword**
    - 문서 : **pin**, **account**, **Card**, **checkPassword**

->



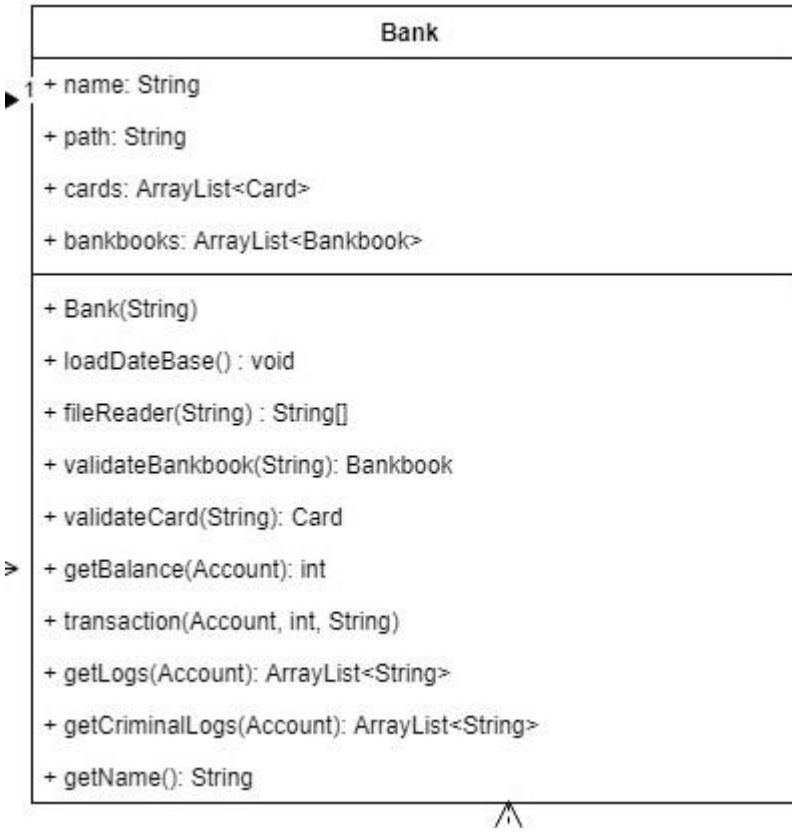
- Bankbook
  - 소스 : **account, Bankbook**, getAccount
  - 문서 : **account, Bankbook**

->



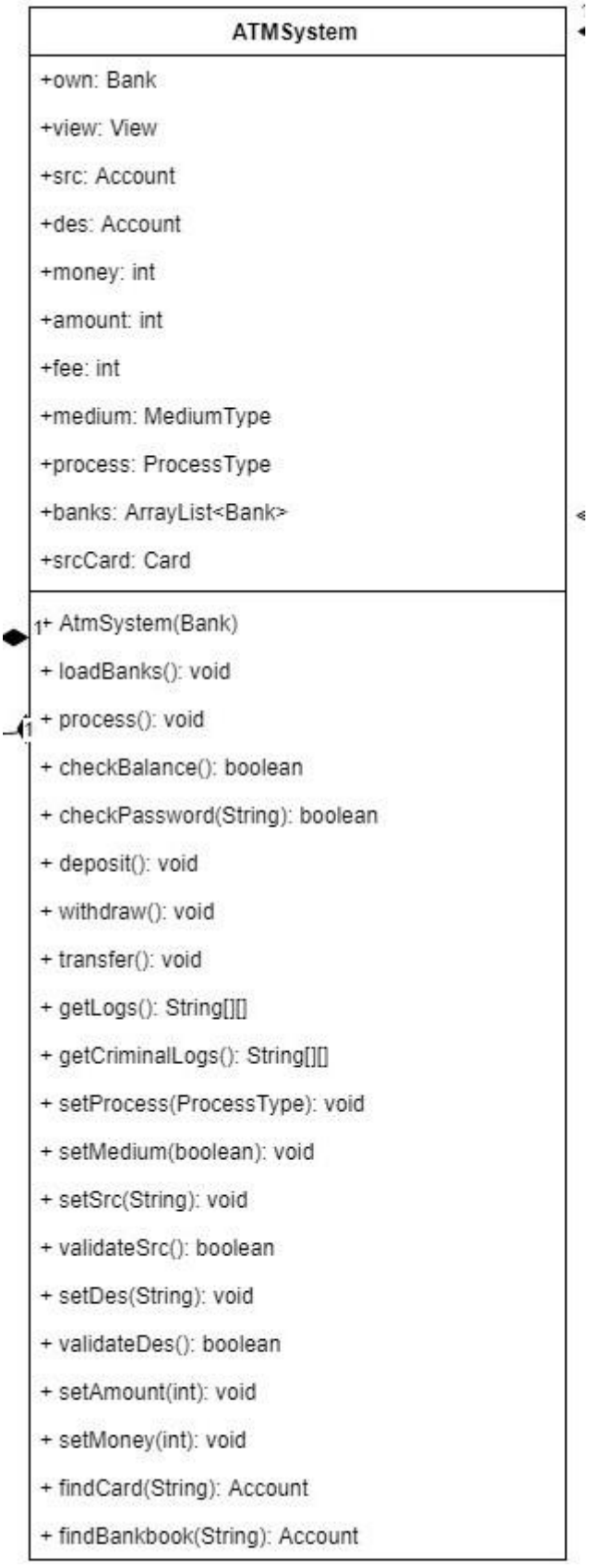
- Bank
  - 소스 : **name**, **Bank**, **loadDataBase**, **fileReader**, **validateBankbook**, **validateCard**, **getBalance**, **transaction**, **getLogs**, **getCriminalLogs**
  - 문서 : **name**, path, cards, bankbooks, **Bank**, **loadDataBase**, **fileReader**, **validateBankbook**, **validateCard**, **getBalance**, **transaction**, **getLogs**, **getCriminalLogs**, getName

