SOFTWARE VERIFICATION

2nd SYSTEM TEST

Team 4

201411259 고수창

201411314 전소영

201412005 이세라

201511304 하지윤

INDEX

1. 2nd Specification Review

Stage 1000

3. Functional Requirements

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

1001.Define Draft Plan

CheckCriminal History	1. 범죄이력을 조회하고자 하는 계좌를 입력하고 입력된 계좌번호가 DB에 존재하는 계좌인지 확인한다. 유효하지 않을 경우 계좌가 유효하

1003. Define Requirements

R 7	CheckCriminalHistory	CheckCriminalHistory

1003. Define Requirements

ATM_STC_007	CheckCriminal History	범죄이력을 조회하고 싶은 계좌를 입력하고 유효하지 않은 계좌를 입력했을 때 경고를

1009. Define System Test case

[Stage 1000 - 1006. Define Business Use Case]

Use Case	6. CheckTransactionHistory
Actor	User
Description	1. 입력한 계좌의 거래내역이 화면에 출력된 것을 확인한다. 2. 끝내기 버튼을 눌러 메인 화면으로 돌아간다.

Use Case	7. CheckTransactionHistory
Actor	User
Description	1. 범죄 이력을 조회하고자 하는 계좌를 입력하고 입력된 계좌번호가 유효하지 않다고 경고가 뜰 경우 다시 입력한다.

- CheckCriminalHistory로 수정이 필요하다.

Stage 2010 Revise Plan

- 이전 문서의 버전과 달라진 것이 많으나 이를 작성하지 않았다.
- 요구사항의 어떤 부분이 추가, 삭제, 수정되었는지 알 수 없다.
- 버전 별 바뀐 점에 대해서 작성할 필요가 있다.

[Stage 2030 – 2031. Define Essential Use Case]

Typical Courses of Events	(A): Actor , (S):System (S) 화면에 금액 입력 창을 띄운다. (A) 입금할 금액을 입력한다. (S) 비밀번호 입력 창을 띄운다. (A) 비밀번호를 입력한다. (S) 입력받은 비밀번호가 MediumCheck에서 입력받은 계좌의 비밀번호와 일치하는지 확인한다. . (S) 비밀번호가 일치할 시 ATM의 소속은행과 계좌의 소속은행을 비교하여 수수료를 계산한다. (S) 수수료를 제외한 금액만큼 계좌의 보유현금을 증가시킨다. (S) 계좌에 돈을 입금한 후 실행한 작업(입금)과 입금한 금액, 계좌의 잔액을 화면에 출력하고 로그에 저장한다.

- Typical Course of Events의 시나리오에 번호를 입력하여 순서를 명확히 할 필요가 있다.

Stage 2030 – 2033, 2034, 2037

- 2033. Define Domain Model

  - 1004에서 제시하였던 User 객체가 보이지 않는다.

- 2034. Refine Glossary

  - Class Diagram에선 Bankbook으로 표시되어 있으나, Glossary에는 Passbook으로 표시되었다. Bankbook과 Passbook을 혼용하여 혼란을 준다.

- 2037. Define State Diagrams

  - 이전 검증에서 수정 권고를 하였으나 문서에서 누락되어 있다.

[Stage 2030 – 2038. Refine System Test Case]

- ATM_STC_003/004/005_001

ATM_STC_003_001	Deposit	입금할 금액을 잘 입력받을 수 있는지 확인한다.
ATM_STC_004_001	Withdraw	출금할 금액을 잘 입력받을 수 있는지 확인한다.
ATM_STC_005_001	Transfer	송금할 금액을 잘 입력받을 수 있는지 확인한다.

- "**잘**"이라는 기준이 모호하다.
- 2031에 음수를 입력 받는 경우 Error처리한다고 명시되어 있으나, 이를 확인 하는 것이 누락되어 있다.

Stage 2040 – 2041, 2042

- 2041. Design Real Use Cases

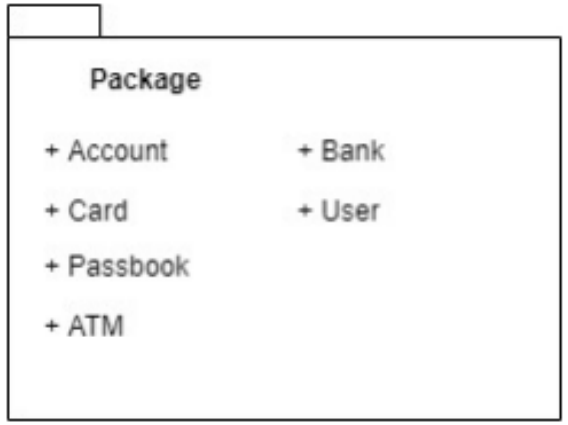
  - 2031과 동일한 이슈를 가지고 있다.

- 2042. Define Reports, UI and Storyboards

  - 이미지 제목 및 설명 누락되었다.

Stage 2040 – 2043. Refine System Architecture

Activity 2043. Refine System Architecture

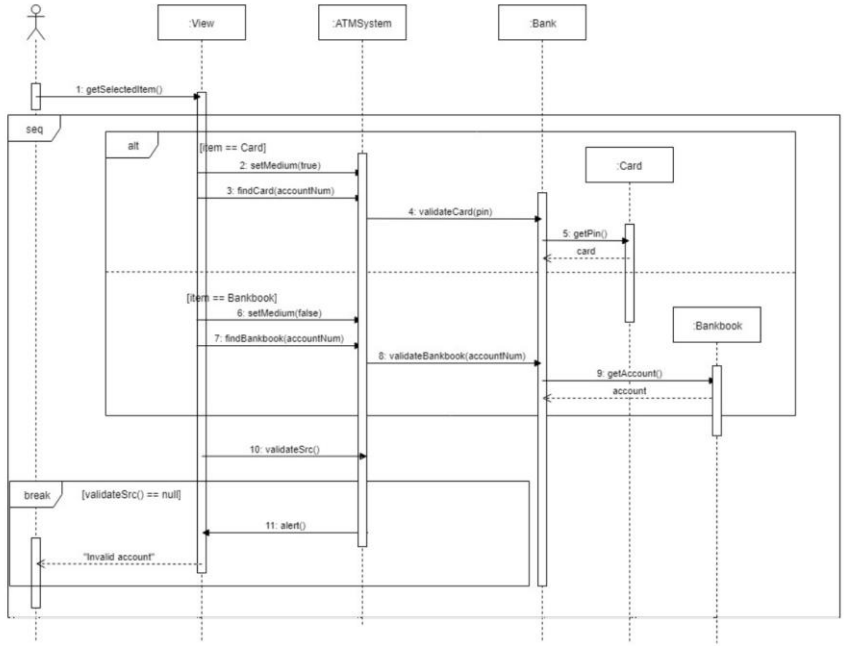


- 실제 소스상에 없는 User, Passbook, ATM 존재한다.

- 소스상에는 Bankbook, Terminate, View, AtmSystem 존재한다.

