Software Verification

2nd System Testing

Static Analysis

Team 5
201410373 고예은
201411266 김수현
201411268 김아름

Index

1. Summary of 1st System Testing
2. Feedback
3. 2nd System Testing
4. Static Analysis
5. Overall

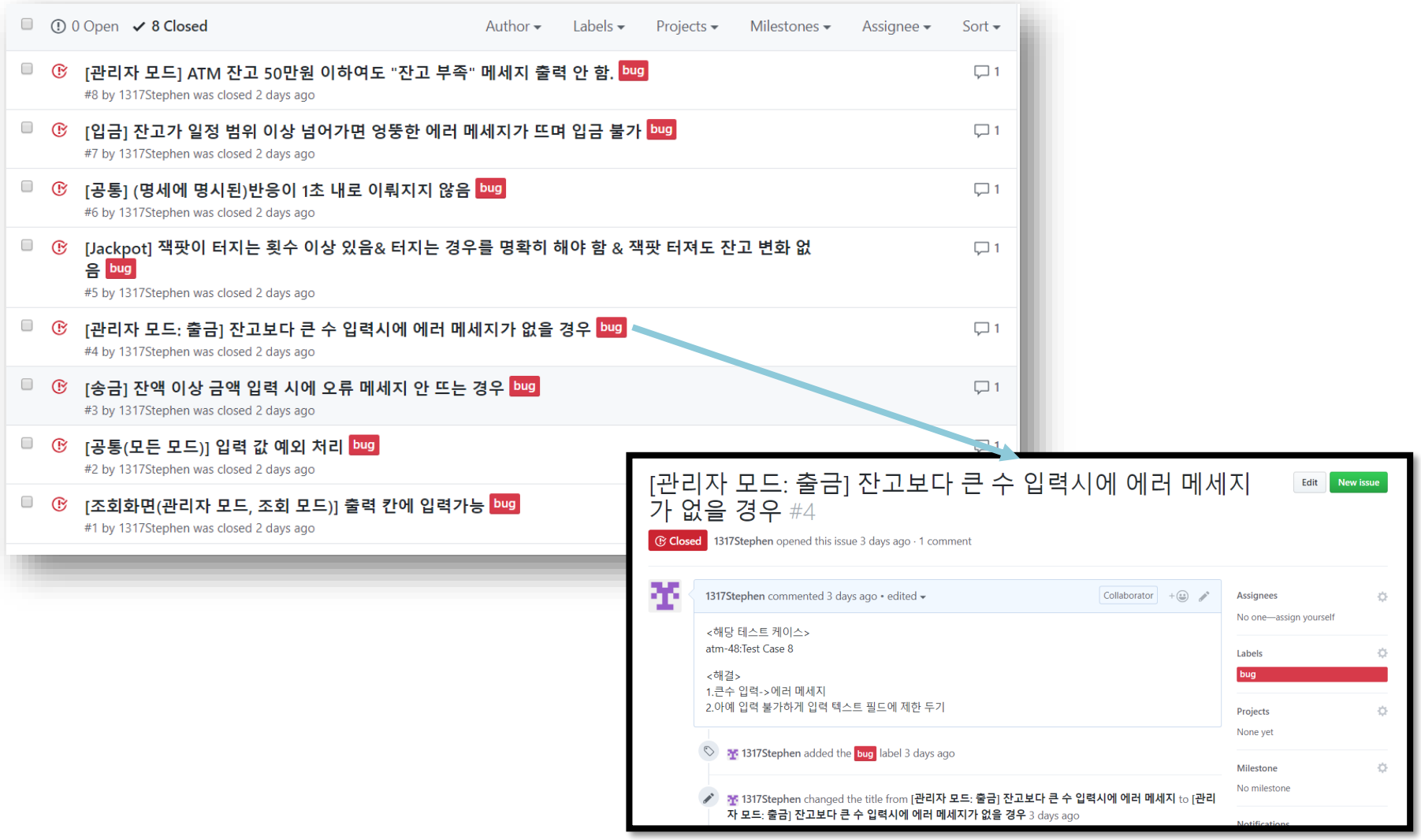
Summary of 1st System Testing

- 109 Test cases, 77 Fail (30% pass)
- 가장 큰 문제는 **예외처리**
- 보고서의 일부와 Test cases의 P/F 정리한 문서를 전달
- 중요 이슈 7개를 선정해 GitHub Issue에 등록