->



- ATMSystem
  - 소스 : own, view, **src**, **des**, **amount**, **money**, fee, **medium**, banks, **process**, srcCard, **AtmSystem**, **loadBanks**, **process**, **checkBalance**, **checkPassword**, **deposit**, **withdraw**, **transfer**, getLogs, getCriminalLogs, setProcess, setMedium, setSrc, **validateSrc**, setDes, **validateDes**, setAmount, setMoney, **findCard**, **findBankbook**
  - 문서 : **src**, **money**, **medium**, **des**, **process**, criminalLogs, logs, **amount**, **AtmSystem**, **loadBanks**, **process**, **checkBalance**, **checkPassword**, **deposit**, **withdraw**, **transfer**, **validateSrc**, **validateDes**, **findCard**, **findBankbook**

->


ATMSystem
+own: Bank
+view: View
+src: Account
+des: Account
+money: int
+amount: int
+fee: int
+medium: MediumType
+process: ProcessType
+banks: ArrayList<Bank>
+srcCard: Card
1
+ AtmSystem(Bank)
+ loadBanks(): void
1
+ process(): void
+ checkBalance(): boolean
+ checkPassword(String): boolean
+ deposit(): void
+ withdraw(): void
+ transfer(): void
+ getLogs(): String[][]
+ getCriminalLogs(): String[][]
+ setProcess(ProcessType): void
+ setMedium(boolean): void
+ setSrc(String): void
+ validateSrc(): boolean
+ setDes(String): void
+ validateDes(): boolean
+ setAmount(int): void
+ setMoney(int): void
+ findCard(String): Account
+ findBankbook(String): Account

1.5 Stage 2050 Implementation

-2051. Implement Class & Methods Definition

- class Account
  - Account()
    - purpose
      - ''Account 의 객체 생성한다.'를 수정해야 한다. Account는 생성자로서 단순히 attribute를 초기화해주는 역할만 수행하며, 별도의 객체를 생성하지 않기 때문이다.
      - 이 부분은 이전 보고서에서 찾아낸 부분이지만, 수정이 안된 것으로 보인다.
  - getBalance()
    - abstract operation
      - 'bank DB 에서 balance 를 불러와 반환한다.'를 수정해야한다. bankDB라는 표현이 정의되어있지 않아 정확히 무엇을 의미하는지 모호하다.
      - 이 부분은 이전 보고서에서 찾아낸 부분이지만, 수정이 안된 것으로 보인다.
  - checkPassword()
    - abstract operation
      - 'bankDB 에서 비밀번호 일치 여부를 반환한다.'를 수정해야 한다. bankDB라는 표현이 정의되어있지 않아 정확히 무엇을 의미하는지 알 수 없다.
      - 이 부분은 이전 보고서에서 찾아낸 부분이지만, 수정이 안된 것으로 보인다.
  - getCriminalLogs()
    - abstract operation
      - 계좌의 범죄 이력과 횟수를 반환한다고 했으나, 실제 프로그램을 실행시켜 보면 횟수가 아닌 범죄 내용 및 범죄 시간에 대해서만 반환한다.

->

Type	Method
Name	Account()
Purpose	attribute 를 초기화한다.
CrossReference	Use Case :R2, R3, R4, R5, R6, R7
Input(Method)	String accountNum, Bank bank
Output(Method)	-
Abstract Operation(Method)	attribute 를 초기화한다.
Exceptional Courses of Events	-

Type	Method
Name	getBalance()
Purpose	계좌 잔액을 확인한다.
CrossReference	Use Case R2, R3, R4, R5, R6
Input(Method)	-

Output(Method)	int bank.getBalance(account: this)
Abstract Operation(Method)	bank 에서 balance 를 가져와 반환한다.
Exceptional Courses of Events	-

Type	Method
Name	checkPassword()
Purpose	비밀번호가 유효한지 확인한다.
CrossReference	Use Case R3, R4, R5
Input(Method)	String pw
Output(Method)	boolean bank.checkPassword(account: this, pw)
Abstract Operation(Method)	Account 의 비밀번호와의 일치여부를 반환한다.
Exceptional Courses of Events	-

Type	Method
Name	getCriminalLogs()
Purpose	범죄이력을 조회한다.
CrossReference	Use Case R7
Input(Method)	-
Output(Method)	ArrayList<String> bank.getCriminalLogs(this)
Abstract Operation(Method)	계좌의 범죄 내용과 범죄 시간을 반환한다,

- class AtmSystem
  - Overview
    - attribute
      - View view가 누락되어있다.
  - AtmSystem()
    - purpose
      - 'AtmSystem 의 객체 생성한다.'를 수정해야 한다.
        AtmSystem은 생성자로서 단순히 attribute를
        초기화해주는 역할만 수행하며, class View의 객체를
        생성하기 때문이다.
      - 이 부분은 이전 보고서에서 찾아낸 부분이지만, 수정이
        안된 것으로 보인다.
  - loadBanks()
    - Input

```
private void loadBanks() {
    File file = new File(System.getProperty("user.dir") + "/res");
    File[] banks = file.listFiles();
    if (banks != null) {
        for (File bank : banks) {
            this.banks.add(new Bank(bank.getName()));
        }
    }
}
```

      - 문서에는 Bank own을 input으로 받도록 나와있으나,
        실제 코드에서는 어떠한 파라미터도 받지 않는다.
  - checkBalance()
    - Output

```
67      public boolean checkBalance() {
68          return (amount + fee < src.getBalance()) && (money - fee + src.getBalance() > 0);
69      }
```

- boolean (amount + fee) < arc.getBalance()로 명시되어있으나, 이는 코드와 다르다.

- Abstract operation
  - 코드와 실제 문서에 나온 내용이 다르므로 이에 대한 통일이 필요하다.

- deposit()
  - Abstract operation

```
79      public void deposit() {
80          src.transaction(money - fee,"입금");
81      }
```

- '계좌의 balance를 amount-fee 만큼 증가시킨다'는 표현이 나와있으나, 실제 코드에서는 money라는 이름의 변수로 표현되어있다. 이에 대한 통일이 필요하다.