[Stage 2040 – 2044. Define Interaction Diagrams]

2. MediumCheck / 5. Transfer



2.MediumCheck

3. findCard / 7. findBankbook

: 실제 소스상에서 atm.setSrc를 이용해 AtmSystem 내부에서 findCard/findBankbook를 자체적으로 호출한다.

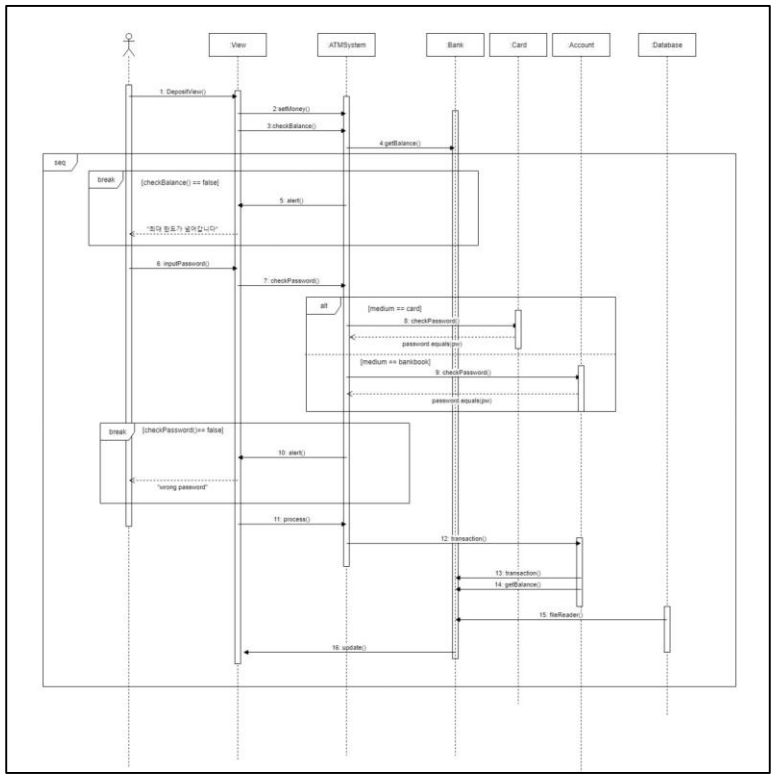
4. validateCard / 8. validateBankbook

: return이 명시되어있지 않다.

5. getPin : 실제 소스상에서 getPin은 card를 return 하지 않고 validateCard가 card를 return한다.

[Stage 2040 – 2044. Define Interaction Diagrams]

3. Deposit / 4. Withdraw / 5. Transfer



3.Deposit

12. transaction : process -> transfer로 가는 부분이 없다.

14. getBalance : 호출하고 return되는 과정이 생략되어 있다.

15. fileReader : 실제 소스에서 FileWriter를 호출한다.

Stage 2040 – 2045. Define Design Class Diagrams

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

Card	
소스	문서
pin	pin
account	account
password	Card
Card	checkPassword
getAccount	
getPin	
checkPassword	

Bankbook	
소스	문서
account	account
Bankbook	Bankbook
getAccount	

** 빨간색으로 처리한 항목은 소스와 문서가 일치하지 않는 항목이다 .*

Stage 2040 – 2045. Define Design Class Diagrams

Bank	
소스	문서
name	name
Bank	path
loadDataBase	cards
fileReader	bankbooks
vaildateBankbook	Bank
vaildateCard	loadDataBase
getBalance	fileReader
transaction	vaildateBankbook
getLogs	vaildateCard
getCriminalLogs	getBalance
	transaction
	getLogs
	getCriminalLogs
	getName

ATM System		
소스		문서
own	getLogs	src
view	getCriminalLogs	money
src	setProcess	medium
des	setMedium	des
amount	setSrc	criminalLogs
money	vaildateSrc	logs
fee	setDes	amount
medium	vaildateDes	AtmSystem
banks	setAmount	loadBanks
process	setMoeny	process
srcCard	findCard	checkBalance
AtmSystem	findBankbook	checkPassword
loadBanks		deposit
process		withdraw
checkBalance		transfer
checkPassword		vaildateSrc
deposit		vaildateDes
withdraw		findCard
transfer		findBankbook

** 빨간색으로 처리한 항목은 소스와 문서가 일치하지 않는 항목이다 .*

[Stage 2050 – 2051. Implement Class & Methods Definition]

- **Account Class**

  - 이전 보고서에서 찾아낸 부분중에 수정 안된 부분이 많다.

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 이력과 횟수를 반환한다,
Exceptional Courses of Events	

- 실제 프로그램에서는 **범죄시간**과 **횟수**를 반환한다.

[Stage 2050 – 2051. Implement Class & Methods Definition]

- **AtmSystem Class**

  - loadBanks()

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

```
37      private void loadBanks() {
38          File file = new File(System.getProperty("user.dir") + "/res");
39          File[] banks = file.listFiles();
40          if (banks != null) {
41              for (File bank : banks) {
42                  this.banks.add(new Bank(bank.getName()));
43              }
44          }
45      }
```

- 문서에는 Bank own을 input으로 받도록 나와있으나, 실제 코드에서는 어떠한 파라미터도 받지 않는다.

[Stage 2050 – 2051. Implement Class & Methods Definition]

- **AtmSystem Class**

  - checkBalance()

Output(Method)	boolean (amount + fee) < arc.getBalance()

문서상의 Output 값

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

실제 소스코드 Output 값

- 문서상의 Output값과 실제 코드와 차이가 있다.

[Stage 2050 – 2051. Implement Class & Methods Definition]

- **AtmSystem Class**

  - deposit()

Abstract Operation(Method)	계좌의 balance 를 amount - fee 만큼 증가시키고 log 에 msg 와 해당 거래내용을 업데이트한다.

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

- 문서와 실제코드에 대해 통일이 필요하다.

[Stage 2050 – 2051. Implement Class & Methods Definition]

- **AtmSystem Class**

  - setDes()

Abstract Operation(Method)	매체에 따라 findCard(accountNum)또는 findBankBook(accountNum)의 결과를 des에 저장한다.

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

- 문서와 실제코드에 대해 통일이 필요하다.