Test Suite	Test Case	Result	Summary
Manager	관리자 프로세스에서 5번 반복하여 입/출금 할 수 있다.	Failed	잔고가 어느 일정 범위 이상 커지면 "잔고 확인하세요"라는 오류 메시지가 출력된다. (잔고 최대값 명세에 존재하지 않음)
Manager	관리자 프로세스에서 10번 반복하여 입/출금 할 수 있다.	Failed	잔고가 어느 일정 범위 이상 커지면 "잔고 확인하세요"라는 오류 메시지가 출력된다. (잔고 최대값 명세에 존재하지 않음)
User	프로그램 초기 실행 시 난수가 제대로 생성된다.	Passed	
User	출금 10번 이내에 Jackpot이 터지는 지 확인한다.	Passed	
User	Jackpot이 터지면 해당 User에게 5만원을 추가 인출하며 ATM의 잔고에서 5만원이 빠져나간다.	Failed	Jackpot이 터져도 ATM의 잔고에는 변화가 생기지 않는다. ATM 잔고가 0일 때도 예외처리 필요
User	1일 1명만 Jackpot이 터진다.	Failed	초기 실행 후 Jackpot이 한 번 터진 후에도 다시 터진다.
User	출금 시에만 Jackpot이 터진다.	Failed	송금 시 사용자의 비밀번호를 입력하고 나서도 Jackpot이 터지는 경우 존재
User	하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.	Failed	같은 계좌로 출금을 5회 이상 반복 했을 때, 연속으로 2번 잭팟이 터지는 경우가 있다
User	모든 입력에 대한 기기의 반응은 1초 이내로 이루어진다.	Failed	integer 이내의 정수값이 아니라면, 1초 이내로 반응이 없다(반응이 아예 없다)
User	거래는 1분 이내에 이루어져야 한다.	Passed	
User	사용자가 보기 편한 화면을 제공해야 한다.	Passed	
User	세 번 이상 연속적으로 출금할 수 있다.	Passed	
User	세 번 이상 연속적으로 입금할 수 있다.	Failed	잔고가 어느 일정 범위 이상 커지면 "비밀번호 확인하세요"라는 오류 메시지가 출력된다.
User	세 번 이상 연속적으로 송금할 수 있다.	Failed	3번 이상 송금 하려고 하면 알맞은 비밀번호여도 오류 메세지가 나온다
User	세 번 이상 연속적으로 조회할 수 있다.	Passed	
User	출금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Failed	출금 시 minus 정수 값을 금액으로 입력하면 해당 정수의 절대값만큼 계좌의 잔액이 증가한다. 예외처리 필요
User	입금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Failed	입금 시 minus 정수 값을 금액으로 입력하면 해당 정수의 절대값만큼 계좌의 잔액이 감소한다. 예외처리 필요
User	송금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Failed	송금 시 minus 정수 값을 금액으로 입력하면 해당 정수의 절대값만큼 계좌의 잔액이 증가한다. 예외처리 필요
Manager	ATM 잔고가 일정수준(50만원) 이하가 되면 잔고부족 메시지를 출력한다.	Failed	ATM 잔고가 50만원 이하가 되어도 아무 메시지도 출력되지 않는다.

Summary	Jackpot에 관한 버그 수정 필요
	입금 기능의 경우 ATM과 계좌 잔고의 최대값을 명세에 서술하지 않아 생긴 버그가 대부분 - 명세를 수정해주세요!
	영수증, 명세서 출력, 잔고 알림 등의 Functional Requirements를 개발하지 않을 계획이라면 명세에 명확히 써주세요!
	ID, PW, 금액 입력 등의 input에 대해 음수, Integer 범위를 넘어가는 값, 숫자가 아닌 문자 등의 다른 키보드 입력을 위한 예외처리가 필요합니다. (=대부분의 Category Partition Test Fail 원인)
	명세와 구현 다른 부분 문서 수정해주세요!
	*pairwise는 CPT와 거의 같은 조건으로 진행되기 때문에 따로 넣지 않았습니다!

Summary of 1st System Testing



Feedback

- 빠르고 친절한 피드백
  - SV 팀의 잦은 연락과 이에 따른 많은 물음들을 빨리 해결해 주기 위해 노력

Feedback

- 109 Test Cases, 109 Pass (30% → 100% pass)