- setDes()
  - Abstract operation

```
134     public void setDes(String accountNum) {
135         if (medium == MediumType.CARD) {
136             src = findCard(accountNum);
137         } else {
138             src = findBankbook(accountNum);
139         }
140     }
```

- '매체에 따라 findCard 또는 findBankBook의 결과를 des에 저장한다'고 명시되어있으나, 실제 코드에서는 src를 설정한다.

- setAmount()
  - Abstract operation
    - amount 값을 설정할 때, 타행 계좌인 경우 fee(수수료)를 설정한다는 설명이 누락되어있다.

- setMoney()
  - Abstract operation
    - money 값을 설정할 때, 타행 계좌인 경우 fee(수수료)를 설정한다는 설명이 누락되어있다.

- findCard()
  - 문서 전체적으로 'bank DB'라는 표현이 반복되어 나온다. 실제 DB를 사용하지도 않고, 텍스트 파일 형태의 단순 파일 시스템을 사용하기 때문에 표현을 바꾸거나, 정의를 새로 해줘야할 필요가 있다.

->

Type	class
Name	AtmSystem
Purpose	ATM 정보를 모아두는 클래스
Overview(class)	enum MediumType{CARD, BANKBOOK}, enum ProcessType{DEPOSIT, WITHDRAW, TRANSFER} Attribute : Bank own, View view, Account src, Account des, int amount,int money, int fee, ArrayList<Bank> banks, MediumType medium, ProcessType process, Card srcCard Method : AtmSystem(),loadBanks(), process(), checkBalance(), checkPassword(), deposit(), withdraw(), transfer(), getLogs(), getCriminalLogs(), setProcess(), setMedium(), setSrc(), validateSrc(), setDes(), validateDes(), setAmount(), setMoney(), findCard(), findBankbook()
CrossReference	Use Case R1, R2, R3, R4, R5, R6, R7
Exceptional Courses of Events	N/A

Type	Method
Name	AtmSystem()
Purpose	AtmSystem 의 Attribute 를 초기화한다.

CrossReference	Use Case R1, R2, R3, R4, R5, R6, R7
Input(Method)	Bank own
Output(Method)	-
Abstract Operation(Method)	AtmSystem 의 Attribute 를 초기화한다.
Exceptional Courses of Events	-

Type	Method
Name	loadBanks()
Purpose	은행 목록에 이용가능한 은행을 추가한다.
CrossReference	Use Case R1, R2, R3, R4, R5, R6, R7
Input(Method)	
Output(Method)	-
Abstract Operation(Method)	banks 배열에 존재하는 Bank 를 추가한다.
Exceptional Courses of Events	-

Type	Method
Name	checkBalance()
Purpose	계좌 잔고가 출금 또는 송금할 금액보다 많은지 확인한다.
CrossReference	Use Case R4, R5
Input(Method)	-
Output(Method)	if(money == 0) boolean (amount + fee) <=src.getBalance() else boolean money-fee+src.getBalance() >= 0
Abstract Operation(Method)	money= 0 일 경우 amount + fee 와 src.getbalance 를 비교한 결과를 반환한다. 아닐 경우 money-fee+src.getBalance()를 비교한 결과를 반환한다.
Exceptional Courses of Events	-

Type	Method
Name	deposit()
Purpose	src 계좌에 입금한다.
CrossReference	Use Case R3
Input(Method)	-
Output(Method)	-
Abstract Operation(Method)	계좌의 balance 를 money - fee 만큼 증가시키고 log 에 msg 와 해당 거래내용을 업데이트한다.
Exceptional Courses of Events	-

Abstract Operation(Method)	this.amount 에 amount 값을 저장한다. 타행 계좌일 경우 fee = amount*0.01 로 세팅한다.
Exceptional Courses of Events	.-

Type	Method
Name	setMoney()
Purpose	입금 금액을 세팅한다.
CrossReference	Use Case R3
Input(Method)	int money
Output(Method)	-
Abstract Operation(Method)	this.money 에 money 값을 저장한다. 타행 계좌일 경우 fee = money*0.01 로 세팅한다.
Exceptional Courses of Events	-

Abstract Operation(Method)	banks 에서 pin 을 이용해 카드와 연결된 계좌의 유효성 유무를 검색하고 존재하면 ret.getAccount 을 반환한다.
Exceptional Courses of Events	어느 banks 에도 존재하지 않는 pin 의 경우 null 을 반환한다.

- class Bank
  - Bank()
    - purpose
      - 'Bank 의 객체를 생성한다.'를 수정해야 한다. Bank는 생성자로서 단순히 attribute를 초기화해주는 역할만 수행하며, 별도의 객체를 생성하지 않기 때문이다.
      - 이 부분은 이전 보고서에서 찾아낸 부분이지만, 수정이 안된 것으로 보인다.
  - loadDataBase()
    - 문서 전체적으로 DB라는 표현이 반복되어 나온다. 실제 DB를 사용하지도 않고, 텍스트 파일 형태의 단순 파일 시스템을 사용하기 때문에 표현을 바꾸거나, 정의를 새로 해줘야할 필요가 있다.
  - validateBankbook()
    - bank DB라는 표현이 등장하는데, 실제 DB를 사용하지도 않고, 이에 대한 정의가 모호하여 특정 변수를 말하는 것인지 분명한 표현으로 바꿀 필요가 있다.
  - validateCard()
    - bank DB라는 표현이 등장하는데, 실제 DB를 사용하지도 않고, 이에 대한 정의가 모호하여 특정 변수를 말하는 것인지 분명한 표현으로 바꿀 필요가 있다.

```
82      public Card validateCard(String pin) {
83          for (Card card : this.cards) {
84              if (card.getPin().equals(pin)) {
85                  return card;
86              }
87          }
88          return null;
89      }
```

    - 'accountNum'에 해당하는 카드를 확인한다고 하였으나, 실제 코드에서는 pin을 확인한다.