Stage 2050 – 2051. Implement Class & Methods Definition

- **Bank Class**

  - validateCard()

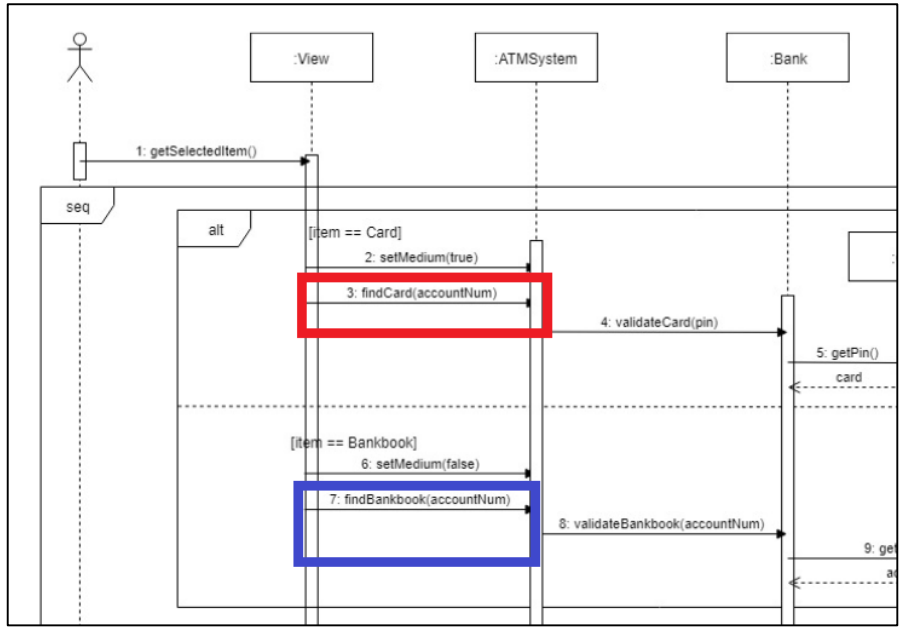
Abstract Operation(Method)	bank DB 의 통장중 accountNum 에 해당하는 카드가 존재하는지 확인하고 존재하면 해당 card 를 반환한다.
Exceptional Courses of Events	bank DB 의 통장중 accountNum 에 해당하는 카드가 존재하는지 확인하고 존재하지 않으면 null 을 반환한다.

```
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      }
```

  - 문서와 실제코드에 대해 통일이 필요하다.

Stage 2050 – 2052. Implement Windows

- **InputSrcView**



다이어그램

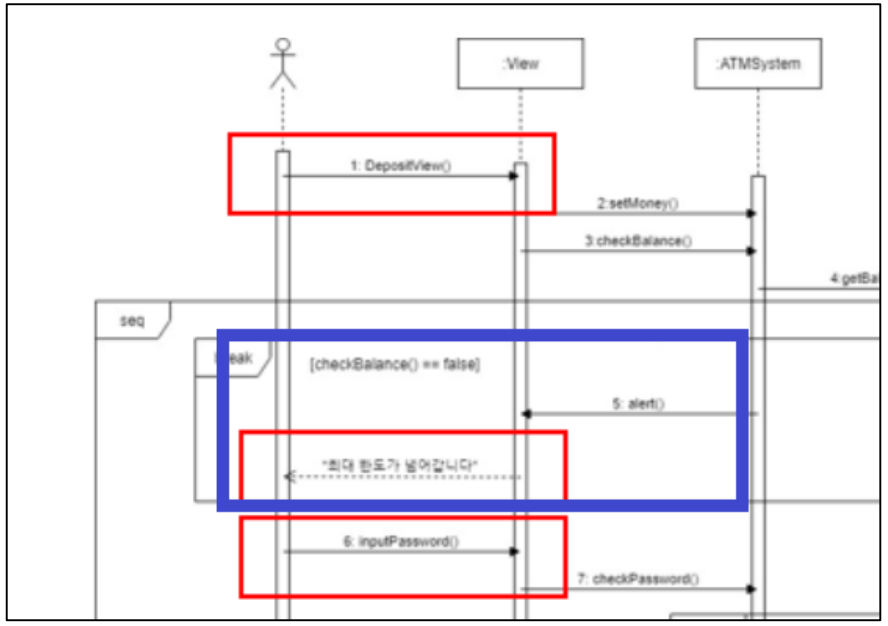
```
460             Object item = selectMedium.getSelectedItem();
461             if (item == null) {
462                 item = "Card";
463             }
464             if (item.equals("Card")) {
465                 atm.setMedium(true);
466             } else {
467                 atm.setMedium(false);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

실제코드

- 코드와 다이어그램 내용의 차이가 존재한다.

Stage 2050 – 2052. Implement Windows

- **alert “최대한도가 넘어갑니다”**



다이어그램

```
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                     }
```

실제코드

- 코드와 다이어그램 내용의 차이가 존재한다.

Stage 2050 – 2052. Implement Windows

- **alert “wrong password”**

다이어그램

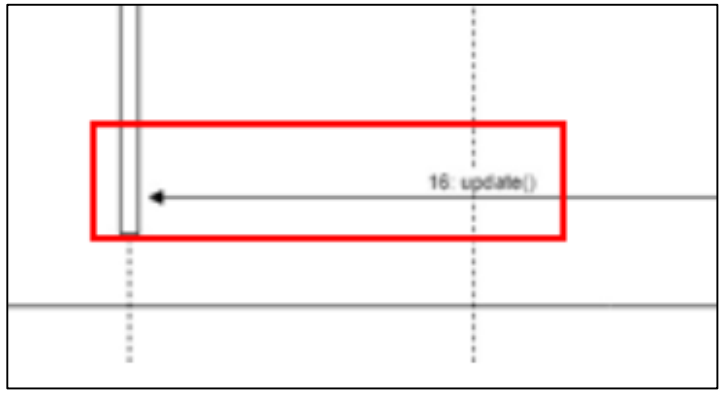
```
503             nextBtn.addActionListener((ActionEvent e) -> {
504                 String pw = String.valueOf(pwInput.getPassword());
505                 if (atm.checkPassword(pw)) {
506                     atm.process();
507                     nextPanel(process.next());
508                    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

실제코드

- 코드와 다이어그램 내용의 차이가 존재한다.

Stage 2050 – 2052. Implement Windows

- **Update**



다이어그램

Name	Update
Type	GUI
Responsibilities	입금이 완료된 후 결과

문서

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

Stage 2050 – 2052. Implement Windows

- **alert “잔액이 부족합니다”**

다이어그램

```
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            });
```

실제코드

- 코드와 다이어그램 내용의 차이가 존재한다.

- **alert “잔액이 부족합니다”**

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 Notes의 내용이 중복되므로, 지워야 한다.