Test Suite	Test Case	Result	Summary	How to fix	Developer Exec Result
Manager	관리자 프로세스에서 5번 반복하여 입/출금 할 수 있다.	Failed	잔고가 어느 일정 범위 이상 커지면 "잔고 확인하세요"라는 오류 메시지가 출력된다. (잔고 최대값 명세에 존재하지 않음)	**잔고는 지폐별 500매로 최댓값 설정,**	Passed
Manager	관리자 프로세스에서 10번 반복하여 입/출금 할 수 있다.	Failed	잔고가 어느 일정 범위 이상 커지면 "잔고 확인하세요"라는 오류 메시지가 출력된다. (잔고 최대값 명세에 존재하지 않음)	**잔고는 지폐별 500매로 최댓값 설정,**	Passed
User	Jackpot이 터지면 해당 User에게 5만원을 추가 인출하며 ATM의 잔고에서 5만원이 빠져나간다.	Failed	Jackpot이 터져도 ATM의 잔고에는 변화가 생기지 않는다. ATM 잔고가 0일 때도 예외처리 필요	**ATM 잔고가 부족하면 사용자의 계좌로 입력된다는 메시지와 함께 사용자의 계좌로 당첨금을 입금**	Passed
User	1일 1명만 Jackpot이 터진다.	Failed	초기 실행 후 Jackpot이 한 번 터진 후에도 다시 터진다.	Timer 오류로 초기화가 되지 않아 생겼던 문제를 해결	Passed
User	출금 시에만 Jackpot이 터진다.	Failed	송금 시 사용자의 비밀번호를 입력하고 나서도 Jackpot이 터지는 경우 존재	출금 상태에서만 Jackpot 이 터지도록 수정	Passed
User	하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.	Failed	같은 계좌로 출금을 5회 이상 반복 했을 때, 연속으로 2번 잭팟이 터지는 경우가 있다.	하루 한번만 Jackpot 이 터지도록 수정	Passed
User	모든 입력에 대한 기기의 반응은 1초 이내로 이루어진다.	Failed	integer 이내의 정수값이 아니라면, 1초 이내로 반응이 없다(반응이 아예 없다)	예외처리	Passed
User	세 번 이상 연속적으로 입금할 수 있다.	Failed	잔고가 어느 일정 범위 이상 커지면 "비밀번호 확인하세요"라는 오류 메시지가 출력된다.	일반계좌의 잔고를 10억 이내로 고정	Passed
User	세 번 이상 연속적으로 송금할 수 있다.	Failed	3번 이상 송금 하려고 하면 알맞은 비밀번호여도 오류 메세지가 나온다	3번이상 송금이 가능하도록 오류 수정	Passed
User	출금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Failed	출금 시 minus 정수 값을 금액으로 입력하면 해당 정수의 절대값만큼 계좌의 잔액이 증가한다. 예외처리 필요	예외처리	Passed
User	입금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Failed	입금 시 minus 정수 값을 금액으로 입력하면 해당 정수의 절대값만큼 계좌의 잔액이 감소한다. 예외처리 필요	예외처리	Passed
User	송금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Failed	송금 시 minus 정수 값을 금액으로 입력하면 해당 정수의 절대값만큼 계좌의 잔액이 증가한다. 예외처리 필요	예외처리	Passed
Manager	ATM 잔고가 일정수준(50만원) 이하가 되면 잔고부족 메시지를 출력한다.	Failed	ATM 잔고가 50만원 이하가 되어도 아무 메시지도 출력되지 않는다.	**시스템 메인화면(메뉴선택화면) 하단에 1만원과 5만원에 대한 부족 메시지 출력(1천원 5천원은 출금 대상이 아니므로 출력 x)**	Passed

2nd System Testing
Specification Review

- 일관성 없는 명세
  - 특정 문서에서 삭제하거나 삽입해서 일관성을 유지해야 한다.

specification	1000	2030	2040	2050/2060	구현 여부
"Jackpot 기능의 악용가능성 (ex. 한사람이 만원 씩 지속적으로 출금)을 제거" 에 대한 명세	O	X	X	X	X
"GUI_Interface"에 대한 명세 (GUI 관련 항목이 나오는 2050 이전의 문서에서는 존재하지 않아야 함)	O	O	O	O	O

2nd System Testing
Specification Review

- 누락된 명세 내역

누락된 명세	누락된 문서
"수수료" 및 "명세서 출력"에 대한 명세	2030, 2040, 2050/2060
"Jackpot 당첨 후 ATM 잔고에서 5만원 감소시킨다"	2031 문서 중 5.Randomw_Jackpot 항목
"E5: 자기 자신의 계좌에는 송금할 수 없다 "	2041 문서 중 3.Remittance 항목
메소드 start에 대한 명세는 존재하지만 소스 코드 상에서 누락	205 문서의 Timer 클래스에서 1.Timer 항목
MainGUI 라는 표현이 소스코드 상에 존재하지 않음	2052