->

Type	Method
Name	Bank()
Purpose	Bank 의 Attribute 를 설정한다.
CrossReference	Use Case R2, R3, R4, R5, R6, R7
Input(Method)	String name
Output(Method)	-
Abstract Operation(Method)	Bank 의 Attribute 를 설정한다.
Exceptional Courses of Events	-

Abstract Operation(Method)	Bank 에 관련된 File 내에 저장되어 있는 account 와 card, bankbook 의 데이터를 불러온다.
Exceptional Courses of Events	-

Abstract Operation(Method)	banks 의 통장중 accountNum 에 해당하는 통장이 존재하는지 확인하고 존재하면 해당 bankbook 을 반환한다.
Exceptional Courses of Events	banks 의 통장중 accountNum 에 해당하는 통장이 존재하는지 확인하고 존재하지 않으면 null 을 반환한다.

Type	Method
Name	validateCard()
Purpose	Card 가 유효한지 확인한다.
CrossReference	Use Case R2
Input(Method)	String pin
Output(Method)	Card card
Abstract Operation(Method)	cards 에서 pin 이 일치하는 카드가 존재하는지 확인하고 존재하면 해당 card 를 반환한다.
Exceptional Courses of Events	cards 에서 pin 이 일치하는 카드가 존재하는지 확인하고 존재하지 않으면 null 을 반환한다.

- class Bankbook

  - Bankbook()
    - Purpose, Abstract operation
      - 'Bankbook의 객체를 생성한다'는 표현이 명시되어 있으나, 이는 생성자로서 단순히 attribute를 초기화해주는 역할만을 수행한다. 또한, 별도의 객체를 생성하지 않기 때문에 수정해야한다.

->

Type	Method
Name	Bankbook()
Purpose	Bankbook 의 Attribute 를 설정한다.
CrossReference	Use Case R2
Input(Method)	Account account
Output(Method)	-
Abstract Operation(Method)	Bankbook 의 Attribute 를 설정하고, this.account 를 input 된 account 로 설정한다.
Exceptional Courses of Events	-

- class Card
  - Card()
    - Purpose
      - 'Card의 객체를 생성한다'고 명시되어 있으나, 이는 생성자로서 단순이 attribute를 초기화해주는 역할만을 수행한다. 또한, 별도의 객체를 생성하지 않기 때문에 수정해야한다.
    - Abstract operation

```
6       public Card(Account account,String pin,String password) {
7           this.pin = pin;
8           this.password = password;
9           this.account = account;
10      }
```

      - 'Bankbook의 객체를 생성한다'고 명시되었으나, 실제 코드에서는 객체를 새로이 생성하지도 않고, 인자값으로 받아서 초기화 해주지도 않는다.

->

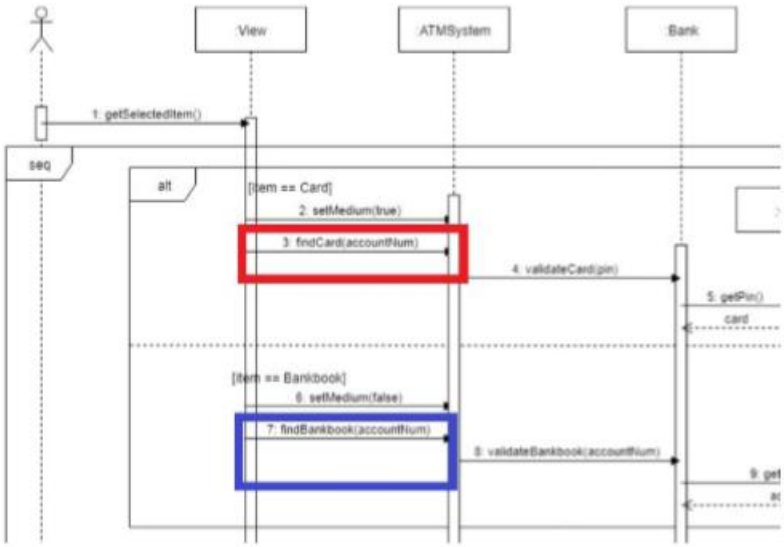
Type	Method
Name	Card()
Purpose	Card 의 Attribute 를 초기화한다.

CrossReference	Use Case R2
Input(Method)	Account account String pin String password
Output(Method)	-
Abstract Operation(Method)	Card 의 Attribute 를 초기화하고, this.account 를 input 된 account 로, this.password 를 input 된 password 로, this.pin 을 input 된 pin 로 설정한다.
Exceptional Courses of Events	-

-2052. Implements Windows

- SelectMenuView
  - 거래 내역 조회를 단순히 '조회'라고 명시하였으나, 이는 추상적인 측면이 있기 때문에 '거래 내역 조회'로 변경하는 것이 바람직하다.
  - Post-Condition
    - 계좌를 입력할 수 있다고는 하나, 신용카드등 매체를 선택하는 과정도 포함되어있기에, '선택한 버튼의 기능으로 진행한다'로 수정해야 한다.
- InputSrcView
  - Diagram

```
460                 Object item = selectMedium.getSelectedItem();
461                 if (item == null) {
462                     item = "Card";
463                 }
464                 if (item.equals("Card")) {
465                     atm.setMedium(true);
466                 } else {
467                     atm.setMedium(false);
468                 }
```



- 코드와 다이어그램 내용의 차이가 존재한다.

- Post-Condition
  - '선택한 버튼의 기능으로 계속 진행한다'라고 명시되어 있으나, 정확히 어떤 기능으로 진행한다는지 모호하므로 '이전 단계에서 선택한 기능의 다음 단계로 계속 진행한다'로 수정해야 한다.