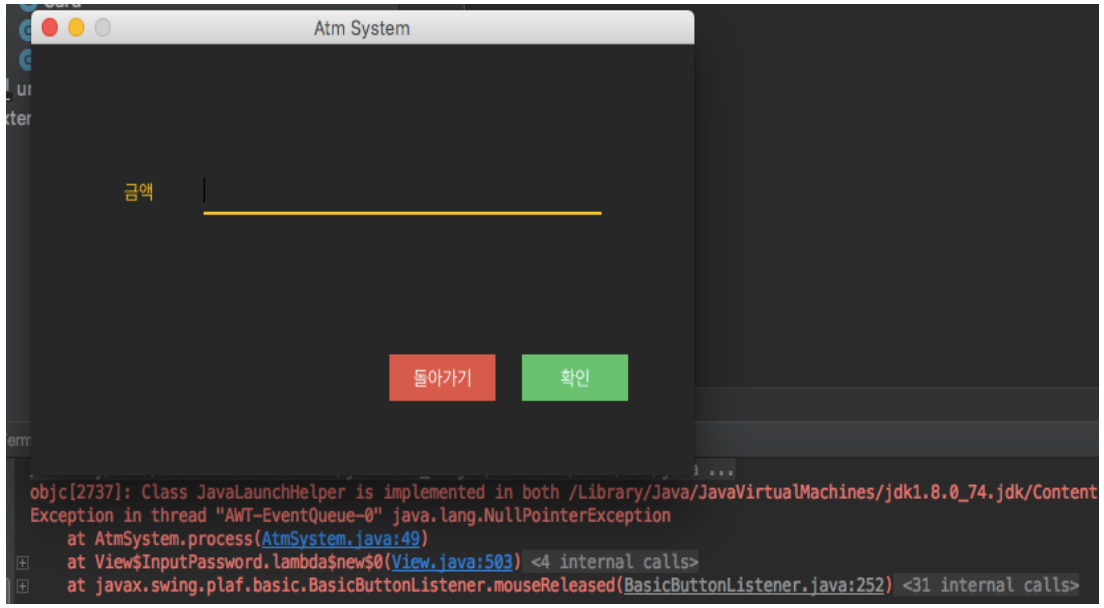
Stage 2060 – 2063. System Testing

- **ATM_STC_005**

- 송금할 계좌를 입력하면, 정상적인 계좌임에도 불구하고 계좌가 존재하지 않는다는 경고가 발생한다. 결과적으로 **송금 기능이 동작하지 않는다**.
- ATM_STC_005_002~005까지 모두 Pass에서 Fail로 표기해야한다.

Stage 2060 – 2063. System Testing

- **ATM_STC_005**



- 초기 프로그램 실행 후, 거래기록 조회를 실행해보면, AtmSystem.process에서 에러가 발생한다. **따라서 거래기록을 조회할 수 없다**.
- 프로그램을 실행하는 동안에 발생된 거래 기록에 대해서는 확인할 수 없고, **프로그램 재실행 이후에 확인**이 가능하다.
- 여러 오류가 발생하고 있기 때문에 Pass가 아닌 Fail로 표기해야 한다.

Stage 2060 – 2063. System Testing

- **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

2. 2nd System Test

Category-Partition Test

- Table Units & Representative Values

Group	Category	Description	Value	Number
Validate	Account	계좌의 존재 유무 확인	계좌번호가 존재할때 / 계좌번호가 존재하지 않을때	1000 / 1001
	Card	카드의 존재 유무 확인	카드가 존재할때 / 카드가 존재하지 않을때	1100 / 1101
	Password	비밀번호 일치 확인	일치하는 비밀번호/일치하지 않는 비밀번호	1200 / 1201
	Account Balance	계좌 내 잔고 확인	계좌의 잔고가 출금하려는 잔고보다 많다(충분) / 계좌의 잔고가 출금하려는 잔고보다 적다(불충분)	1300 / 1301
Transaction	Transaction	입금	타행인경우 거래 금액에서 수수료를 제한 금액을 입금한다(잔액 추가) / 당행인경우 거래 금액을 입금한다(잔액 추가)	2000 / 2001
		출금	타행인경우 거래 금액에서 수수료를 더한 금액을 출금한다(잔액 감소) / 당행인경우 거래 금액을 출금한다(잔액 감소)	2100 / 2101
		송금	타행인경우 거래 금액에서 수수료를 더한 금액을 출금하고(잔액 감소), 송금 대상 계좌의 잔액을 거래 금액만큼 증가시킨다 / 당행인경우 거래 금액을 출금하고(잔액 감소), 송금 대상 계좌의 잔액을 거래 금액만큼 증가시킨다	2200 / 2201
Print	Transaction Result	입,출, 송금 결과 출력	입금, 출금, 송금 후 계좌에 남은 실제 잔액을 화면에 표시한다	3000
	Criminal History	범죄기록 출력	범죄기록이 있는 경우 / 범죄기록이 없는 경우	3100 / 3101
	Transaction History	거래내역 출력	거래내역이 있는 경우 / 거래내역이 없는 경우	3200 / 3201
Select	Medium	매체 선택 (카드/통장)	카드를 선택한 경우 / 통장을 선택한 경우	4000 /4001
Input	Account Number	계좌번호 입력	숫자로만 입력한 경우 / 기타 문자와 숫자를 섞어 쓴 경우(숫자 이외의 문자가 들어간 경우) / 미입력	5000 / 5001 / 5002
	Card number	카드번호 입력	숫자로만 입력한 경우 / 기타 문자와 숫자를 섞어 쓴 경우(숫자 이외의 문자가 들어간 경우) / 미입력	5100 / 5101 / 5102
	Password Number	비밀번호 입력	비밀번호를 입력받는다 / 입력받지 않은 경우 / 미입력	5200 / 5201 / 5202
	Transaction Amount	거래할 금액 입력	숫자로만 입력한 경우 / 기타 문자와 숫자를 섞어 쓴 경우(숫자 이외의 문자가 들어간 경우) / 미입력/ 음수	5300 / 5301 / 5302 / 5303
Access	Transaction Log	txt 파일에 접근	거래기록 파일이 있는 경우 / 거래기록 파일이 없는 경우	6000 / 6001
	Criminal Log	txt 파일에 접근	범죄기록 파일이 있는 경우 / 범죄기록 파일이 없는 경우	6100 / 6101

Category-Partition Test

- **Error Constraints 적용**

Group	Category	Error Constraints
Validate	Account	[error]계좌번호 미존재
	Card	[error] 카드번호 미존재
	Password	[error] 비밀번호 불일치
	Account Balance	[error]계좌잔고 불충분
Input	Account Number	[error] 숫자 이외의 문자 [error] 미입력
	Card number	[error] 숫자 이외의 문자 [error] 미입력
	Password Number	[error] 숫자 이외의 문자 [error] 미입력
	Transaction Amount	[error] 숫자 이외의 문자 [error] 미입력 [error] 음수
Access	Transaction Log	[error] log 無
	Criminal Log	[error] log 無

- Test Frame 개수

적용 전 : 442368개
Error Constraints 적용 후 : 143개

▷ 99.97% 감소

Category-Partition Test

- **Single Constraints 적용**

Group	Category	Single Constraints
Select	Medium	[singe] Card 선택

- Test Frame 개수

적용 전 : 143개
Single Constraints 적용 후 : 80개

▷ 40.06% 감소

Category-Partition Test

- **Property Constraints 적용**

Group	Category	Property Constraints	Description
Select	Medium	property BOOK	if BOOK: 매체로 통장(계좌번호)을 사용하는 경우
		property CARD	if CARD: 매체로 카드를 사용하는 경우
Access	Transaction Log	proerty HISTORY	if ACTION: 거래내역이 존재하는 경우
	Criminal Log	property CRIM	if CRIM: 범죄내역이 존재하는 경우
Transcation	Transaction	property DIFER	if DIFER: 타행거래인 경우 if !DIFER: 당행거래인 경우