2nd System Testing
Specification Review

- 반환(return) 자료형 불일치
  - 2050/2060 단계에서의 명세와 소스 코드 상의 자료형 불일치

불일치 하는 항목	관련 문서
ATMBalaceUpdate 메소드 (input_count > inputcount)	2051 문서의 MoneyCount 클래스
SearchID 메소드	2051 문서의 Bank 클래스
getAccountBalance 메소드	2051 문서의 Bank 클래스

2nd System Testing

Specification Review

- 모호한 표현들

모호한 문장	문제점	관련 문서
Jackpot 기능의 악용 가능성 (ex. 한사람이 만원 씩 지속적으로 출금)을 제거	출금 기준 금액 혹은 출금 간격에 대한 언급 필요	2010
당첨되는 순서를 랜덤하게 선택한다.	Random 의 범위에 대한 설명	2031 문서 중 5.Random_Jackpot 항목
inputID와 일치하는 계좌번호를 찾음	클래스의 상세 명세에 관한 항목이므로, 소스 코드 상에서 "계좌번호"가 어떠한 변수를 가리키는지에 대한 언급 필요	2051 문서 중 Account 클래스의 Account.checkID 메소드
inputID와 일치하는 Card의 ID를 찾음	클래스의 상세 명세에 관한 항목이므로, 소스 코드 상에서 "Card의 ID"가 어떠한 변수를 가리키는지에 대한 언급 필요	2051 문서 중 Account 클래스의 SearchCardID 메소드
Account에 저장되어 있는 잔고를 변화시키기 위한 메소드	변화시킨다는 것이 증가인지 감소인지 명확히 표현해야 한다	2051 문서 중 Account 클래스의 balanceAccount 메소드

2nd System Testing
Specification Review

- 오타

오타	관련 문서
Typical Courses of Events 항목에 13번이 2개 존재	2031 문서 중 1.Withdraw 항목
InputMID, InputMPW, ManagerIID을 InputID, InputPW, ManagerID로 수정해야 한다	2051 문서에서 Manager 클래스에서 Manager.checkID와 Manager.checkPW 클래스

2nd System Testing

Specification Review

- System Test Case 항목의 심각한 불일치 (1009, 2038)
  - 구현 단계에서 어느 정도 추가, 누락 혹은 변경이 있을 수도 있지만, 일치하는 항목이 굉장히 적으며, Test Case의 번호나 그룹 번호가 불일치한다.

Test Number	Test 항목	Description	Use Case	System Function
1	출금	유효한 계좌를 입력받으면 이용자가 입력한 금액을 계좌의 잔고와 ATM 잔고를 확인해서 제대로 이용자에게 지급하는지 확인하고, 명세서를 출력하는지 확인	withdraw	
2	입금	유효한 계좌를 입력받으면 이용자가 넣은 금액을 통장 잔고에 제대로 더하고, ATM 잔고를 확인하고 명세서를 출력하는지 확인	deposit	
3	송금	유효한 계좌를 입력받으면 이용자가 입력한 금액을 알맞은 계좌에 입금하는지 확인하고, 이용자의 계좌잔고가 줄어드는지 확인	remittance	

Test Num	Test 항목	Description	Use case
1-1	input_menu Test	사용자가 입력한 해당 메뉴 화면으로 올바르게 진입하는지 확인한다.	R1.2 ,R.2, R.3, R.4, R.5
2-1	input_ID	사용자가 입력한 계좌 혹은 카드번호가 유효하지 않으면 오류메시지를 출력하는지 확인한다.	R1.2, R.2,R.3,R.4
2-2	input_ID	사용자가 입력한 계좌 혹은 카드번호가 유효하면 다음 단계로 넘어가는지 확인한다.	R1.2, R.2,R.3,R.4
3-1	input_PW	사용자가 입력한 비밀번호가 계좌번호의 비밀번호가 아니면 오류메시지를 출력하는지 확인한다	R1.2, R.3, R.4
3-2	input_PW	사용자가 입력한 비밀번호가 계좌번호의 비밀번호가 맞으면 다음단계로 넘어가는지 확인한다.	R1.2, R.3, R.4
4-1	input_amount	출금의 경우 사용자가 입력한 금액이 계좌 잔고보다 많으면 오류메시지를 출력하는지 확인한다.	R1.2, R.2, R.3
4-2	input_amount	출금의 경우 사용자가 입력한 금액이 ATM 잔고보다 많으면 오류메시지를 출력하는지 확인한다.	R1.2, R.2, R.3
4-3	input_amount	송금의 경우 사용자가 입력한 금액이 계좌 잔고보다 많으면 오류메시지를 출력하는지 확인한다.	R1.2, R.2, R.3