->

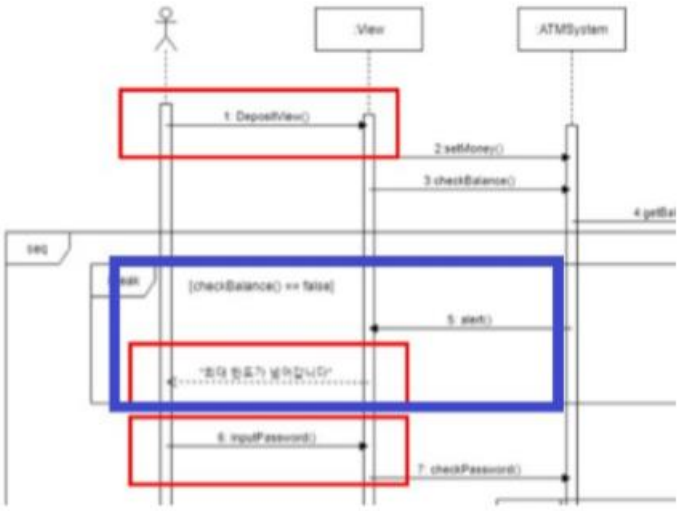
Name	SelectMenuView
Type	GUI
Responsibilities	입금, 출금, 송금, 거래 내역 조회, 범죄이력 버튼 중 하나를 누른다.
Cross Reference	Functional Requirement: R 1
Notes	입금, 출금, 송금, 조회, 범죄이력 조회 버튼 중 하나를 누른다.
Post-Condition	선택한 버튼의 기능으로 진행한다.
Pre-Condition	ATM이 켜진 상황이어야 한다.

Name	InputSrcView()
Type	GUI
Responsibilities	카드 번호를 입력할지 통장 번호를 입력할지 선택하고 그에 따라 카드 번호 또는 통장 번호를 입력할 수 있다.
Cross Reference	Functional Requirement: R 2
Notes	카드와 통장 중 하나를 선택한 뒤 번호를 입력한다.
Post-Condition	이전 단계에서 선택한 기능의 다음 단계로 계속 진행한다.
Pre-Condition	메뉴 선택이 되어 있다.

- DepositView()
  - Pre-Condition
    - '계좌 번호가 유효한 상태여야 한다'를 '해당 번호가 유효한 상태여야 한다'로 바꿔야 한다.

- alert "최대한도가 넘어갑니다"
  - Diagram

```
191     private class DepositView extends JPanel {
192         private DepositView() {
193             setLayout(null);
194             setBackground(bgColor);
195             setForeground(fontColor);
196 
197             InputLabel amountLabel = new InputLabel("금액",80);
198             add(amountLabel);
199 
200             NumberInput amountInput = new NumberInput(80);
201             add(amountInput);
202 
203             NextBtn nextBtn = new NextBtn();
204             nextBtn.addActionListener((ActionEvent e) -> {
205                 int amount = amountInput.getAmount();
206                 if (amount != 0) {
207                     atm.setMoney(amount);
208                     if (atm.checkBalance()) {
209                         nextPanel(process.next());
210                     } else {
211                         alert("최대한도가 넘어갑니다.");
212                     }
```

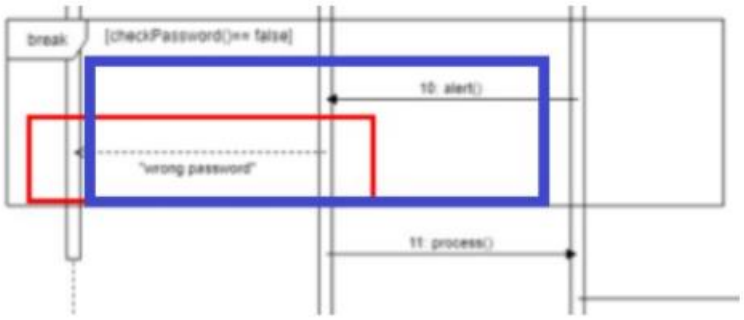


(파란색으로 표시한 부분)

- 코드와 다이어그램 내용의 차이가 존재한다.

- inputPassword
  - Responsibilities
    - '계좌의 비밀번호'를 '이전 단계에서 선택한 매체(통장 혹은 카드)에 대한 비밀번호'로 수정해야 한다.
- alert "wrong password"
  - Diagram

```
503             nextBtn.addActionListener((ActionEvent e) -> {
504                 String pw = String.valueOf(pwInput.getPassword());
505                 if (atm.checkPassword(pw)) {
506                     atm.process();
507                     nextPanel(process.next());
508                     pwInput.setText(null);
509                 } else {
510                     alert("wrong password");
511                 }
512             });
```



(파란색으로 표시한 부분)

- 코드와 다이어그램 내용의 차이가 존재한다.

- update

Name	Update
Type	GUI
Responsibilities	입금이 완료된 후 결과

- 'update'가 맞는 표현이나, 문서에는 'Update'로 표기되어 있다.

->다이어그램 수정,

Name	DepositView()
Type	GUI
Responsibilities	입금할 금액을 입력할 수 있다.
Cross Reference	Functional Requirement: R 3
Notes	
Post-Condition	비밀번호를 입력할 수 있다.
Pre-Condition	카드 번호나 통장 번호를 입력하고 해당 번호가 유효한 상태여야 한다.

Name	inputPassword
Type	GUI
Responsibilities	이전 단계에서 선택한 매체(통장 혹은 카드)에 대한 비밀번호를 입력할 수 있다.

Cross Reference	Functional Requirement: R 3
Notes	비밀번호를 입력받은 뒤 유효하지 않을 경우 에러 메시지를 출력하는 화면으로 넘어간다.
Post-Condition	비밀번호가 유효할 경우 입금이 계속 진행되고 결과창이 뜬다.
Pre-Condition	입금할 돈을 입력받은 상황이어야 한다.

Name	update
Type	GUI
Responsibilities	입금이 완료된 후 결과와 내역을 보여준다.
Cross Reference	R 3
Notes	작업, 금액, 잔액의 입금 내역을 보여준다.
Post-Condition	돌아가기 버튼 또는 끝내기 버튼을 누를 수 있다.
Pre-Condition	유효한 비밀번호를 입력하고 입금이 계속 진행된다.