- Test Frame 개수

적용 전 : 80개
Property Constraints 적용 후 : 32개

▷32.00% 감소

Category-Partition Test

- **Test Result(1/2)**

	Test Case	Result
1	4000	P
2	6001	F
3	6101	P
4	1001	P
5	1101	P
6	1201	P
7	1301	P
8	5001	P
9	5002	P
10	5101	P
11	5102	P
12	5201	P
13	5202	P
14	5301	P
15	5302	P
16	5203	P
17	4001.6000.1000.1200.1300.2000.3000.3101.3200.5000.5200.5300	P

Category-Partition Test

- **Test Result(2/2)**

18	4001.6000.1000.1200.1300.2000.3000.3101.3201.5000.5200.5300	F
19	4001.6000.1000.1200.1300.2001.3000.3101.3200.5000.5200.5300	P
20	4001.6000.1000.1200.1300.2001.3000.3101.3201.5000.5200.5300	F
21	4001.6000.1000.1200.1300.2101.3000.3101.3200.5000.5200.5300	P
22	4001.6000.1000.1200.1300.2101.3000.3101.3201.5000.5200.5300	P
23	4001.6000.1000.1200.1300.2201.3000.3101.3200.5000.5200.5300	F
24	4001.6000.1000.1200.1300.2201.3000.3101.3201.5000.5200.5300	F
25	4001.6100.1000.1200.1300.2000.3000.3100.3201.5000.5200.5300	F
26	4001.6100.1000.1200.1300.2000.3000.3101.3201.5000.5200.5300	F
27	4001.6100.1000.1200.1300.2001.3000.3101.3201.5000.5200.5300	F
28	4001.6100.1000.1200.1300.2101.3000.3100.3201.5000.5200.5300	P
29	4001.6100.1000.1200.1300.2101.3000.3101.3201.5000.5200.5300	P
30	4001.6100.1000.1200.1300.2201.3000.3100.3201.5000.5200.5300	F
31	4001.6100.1000.1200.1300.2201.3000.3101.3201.5000.5200.5300	F

Pairwise Test

- **Test Case**

Bank	Action	Log File	PW	Account
Same	Deposit	Exist both	Vaild	Vaild
Different	Withdraw	Exist Criminal only	Not Vaild	Not Vaild
	Transfer	Exist History only		
	History	Non		
	Criminal History			

Pairwise Test

- **Test Result**

	Bank	Action	Log File	PW	Account	Result
1	Same	Criminal	Non	-	Not Vaild	P
2	Different	Criminal	Exist both	-	Vaild	P
3	Same	History	Exist History Only	Vaild	Vaild	F
4	Different	Deposit	Exist History Only	Not Vaild	Not Vaild	P
5	Same	Withdraw	Exist both	Vaild	Not Vaild	P
6	Different	Criminal	Exist History Only	-	Not Vaild	P
7	Same	Criminal	Exist Criminal Only	-	Vaild	P
8	Different	Withdraw	Exist Criminal Only	Not Vaild	Vaild	P
9	Different	History	Non	Not Vaild	Vaild	F
10	Same	Transfer	Exist Criminal Only	Vaild	Not Vaild	F
11	Different	Withdraw	Non	Not Vaild	Vaild	P
12	Same	History	Exist both	Vaild	Not Vaild	F
13	Same	History	Exist Criminal Only	Not Vaild	Not Vaild	F
14	Same	Deposit	Non	Vaild	Vaild	F
15	Different	Transfer	Exist both	Not Vaild	Vaild	P
16	Same	Withdraw	Exist History Only	Vaild	Vaild	P
17	Same	Deposit	Exist both	Vaild	Vaild	F
18	Same	Deposit	Exist Criminal Only	Vaild	Vaild	P
19	Same	Transfer	Exist History Only	Vaild	Not Vaild	F
20	Different	Transfer	Non	Not Vaild	Not Vaild	F

- Pairwise Test Case : 20
- Pass 11 / Fail 9

▷11/20 = 55% Pass

Brute Force Test

- **Test Case & Result(1/2)**

Number	Test Case	Result
1	입금 시 입금 금액에 음수를 넣고 확인버튼을 누른다.	P
2	출금 시 출금 금액에 음수를 넣고 확인버튼을 누른다.	P
3	송금 시 송금 금액에 음수를 넣고 확인버튼을 누른다.	P
4	입금 시 입금 금액에 숫자 이외의 문자를 넣고 확인버튼을 누른다.	P
5	출금 시 출금 금액에 숫자 이외의 문자를 넣고 확인버튼을 누른다.	P
6	송금 시 송금 금액에 숫자 이외의 문자를 넣고 확인버튼을 누른다.	P
7	입금 시 입금 금액에 int 범위 이외의 숫자를 넣고 확인버튼을 누른다.	P
8	출금 시 출금 금액에 int 범위 이외의 숫자를 넣고 확인버튼을 누른다.	P
9	송금 시 송금 금액에 int 범위 이외의 숫자를 넣고 확인버튼을 누른다.	P
10	입금 시 입금 금액에 아무것도 넣지 않고 확인버튼을 누른다.	P
11	출금 시 출금 금액에 아무것도 넣지 않고 확인버튼을 누른다.	P
12	송금 시 송금 금액에 아무것도 넣지 않고 확인버튼을 누른다.	P

Brute Force Test

- **Test Case & Result(2/2)**

13	입금 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	F
14	출금 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	F
15	송금 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	F
16	거래 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	F
17	범죄 이력이 많을 경우 1초 이내에 처리 되는지 확인 한다.	F
18	거래 내역이 많을 경우, 거래 내역 조회 기능에서 Simple한 UI/UX를 가지는지 확인한다.	F
19	범죄 내역이 많을 경우, 범죄 이력 조회 기능에서 Simple한 UI/UX를 가지는지 확인한다.	F
20	반복적인 거래를 하는 경우, Simple한 UI/UX를 가지는지 확인한다.	F