2nd System Testing

Category-Partition & Pairwise Testing

- 문서 변경으로 인해 달라진 Expected Results 를 적절히 수정 한 후에 테스트 하였다.
- Category-Partition Testing: 64개 중 64개 Pass (100% Pass)
- Pairwise Testing: 26개 중 26개 Pass (100% Pass)

Test Suite	Test Case	Latest Exec Result
User	atm-5:Test Case 1	Passed
User	atm-6:Test Case 2	Passed
User	atm-7:Test Case 3	Passed
User	atm-8:Test Case 4	Passed
User	atm-9:Test Case 5	Passed
User	atm-10:Test Case 6	Passed
User	atm-11:Test Case 7	Passed
User	atm-12:Test Case 8	Passed
User	atm-13:Test Case 9	Passed
User	atm-14:Test Case 10	Passed
User	atm-15:Test Case 11	Passed
User	atm-16:Test Case 12	Passed
User	atm-17:Test Case 13	Passed
User	atm-18:Test Case 14	Passed
User	atm-19:Test Case 15	Passed
User	atm-20:Test Case 16	Passed
User	atm-21:Test Case 17	Passed
User	atm-22:Test Case 18	Passed
User	atm-23:Test Case 19	Passed
User	atm-24:Test Case 20	Passed

Test Suite	Test Case	Latest Exec Result
User	atm-73:[Pairwise]Test Case1	Passed
User	atm-74:[Pairwise]Test Case2	Passed
User	atm-75:[Pairwise]Test Case3	Passed
User	atm-76:[Pairwise]Test Case4	Passed
User	atm-77:[Pairwise]Test Case5	Passed
User	atm-78:[Pairwise]Test Case6	Passed
User	atm-79:[Pairwise]Test Case7	Passed
User	atm-80:[Pairwise]Test Case8	Passed
User	atm-81:[Pairwise]Test Case9	Passed
User	atm-82:[Pairwise]Test Case10	Passed
User	atm-83:[Pairwise]Test Case11	Passed
User	atm-84:[Pairwise]Test Case12	Passed
User	atm-85:[Pairwise]Test Case13	Passed
User	atm-86:[Pairwise]Test Case14	Passed
Manager	atm-87:[Pairwise]Test Case1	Passed
Manager	atm-88:[Pairwise]Test Case2	Passed
Manager	atm-89:[Pairwise]Test Case3	Passed
Manager	atm-90:[Pairwise]Test Case4	Passed
Manager	atm-91:[Pairwise]Test Case5	Passed
Manager	atm-92:[Pairwise]Test Case6	Passed
Manager	atm-93:[Pairwise]Test Case7	Passed
Manager	atm-94:[Pairwise]Test Case8	Passed
Manager	atm-95:[Pairwise]Test Case9	Passed
Manager	atm-96:[Pairwise]Test Case10	Passed
Manager	atm-97:[Pairwise]Test Case11	Passed
Manager	atm-98:[Pairwise]Test Case12	Passed

2nd System Testing

Brute Force Testing 1

- 문서 변경에 맞추어 6개의 Test Case를 추가적으로 생성하여 테스트 하였다.
- 22/25 (88% Pass)

Test Suite	Test Case	Result
Manager	관리자 프로세스에서 5번 반복하여 입/출금 할 수 있다.	Passed
Manager	관리자 프로세스에서 10번 반복하여 입/출금 할 수 있다.	Passed
User	프로그램 초기 실행 시 난수가 제대로 생성된다.	Passed
User	출금 10번 이내에 Jackpot이 터지는 지 확인한다.	Passed
User	Jackpot이 터지면 해당 User에게 5만원을 추가 인출하며 ATM의 잔고에서 5만원이 빠져나간다.	Passed
User	1일 1명만 Jackpot이 터진다.	Failed
User	출금 시에만 Jackpot이 터진다.	Passed
User	하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.	Failed
User	모든 입력에 대한 기기의 반응은 1초 이내로 이루어진다.	Passed
User	거래는 1분 이내에 이루어져야 한다.	Passed
User	사용자가 보기 편한 화면을 제공해야 한다.	Passed
User	세 번 이상 연속적으로 출금할 수 있다.	Passed
User	세 번 이상 연속적으로 입금할 수 있다.	Passed
User	세 번 이상 연속적으로 송금할 수 있다.	Passed
User	세 번 이상 연속적으로 조회할 수 있다.	Passed
User	출금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	입금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	송금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
Manager	ATM 잔고가 일정수준(50만원) 이하가 되면 잔고부족 메시지를 출력한다.	Passed
User	출금 시 수수료 1200원을 더하여 출금한다.	Passed
User	송금 시 수수료 1200원을 더하여 출금한다.	Passed
User	입금 시 ATM 잔고 지폐별 500개를 초과하면 오류 메시지를 출력한다.	Passed
User	입금 시 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.	Passed
User	송금 시 상대방 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.	Failed
User	Jackpot이 터질 때 ATM 잔고가 부족하면 사용자의 계좌로 5만원을 입금하고 알림 메시지를 출력한다.	Passed