- WithdrawView()
  - Pre-Condition
    - '계좌 번호가 유효한 상태여야 한다'를 '해당 번호가 유효한 상태여야 한다'로 바꿔야 한다.
- alert "잔액이 부족합니다"
  - Diagram

```
234             nextBtn.addActionListener((ActionEvent e) -> {
235                 int amount = amountInput.getAmount();
236                 if (amount != 0) {
237                     atm.setAmount(amount);
238                     if (atm.checkBalance()) {
239                         nextPanel(process.next());
240                     } else {
241                         alert("잔액이 부족합니다.");
242                         process.init();
243                         nextPanel(selectMenuView);
244                     }
245                 }
246             });
```

(파란색으로 표시한 부분)

- 코드와 다이어그램 내용의 차이가 존재한다.
  - Note

Name	alert "잔액이 부족합니다"
Type	GUI
Responsibilities	계좌에서 출금할 돈이 잔고보다 더 클 때 알린다.
Cross Reference	Functional Requirement: R 4
Notes	계좌에서 출금할 돈이 잔고보다 더 클 때 알린다.

- Responsibilities 내용과 중복되므로, 지워야 한다.

- alert "wrong password"
  - Diagram

- 코드와 다이어그램 내용의 차이가 존재한다. 이는 위의 InputPassword 클래스에서 발생했던 것과 같다.

  - update

Name	Update
Type	GUI
Responsibilities	출금이 완료된 후 결고
Cross Reference	R 4

- 'update'가 맞는 표현이나, 문서에는 'Update'로 표기되어 있다.

- TransferView()
  - Pre-Condition
    - '계좌 번호가 유효한 상태여야 한다'를 '해당 번호가 유효한 상태여야 한다'로 바꿔야 한다. 카드 번호를 입력하는 경우가 있기 때문.
- alert "계좌가 존재하지 않습니다"
  - Diagram

```
270             NextBtn nextBtn = new NextBtn();
271             nextBtn.addActionListener((ActionEvent e) -> {
272                 int amount = amountInput.getAmount();
273                 if (amount != 0) {
274                     atm.setAmount(amount);
275                     atm.setDes(desInput.getText());
276                     desInput.setText("");
277                     if (!atm.validateDes()) {
278                         alert("계좌가 존재하지 않습니다.");
```

    - 코드와 다이어그램 내용의 차이가 존재한다.
- alert "잔고가 부족합니다"
  - Diagram

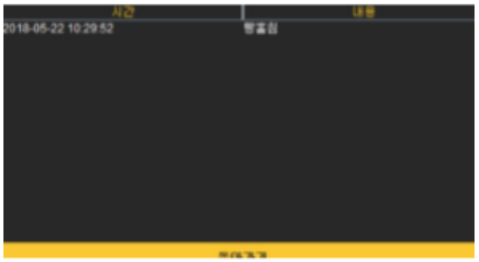
```
277                     if (!atm.validateDes()) {
278                         alert("계좌가 존재하지 않습니다.");
279                     } else if (!atm.checkBalance()) {
280                         alert("잔고가 부족합니다.");
281                         process.init();
282                         nextPanel(selectMenuView);
283                     } else {
284                         nextPanel(process.next());
285                     }
```

- 코드와 다이어그램 내용의 차이가 존재한다.

- alert “wrong password”
  - Diagram
    - 코드와 다이어그램 내용의 차이가 존재한다. 이는 위의 InputPassword 클래스에서 발생했던 것과 같다.
- update

Name	Update
Type	GUI
Responsibilities	송금이 완료된
Cross Reference	R 5

  - ‘update’가 맞는 표현이나, 문서에는 ‘Update’로 표기되어 있다.
- update
  - Post-Condition



  - ‘돌아가기 버튼 또는 끝내기 버튼을 누를 수 있다’라고 명시되어 있으나, 실제 프로그램 동작에서는 돌아가기 버튼만 구현 되어있다.

->다이어그램 수정,

Name	WithdrawView()
Type	GUI
Responsibilities	출금할 금액을 입력할 수 있다.
Cross Reference	Functional Requirement: R 4
Notes	
Post-Condition	비밀번호를 입력할 수 있다.
Pre-Condition	카드 번호나 통장 번호를 입력하고 해당 번호가 유효한 상태여야 한다.

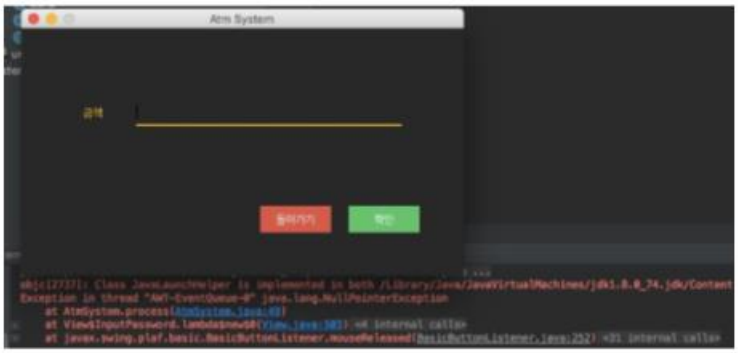
Name	TransferView()
Type	GUI
Responsibilities	송금할 금액과 보낼 계좌 번호를 입력할 수 있다.
Cross Reference	Functional Requirement: R 5
Notes	
Post-Condition	비밀번호를 입력할 수 있다.
Pre-Condition	카드 번호나 통장 번호를 입력하고 해당 번호가 유효한 상태여야 한다.

Name	update
Type	GUI
Responsibilities	범죄 이력 내역을 보여준다.
Cross Reference	R 7
Notes	계좌의 범죄 이력 내역을 보여준다.
Post-Condition	돌아가기 버튼을 누를 수 있다.
Pre-Condition	유효한 계좌를 입력하고 범죄 이력 조회가 계속 진행된다.