8 / 20 = 40% Pass

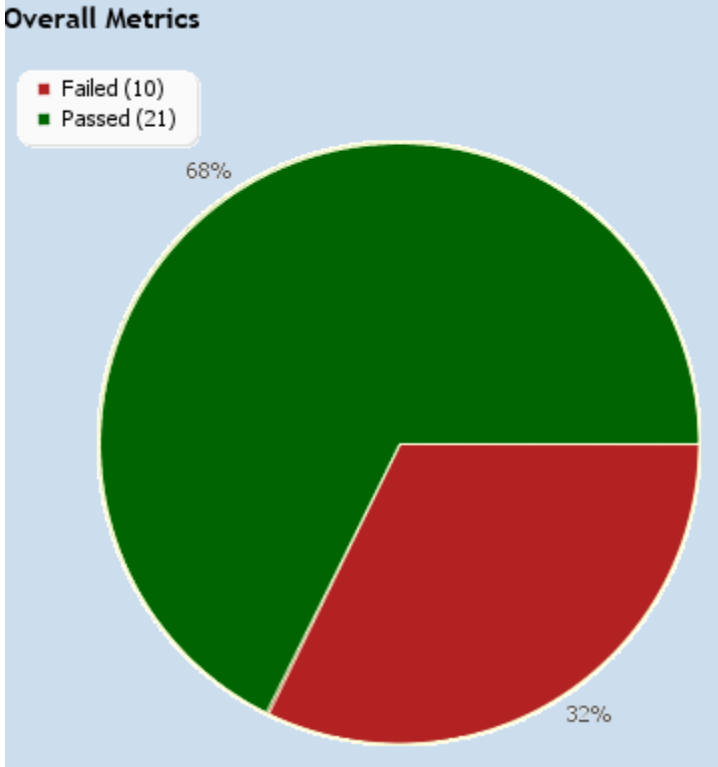
Brute Force Test

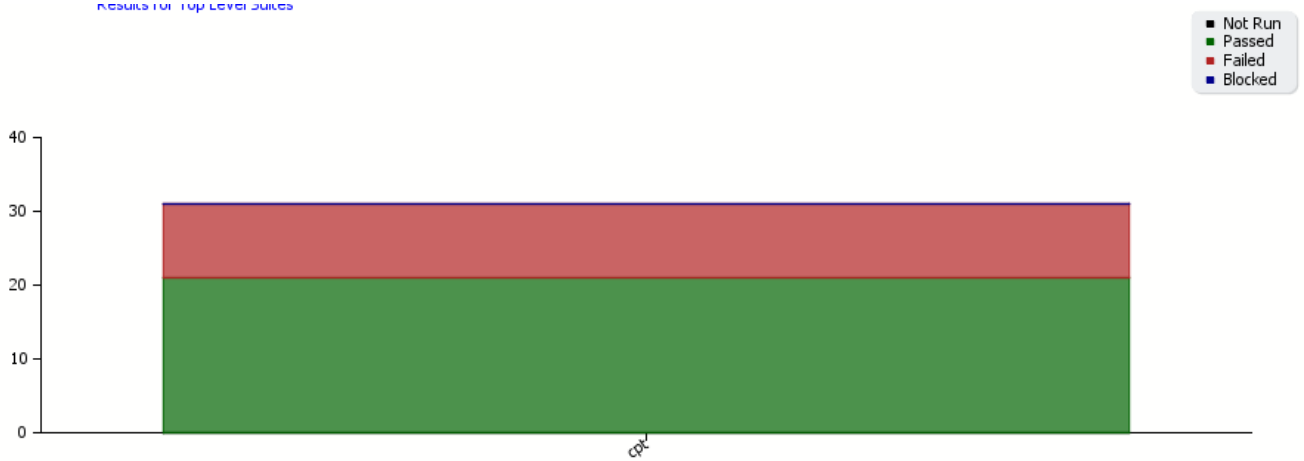
- **Failed Case Report**

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

TestLink

- **Result**





Overall Build Status

Build	Assigned	Not Run	[%]	Passed	[%]	Failed	[%]
build	31	0	0.0	21	67.7	10	32.3

3. Static Analysis

CTIP - Github Issue Tracker

[CPT]4001.6000.1000.1200.1300.2201.3000.3101.3201.5000.5200.5300 CPT bug
#35 opened 10 hours ago by 3ssarah 2차 System Test...

[CPT]4001.6000.1000.1200.1300.2201.3000.3101.3200.5000.5200.5300 CPT bug
#34 opened 10 hours ago by 3ssarah 2차 System Test...

[CPT]4001.6000.1000.1200.1300.2001.3000.3101.3201.5000.5200.5300 CPT bug
#33 opened 10 hours ago by 3ssarah 2차 System Test...

[CPT]4001.6000.1000.1200.1300.2000.3000.3101.3201.5000.5200.5300 CPT bug
#32 opened 10 hours ago by 3ssarah 2차 System Test...

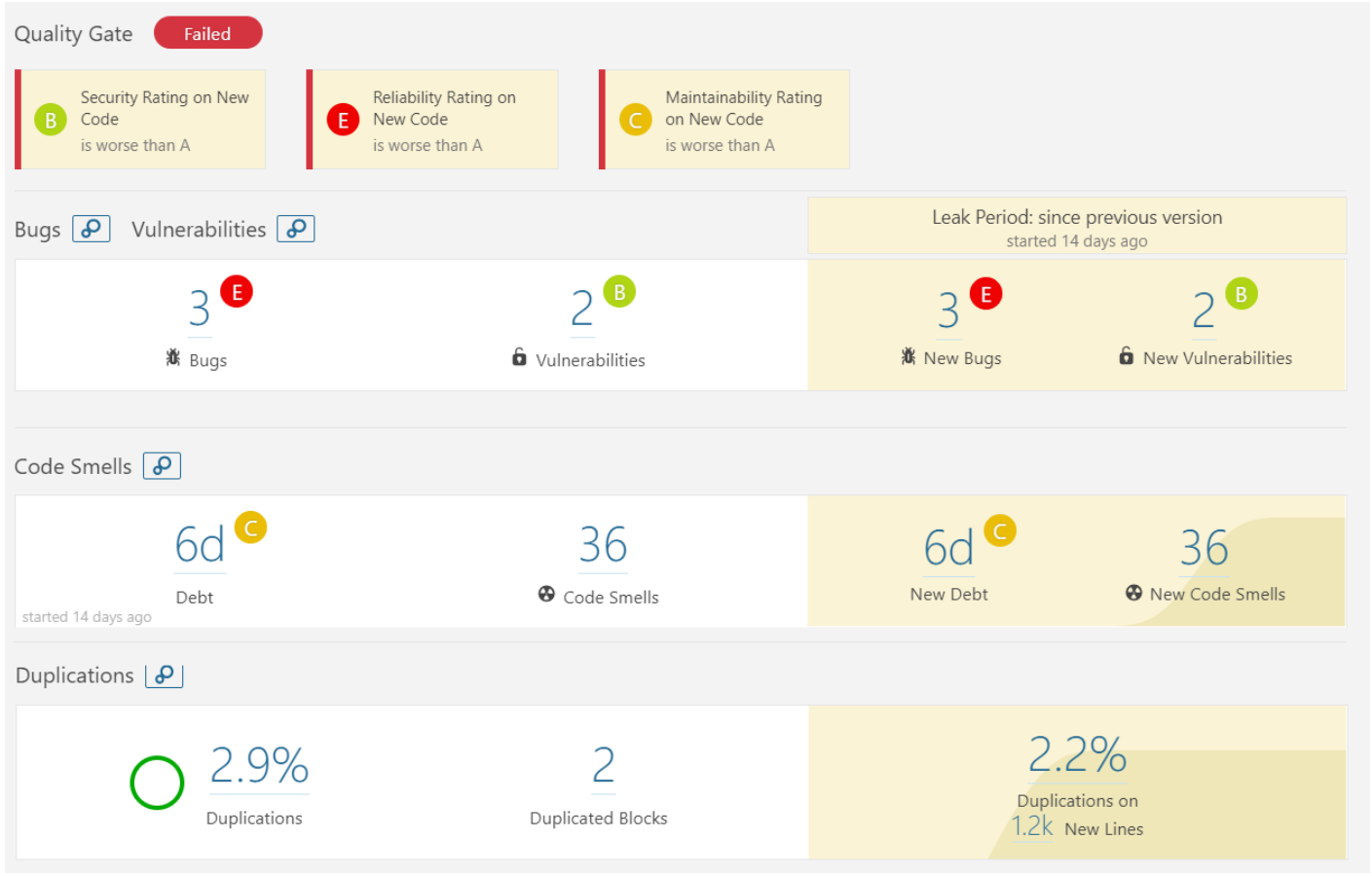
[CPT]6001 CPT bug
#31 opened 10 hours ago by 3ssarah 2차 System Test...