Test Suite	Test Case
User	초기 실행 후 Jackpot이 한 번 터진 후에도 다시 터진다.
User	같은 계좌로 출금을 5회 이상 반복 했을 때, 연속으로 2번 Jackpot이 터지는 경우가 있다
User	송금 시 상대방 계좌 잔고 10억원을 초과하면 잘못된 오류 메시지가 출력된다.

2nd System Testing

Brute Force Testing 2

- 이전에 생각치 못했던 2개의 Test Case 를 추가적으로 생성
- SM 팀의 의견을 반영하여 소스 코드를 적절히 수정하여 다시 한 번 테스트
- 23/27 (85% Pass)

```
public Bank(int code) {
    BankCode=code;
    switch (code) {
    case 1:
                        //3245(2)
        account.add(new Account(325,6124,"감귤",10000090, code));
        account.add(new Account(3245,612,"감",70000, code));
        account.add(new Account(222,612,"test1",70000, code));
        break;
    case 2:
                        //222(2)
        account.add(new Account(6324,1111,"오렌지",100090, code));
        account.add(new Account(333,612,"test2",70000, code));
        account.add(new Account(444,612,"test3",70000, code));
        break;
    case 3:
        account.add(new Account(6114,1111,"오지",10090, code));
        account.add(new Account(555,612,"test4",70000, code));
        account.add(new Account(666,612,"test5",70000, code));
        account.add(new Account(777,612,"test6",70000, code));
        break;
    default:
        break;
    }
}
```

```
public void run() {
    String data = "";
    super.run();
    while(true) {
        try {
            PrintWriter fw = new PrintWriter("date.txt");
            data=String.valueOf(cal.get(Calendar.YEAR));
            fw.println(data);
            data=String.valueOf(cal.get(Calendar.MONTH));
            fw.println(data);

            data=String.valueOf(cal.get(Calendar.DATE));
            System.out.println("data: " + data);

            fw.println(data);
            data=String.valueOf(cal.get(Calendar.HOUR_OF_DAY));
            fw.println(data);
            data=String.valueOf(cal.get(Calendar.MINUTE));
            fw.println(data);
            data=String.valueOf(cal.get(Calendar.SECOND));
            fw.println(data);
            fw.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            sleep(200);

        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        cal.set(Calendar.MINUTE, cal.get(Calendar.MINUTE)+1);

        if(checktime()) {
            MainFrame.getSystem().set_usercount(1);
            updateJackpotNum();
        }
    }
```

2nd System Testing

Brute Force Testing 2

- 이전에 생각치 못했던 2개의 Test Case 를 추가적으로 생성
- SM 팀의 의견을 반영하여 소스 코드를 적절히 수정하여 다시 한 번 테스트
- 23/27 (85% Pass)

Test Suite	Test Case	Result
Manager	관리자 프로세스에서 5번 반복하여 입/출금 할 수 있다.	Passed
Manager	관리자 프로세스에서 10번 반복하여 입/출금 할 수 있다.	Passed
User	프로그램 초기 실행 시 난수가 제대로 생성된다.	Passed
User	출금 10번 이내에 Jackpot이 터지는 지 확인한다.	Passed
User	Jackpot이 터지면 해당 User에게 5만원을 추가 인출하며 ATM의 잔고에서 5만원이 빠져나간다.	Passed
User	1일 1명만 Jackpot이 터진다.	Passed
User	출금 시에만 Jackpot이 터진다.	Passed
User	하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.	Failed
User	모든 입력에 대한 기기의 반응은 1초 이내로 이루어진다.	Passed
User	거래는 1분 이내에 이루어져야 한다.	Passed
User	사용자가 보기 편한 화면을 제공해야 한다.	Passed
User	세 번 이상 연속적으로 출금할 수 있다.	Passed
User	세 번 이상 연속적으로 입금할 수 있다.	Passed
User	세 번 이상 연속적으로 송금할 수 있다.	Passed
User	세 번 이상 연속적으로 조회할 수 있다.	Passed
User	출금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	입금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	송금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
Manager	ATM 잔고가 일정수준(50만원) 이하가 되면 잔고부족 메시지를 출력한다.	Passed
User	출금 시 수수료 1200원을 더하여 출금한다.	Passed
User	송금 시 수수료 1200원을 더하여 출금한다.	Passed
User	입금 시 ATM 잔고 지폐 별 500개를 초과하면 오류 메시지를 출력한다.	Passed
User	입금 시 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.	Passed
User	송금 시 상대방 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.	Failed
User	Jackpot이 터질 때 ATM 잔고가 부족하면 사용자의 계좌로 5만원을 입금하고 알림 메시지를 출력한다.	Passed
User	복수의 화폐에서 최대치(500매)를 넘어선 매수를 입금하려고 하면, 초과된 모든 지폐에 대해 오류 메시지를 출력한다.	Failed
Manager	복수의 화폐에서 최대치(500매)를 넘어선 매수를 입금하려고 하면, 초과된 모든 지폐에 대해 오류 메시지를 출력한다.	Failed