1.6 Stage 2060 Unit test & System test

- ATM_STC_003~005
  - '계좌번호에 해당되는 비밀번호를 입력되었을 때 수수료를 제외하고 출금되는지 확인 한다'를 '계좌번호에 해당되는 비밀번호를 입력되었을 때 수수료를 더하고 출금되는지 확인 한다'로 고쳐야 한다.
- ATM_STC_005

  - 송금 대상 계좌를 입력하면, 정상적인 계좌임에도 불구하고 계좌가 존재하지 않는다는 경고가 발생한다. 결과적으로 송금 기능이 동작하지 않는다. (송금 대상 계좌를 입력할 때, 입력창에 콤마(,)가 찍혀서 출력되는데, 이것이 문제의 원인인 것 같다)
  - ATM_STC_005_002~005까지 모두 Pass에서 Fail로 표기해야한다.
- ATM_STC_006



  - 초기 프로그램 실행 이후 거래기록 조회를 실행해보면, AtmSystem.process에서 에러가 발생한다. 따라서 거래기록을 조회할 수 없다.
  - 입금 이후 거래기록을 조회하면 동작하는 것을 확인할 수 있지만, 직전의 입금 기록에 대해서는 조회할 수 없다. 프로그램을 종료하고 새로운 거래를 실행하면, 프로그램 종료 이전의 거래 기록을 확인할 수 있다. 즉, 최신 거래 기록을 바로 확인할 수 없다.
  - 프로그램을 실행하는 동안에 발생된 거래 기록에 대해서는 확인할 수 없고, 프로그램 재실행 이후에 확인이 가능한다.
  - 여러 오류가 발생하고 있기 때문에 Pass가 아닌 Fail로 표기해야 한다.
- CheckTransactionHistory, CheckCriminalHistory
  - 이력을 조회하고 싶은 계좌를 입력하라고 명시되어있으나, 실제 카드 pin번호를 통한 이력 조회도 가능하기 때문에 System test case의 추가가 필요하다.
- CheckCriminalHistory와 Safe Transaction의 정확한 차이가 무엇인지 모호하다. Safe Transaction 항목은 제거해야할 것으로 보인다.
- OS-Independent
  - 1003에서 Windows 7, Windows 10, Mac OS가 구동 가능한 OS로 명시되어있으나, 2063에서는 'OS에 무관하게'라고 표현되어 있으므로 이에 대한 수정이 필요하다. (실제 시스템케이스를 통과하였으나, 정확히 어떤 OS에 대해서 테스트를 진행했는지 명시해야 한다)
- Simple한 UI/UX에 대한 System test case가 누락되어있다.

->

ATM_STC_003 _002	Deposit	계좌 번호에 해당되는 비밀번호를 입력했을 때 수수료를 더하고 입금되는지 확인한다.	R 3	P

ATM_STC_006_001	CheckTransaction History	입력한 계좌의 거래내역이 화면에 정상적으로 출력되는지 확인한다.	R 6	P
ATM_STC_006_002	CheckTransactionHistory	입력한 카드번호의 거래내역이 화면에 정상적으로 출력되는지 확인한다.	R 6	p
ATM_STC_008	Performance	모든 버튼 입력 및 이벤트에 대하여 1초 이내로 반응하는지 확인한다.	R 9	F
ATM_STC_009_001	OS - Independent	프로그램이 window10에서 정상적으로 동작하는지 확인한다.	R 10	P
ATM_STC_009_002	OS-Independent	프로그램이 Mac OS에서 정상적으로 동작하는지 확인한다.	R 10	p

4. Brute Force Testing Report

Number	Test Case	Result
13	입금메뉴 선택 후, log.txt파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다	F
14	출금메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
15	송금메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
16	조회메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
17	범죄이력 조회 메뉴 선택 후, criminalLog.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
18	조회 버튼을 눌러 1,622,925개 내역이 있는 거래 계좌를 입력하고 확인 버튼을 눌렀을 경우 페이징 처리가 되어 있지 않고 단순 스크롤로 약 160만여개의 거래 내역을 보여주기 때문에 Simple한 UI/UX를 가진다고 보기 어렵다.	F
19	범죄 이력 조회 버튼을 눌러 1,563,312개 내역이 있는 거래 계좌를 입력하고 확인 버튼을 눌렀을 경우 페이징 처리가 되어 있지 않고 단순 스크롤로 약 150만여개의 거래 내역을 보여주기 때문에 Simple한 UI/UX를 가진다고 보기 어렵다.	F
20	여러번 기능을 반복 할 경우 JPanel의 UI Component가 Clear 되지 않고 화면에 겹쳐서 보이는 현상이 나타난다.	F

->

13~17: 시간을 단축해보려 분석해보았으나, 구조 상 문제로 인해 1초 이내로 시간을 줄이기 어렵

다고 판단하였다.

18~19: 처음 기획했을 때 Simple한 UI/UX의 기준을 5가지 이하의 구별하기 쉬운 보색으로 구성된 색배치, 화면에 5개 이하의 버튼, 내역 조회에서는 시간과 내용을 구분하기 쉽도록 만드는 것으로 정했었고, 현재도 Simple한 UI/UX라고 판단하고 있기 때문에 현재 상태 그대로 유지하기로 하였다.

20: 코드를 수정하였다.

2. 정적 분석

1. Sonarqube

->

Test Summary

19	0	0	0.060s
tests	failures	ignored	duration

100% successful

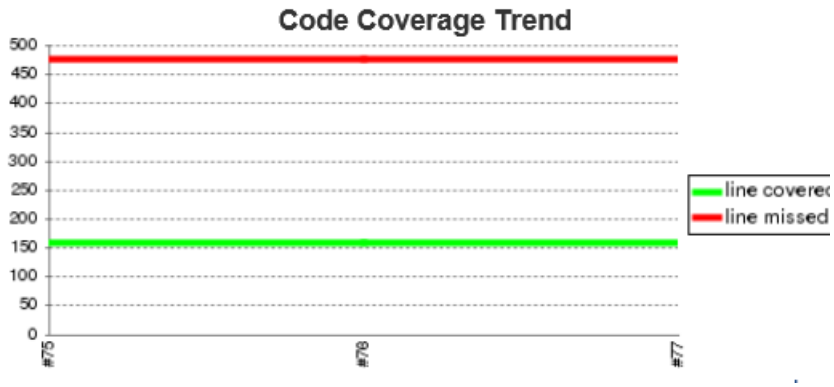
Packages Classes

Packages

Package	Tests	Failures	Ignored	Duration	Success rate
default-package	19	0	0	0.060s	100%

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
AccountTest	7	0	0	0.004s	100%
BankTest	7	0	0	0.045s	100%
BankbookTest	1	0	0	0.005s	100%
CardTest	4	0	0	0.006s	100%



->테스트는 성공적이지만 코드 커버리지가 낮으므로 에러가 발생할 수 있는 여지 존재.