[BFT]반복적인 거래를 하는 경우, Simple한 UI/UX bug
#30 opened 10 hours ago by 3ssarah

[Brute Force Testing]범죄 내역이 많을 때, 범죄 이력 조회 기능이 Simple한 UI/UX? bug
#20 opened 3 days ago by 3ssarah 1차 System test...

[Brute Force Testing]거래 내역이 많을 때, 거래 내역 조회 기능이 Simple한 UI/UX? bug
#19 opened 3 days ago by 3ssarah 1차 System test...

CTIP - SonarQube



CTIP – 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	

Account.java:6, ImmutableField, Priority: Normal

Private field 'password' could be made final; it is only initialized in the declaration or constructor.

Identifies private fields whose values never change once they are initialized either in the declaration of the field or by a constructor. This helps in converting existing classes to becoming immutable ones.

```
public class Foo {
  private int x; // could be final
  public Foo() {
      x = 7;
  }
  public void foo() {
     int a = x + 2;
  }
}
```

Account.java:8, MethodArgumentCouldBeFinal, Priority: Normal

Parameter 'accountNum' is not assigned and could be declared final.

A method argument that is never re-assigned within the method can be declared final.

```
public void foo1 (String param) {       // do stuff with param never assigning it

}

public void foo2 (final String param) { // better, do stuff with param never assigning it

}
```

CTIP - 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	

Bank.java:23, DM_DEFAULT_ENCODING, Priority: High

Dm: Found reliance on default encoding in Bank.fileReader(String): new java.io.FileReader(File)

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Bank.java:24, OS_OPEN_STREAM, Priority: Normal

OS: Bank.fileReader(String) may fail to close stream

The method creates an IO stream object, does not assign it to any fields, pass it to other methods that might close it, or return it, and does not appear to close the stream on all paths out of the method. This may result in a file descriptor leak. It is generally a good idea to use a `finally` block to ensure that streams are closed.

Bank.java:74, OBL_UNSATISFIED_OBLIGATION_EXCEPTION_EDGE, Priority: Normal

OBL: Bank.transaction(Account, int, String) may fail to clean up java.io.Writer on checked exception

This method may fail to clean up (close, dispose of) a stream, database object, or other resource requiring an explicit cleanup operation.

In general, if a method opens a stream or other resource, the method should use a try/finally block to ensure that the stream or resource is cleaned up before the method returns.

This bug pattern is essentially the same as the OS_OPEN_STREAM and ODR_OPEN_DATABASE_RESOURCE bug patterns, but is based on a different (and hopefully better) static analysis technique. We are interested is getting feedback about the usefulness of this bug pattern. To send feedback, either:

- send email to findbugs@cs.umd.edu
- file a bug report: http://findbugs.sourceforge.net/reportingBugs.html

In particular, the false-positive suppression heuristics for this bug pattern have not been extensively tuned, so reports about false positives are helpful to us.

See Weimer and Necula, *Finding and Preventing Run-Time Error Handling Mistakes*, for a description of the analysis technique.

Bank.java:74, DM_DEFAULT_ENCODING, Priority: High

Dm: Found reliance on default encoding in Bank.transaction(Account, int, String): new java.io.FileWriter(File, boolean)

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Bankbook.java:5, URF_UNREAD_FIELD, Priority: Normal

UrF: Unread field: Bankbook.account

This field is never read. Consider removing it from the class.

Card.java:5, URF_UNREAD_FIELD, Priority: Normal

UrF: Unread field: Card.account

This field is never read. Consider removing it from the class.

View.java:320, SIC_INNER_SHOULD_BE_STATIC, Priority: Normal

SIC: Should View$Btn be a _static_ inner class?

This class is an inner class, but does not use its embedded reference to the object which created it. This reference makes the instances of the class larger, and may keep the reference to the creator object alive longer than necessary. If possible, the class should be made static.

View.java:429, SIC_INNER_SHOULD_BE_STATIC, Priority: Normal

SIC: Should View$NextBtn be a _static_ inner class?

This class is an inner class, but does not use its embedded reference to the object which created it. This reference makes the instances of the class larger, and may keep the reference to the creator object alive longer than necessary. If possible, the class should be made static.

CTIP - 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	

Account.java:8, EmptyLineSeparatorCheck, Priority: Normal

'CTOR_DEF' should be separated from previous statement.

Checks for empty line separators after header, package, all import declarations, fields, constructors, methods, nested classes, static initializers and instance initializers.

Account.java:14, EmptyLineSeparatorCheck, Priority: Normal

'METHOD_DEF' should be separated from previous statement.

Checks for empty line separators after header, package, all import declarations, fields, constructors, methods, nested classes, static initializers and instance initializers.

Account.java:18, EmptyLineSeparatorCheck, Priority: Normal

'METHOD_DEF' should be separated from previous statement.

Checks for empty line separators after header, package, all import declarations, fields, constructors, methods, nested classes, static initializers and instance initializers.

Account.java:22, EmptyLineSeparatorCheck, Priority: Normal

'METHOD_DEF' should be separated from previous statement.

Checks for empty line separators after header, package, all import declarations, fields, constructors, methods, nested classes, static initializers and instance initializers.

Account.java:26, EmptyLineSeparatorCheck, Priority: Normal

'METHOD_DEF' should be separated from previous statement.

Checks for empty line separators after header, package, all import declarations, fields, constructors, methods, nested classes, static initializers and instance initializers.

Account.java:30, EmptyLineSeparatorCheck, Priority: Normal

'METHOD_DEF' should be separated from previous statement.

Checks for empty line separators after header, package, all import declarations, fields, constructors, methods, nested classes, static initializers and instance initializers.