Test Suite	Test Case
User	여러 화폐의 최대 수량(500)을 동시에 넘었을 때, 복수의 오류 메시지를 출력하지 않는다.
User	같은 계좌로 출금을 여러 번 반복했을 때, 다시 Jackpot이 터진다.
User	송금 시 상대방 계좌 잔고 10억원을 초과하면 잘못된 오류 메시지가 출력된다.

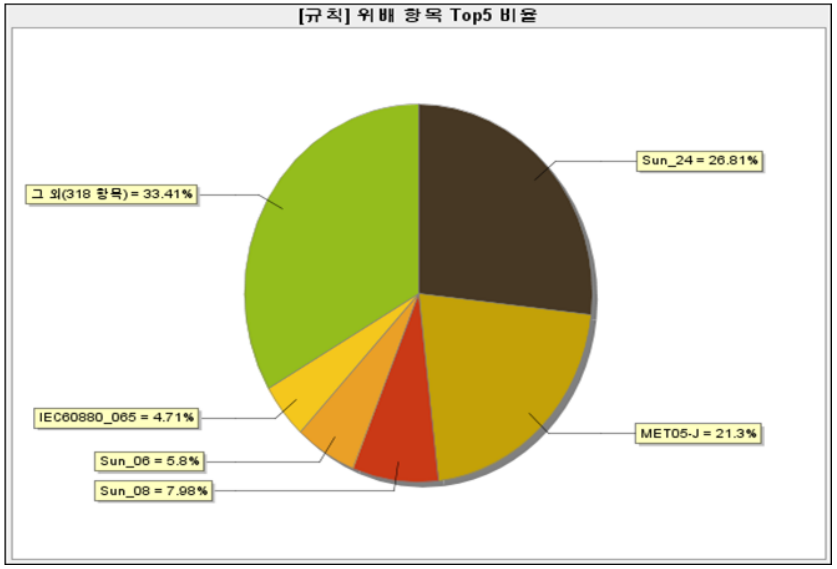
Static Analysis

1. Code Inspector
2. SonarQube
3. Findbugs
4. PMD

Static Analysis

Code Inspector

- 5개의 규칙 모음(총 225개의 규칙)들을 적용하였다.
- 54개의 위배된 규칙 중 가장 문제가 되는 규칙 12개를 선정하였다.



Sun_24: 26%

MET05-J: 21%

Sun_08: 7%

규칙	규칙 모음	위배 수	무시된 위배 수	규칙 설명
Sun_24	Sun_Code_Conventions_for_Java	467	0	숫자 상수 사용 금지
MET05-J	CERT_Secure_Coding_Standard	371	0	생성자에서 override 가능한 메서드를 호출하면 안 된다
Sun_08	Sun_Code_Conventions_for_Java	139	0	블록의 시작 부분에서만 선언 검사
Sun_06	Sun_Code_Conventions_for_Java	101	0	소스 라인 길이 검사
IEC60880_065	IEC_60880	82	0	if 가 있다면 else가 반드시 있는 지 검사

Static Analysis

Code Inspector

- 변수를 선언할 때는 소문자로 시작해야 한다.

```
121 :     private String Name;
```
[EGOV_26_VariableNamingConventions] 필드의 이름(Name)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

```
132 :     public Set<Card> Have;
```
[EGOV_26_VariableNamingConventions] 필드의 이름(Have)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

```
126 :     private int Balance;
```
[EGOV_26_VariableNamingConventions] 필드의 이름(Balance)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

- 개발 완료 후에는 콘솔 창 출력을 지워야 한다.

```
150 :             System.out.println(UserCount);
```
[EGOV_25_SystemPrintln] 메서드 (println)가 사용됨.