2. PMD

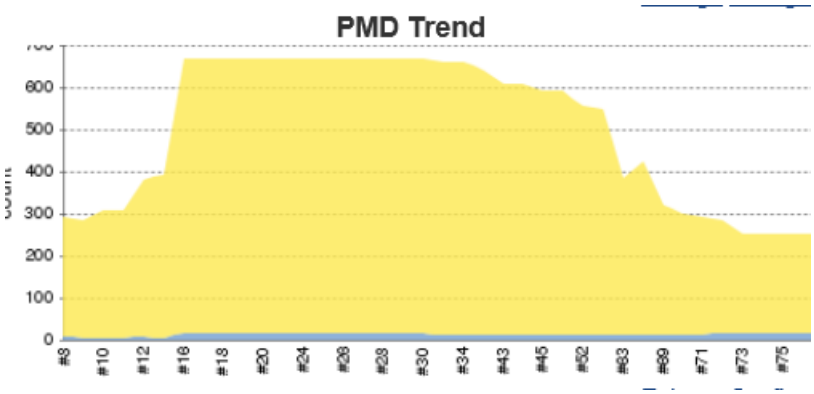
Details

Folders | Files | People | Categories | **Types** | Warnings | Origin | Details | New | Fixed | Normal | Low

Type	Total	Distribution
AccessorClassGeneration	34	
AccessorMethodGeneration	92	
AssignmentInOperand	2	
AtLeastOneConstructor	5	
AvoidCatchingGenericException	3	
AvoidDuplicateLiterals	2	
AvoidInstantiatingObjectsInLoops	5	
CallSuperInConstructor	10	
CommentDefaultAccessModifier	16	
CommentRequired	184	
ConfusingTernary	2	
DataflowAnomalyAnalysis	14	
DefaultPackage	16	
ImmutableField	29	
LawOfDemeter	46	
LocalVariableCouldBeFinal	107	
LooseCoupling	18	
MethodArgumentCouldBeFinal	60	
NullAssignment	4	
OnlyOneReturn	9	
PositionLiteralsFirstInComparisons	2	
SimpleDateFormatNeedsLocale	1	
TooManyMethods	1	
UseCollectionIsEmpty	1	
UseUtilityClass	1	
UselessParentheses	2	
Total	666	

총 666개의 문제 발생.

->



300개 이하로 줄였다. 완전히 줄이는 것은 설계 상 문제로 힘들다고 판단하였다.

3. Findbugs

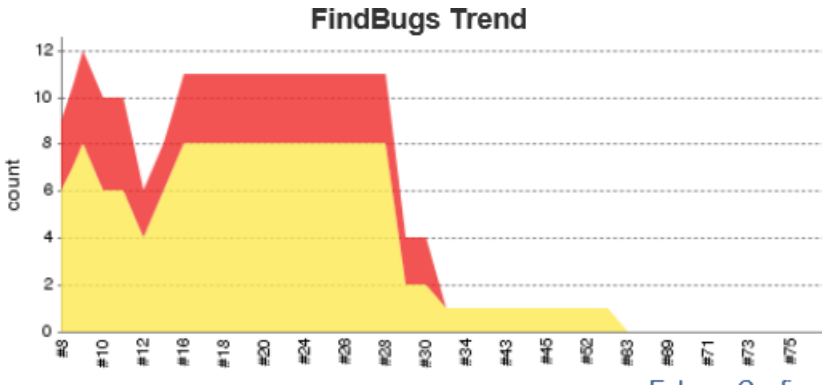
Details

Files | Categories | **Types** | Warnings | Origin | Details | New | High | Normal

Type	Total	Distribution
DM_DEFAULT_ENCODING	2	
OBL_UNSATISFIED_OBLIGATION_EXCEPTION_EDGE	1	
OS_OPEN_STREAM	1	
SIC_INNER_SHOULD_BE_STATIC	2	
URF_UNREAD_FIELD	2	
Total	8	

총 11개의 문제 발생.

->



완전히 제거하였다.

4. CheckStyle

Summary

Total	High Priority	Normal Priority	Low Priority
77	0	77	0

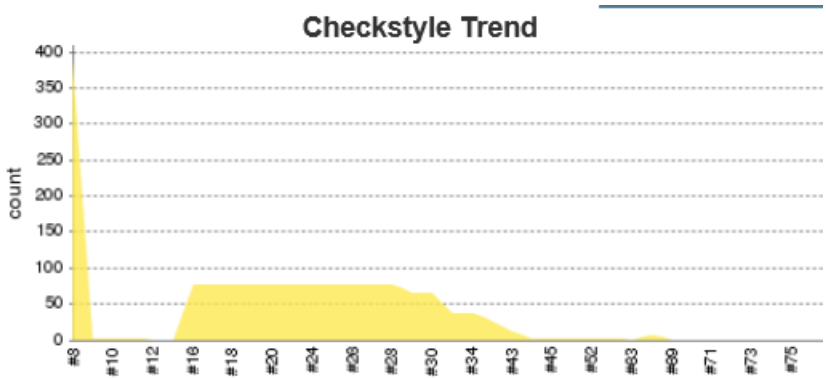
Details

Folders | Files | People | **Categories** | Types | Warnings | Origin | Details | New

Category	Total	Distribution
Imports	15	
Indentation	2	
Naming	4	
Whitespace	56	
Total	77	

총 77개의 문제 발생.

->


Checkstyle Trend
count
400
350
300
250
200
150
100
50
0
#8
#10
#12
#16
#18
#20
#24
#26
#28
#30
#34
#43
#45
#52
#63
#69
#71
#73
#75

완전히 제거하였다.