- **규칙 & 심각도**

규칙 모음	포함된 규칙 수	위배 수	무시된 위배 수	설명
Sun_Code_Conventions_for_Java	27	159	0	파일이름, 파일 구성, 들여쓰기, 주석, 선언, 문장, 이름 규칙, 프로그래밍 practice등 Sun에서 따르기를 권고하는 표준 코딩 가이드라인
CODESCROLL_JAVA_RULES	72	115	0	일반적인 자바프로그램에서 지켜야 할 규칙과 권고들

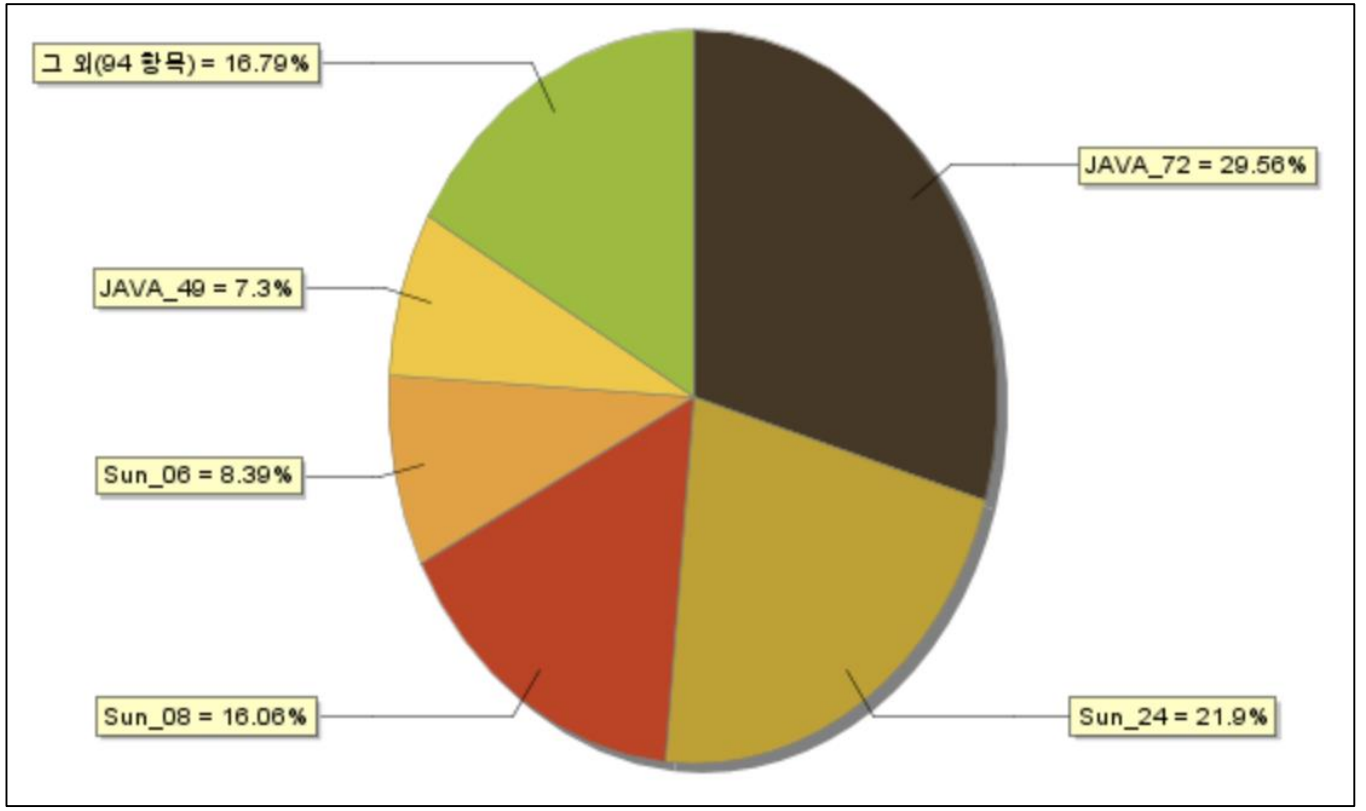
심각도	위배 수	무시된 위배 수	포함된 규칙 수	위배 규칙 수	*SCR	설명
매우높음	2	0	20	1	95%	프로그램에 매우 심각한 영향을 미치므로 오류 수정이 필요함
높음	113	0	46	4	91.3%	프로그램에 심각한 영향을 줄 수 있으므로 오류 수정이 강력히 권장됨
낮음	7	0	10	3	70%	프로그램에 영향을 줄 수 있으므로 오류 수정이 권장됨
매우낮음	152	0	23	6	73.91%	프로그램에 큰 영향은 없지만 오류 수정이 권장됨
기타	0	0	0	0	0%	심각도가 정의되지 않음

CodeScroll

소스	위배 수	무시된 위배 수	위배 규칙 수	*RCR	**RVD
View.java	151	0	9	90.91%	0.31
Bank.java	51	0	12	87.88%	0.43
AtmSystem.java	45	0	9	90.91%	0.29
Account.java	11	0	4	95.96%	0.34
Card.java	7	0	4	95.96%	0.36
Bankbook.java	5	0	4	95.96%	0.55
Terminate.java	4	0	4	95.96%	0.66

View.java에서 가장 많이 위배 항목 발견

- 위배항목 TOP 5 비율


그 외(94 항목) = 16.79%
JAVA_72 = 29.56%
JAVA_49 = 7.3%
Sun_06 = 8.39%
Sun_08 = 16.06%
Sun_24 = 21.9%

CodeScroll

규칙	규칙 모음	위배 수	무시된 위배 수	규칙 설명
Sun_03	Sun_Code_Conventions_for_Java	7	0	소스 파일 시작의 C 스타일 주석 검사
Sun_04	Sun_Code_Conventions_for_Java	7	0	파일에 package 선언이 있는지 검사
JAVA_71	CODESCROLL_JAVA_RULES	7	0	클래스의 설명 주석이 있어야 함
Sun_09	Sun_Code_Conventions_for_Java	5	0	필드 hide 제한
JAVA_70	CODESCROLL_JAVA_RULES	5	0	변수 hiding 금지
JAVA_44	CODESCROLL_JAVA_RULES	2	0	printStackTrace 메서드 사용 금지
Sun_10	Sun_Code_Conventions_for_Java	1	0	지역 변수 선언 시 초기화 검사
Sun_26	Sun_Code_Conventions_for_Java	1	0	괄호 안의 수식에 연산자 혼용 금지
JAVA_57	CODESCROLL_JAVA_RULES	0	0	null로 초기화된 인스턴스 역참조 금지
JAVA_26	CODESCROLL_JAVA_RULES	0	0	catch 절에서 MissingResourceException을 바로 출력 금지

일부 위배 목록

4. Overall

System Test Result

- **Category-Partition-Test**

  21/31 = **67.74%** Pass

- **Pairwise Test**

  11/20 = **55%** Pass

- **Brute Force Test**

  8/20 = **40%** Pass

Summary

- ✓ Specification의 **구체적인 기준**이 제시되어 있지 않고 모호한 것이 많다.
- ✓ 문서에서 용어의 **통일이 되지 않은 경우**가 종종 존재한다.
- ✓ 이전 단계에서 수정 사항을 권고하였으나, **수정되지 않은 부분**이 다수 존재한다.
- ✓ 성능과 사용자 인터페이스 같은 **non functional requirement**를 만족시키지 못하였다.
- ✓ 문서와 실제 작성된 코드간에 **일치하지 않는 부분**이 다수 존재한다.

[THANK YOU]