[ERR02-J] 적절한 로그를 사용하지 않음.

```
151 :             System.out.println(JackpotNum);
```
[EGOV_25_SystemPrintln] 메서드 (println)가 사용됨.

Static Analysis

Code Inspector

- 디버깅 관련 정보의 비공개를 위하여 예외 상황 발생 시 스택 정보를 출력하지 말아야 한다.

```
16 :            } catch (Exception e) {
```
[ERR08-J] (Exception)을 catch함.
```
17 :                    e.printStackTrace();
```
[JAVA_44] 메서드 (printStackTrace)가 사용됨.

```
52 :                } catch (IOException e) {
53 :                    // TODO Auto-generated catch block
54 :                    e.printStackTrace();
```
[JAVA_44] 메서드 (printStackTrace)가 사용됨.

- 이미 선언된 변수를 다시 선언 하지 말아야 한다.

```
6 :     private static MainFrame frame;
7 :     public static void main(String[] args) {
```
[JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음
```
8 :             // TODO Auto-generated method stub
9 :
10 :            try {
11 :                    Main_controller main = new Main_controller();
12 :                    MainFrame frame = main.getMainFrame();
```
[Sun_09] identifier(frame)가 이미 필드로 선언되어 있어 hide가 발생함

[Sun_23] 클래스 object를 이용하여 static 멤버에접근.

[JAVA_70] identifier(frame)가 이미 필드로 선언되어 있어 hide가 발생함

Static Analysis

Code Inspector

- 의미 없는(공란의) 주석 과다. (26줄의 코드 중 괄호만 있는 줄 포함 8줄만 유효한 코드)

```
8 : public class Manager {
  [JAVA_71] 클래스의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음
  [OBJ04-J] mutable 클래스가 신뢰할 수 없는 코드에 객체를 안전하게 전달하기 위한 복사 기능을 제공하지 않음.
9 :
10 :     /**
11 :      * Default constructor
12 :      */
13 :     public Manager(int id, int pw) {
  [JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음
14 :             ManagerID=id;
15 :             ManagerPW=pw;
16 :     }
17 :
18 :     /**
19 :      *
20 :      */
21 :     private int ManagerID;
  [EGOV_26_VariableNamingConventions] 필드의 이름(ManagerID)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.
  [Sun_20] 필드의 이름(ManagerID)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.
  [Sun_05] 필드의 선언문이 메서드 선언문보다 코드상 뒤 쪽에 존재함
22 :
23 :     /**
24 :      *
25 :      */
26 :     private int ManagerPW;
  [EGOV_26_VariableNamingConventions] 필드의 이름(ManagerPW)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.
  [Sun_20] 필드의 이름(ManagerPW)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.
  [Sun_05] 필드의 선언문이 메서드 선언문보다 코드상 뒤 쪽에 존재함
27 :
28 :
29 :     /**
30 :      * @param inputID
31 :      * @return
32 :      */
33 :     public Boolean checkID(int inputID) {
  [JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음
```

Static Analysis

Code Inspector

- Public 메소드에서 private 변수를 그대로 반환

```
32 :     public MoneyCount getATMcount() {
   [JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음
33 :         return moneycount;
   [OBJ05-J] public 메서드에서 private 필드 리턴 시 레퍼런스를 그대로 리턴함.
34 :     }
```

- non-static 필드가 public으로 선언되어 있고, 메소드의 정의가 끝난 클래스의 중간지점에서 변수의 선언 (클래스의 처음에 public static 혹은 private, protected 로 선언해야 한다.)

```
132 :     public Set<Card> Have;
   [Sun_22] non-static 필드가 public으로 선언.
```

Static Analysis

Code Inspector

- 데이터를 넣어두고 사용하지 않는 불필요한 변수의 선언 (TotalMoney, inputPW)

```
46 :     private int TotalMoney;
```

[EGOV_26_VariableNamingConventions] 필드의 이름(TotalMoney)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_20] 필드의 이름(TotalMoney)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_05] 필드의 선언문이 메서드 선언문보다 코드상 뒤 쪽에 존재함

[EGOV_37_UnusedPrivateField] private 필드 TotalMoney 을 사용하지 않음

- switch 문은 두 개 이상의 case를 가져야 한다(if 문을 사용해야 함)

```
122         switch (inputMenu) {
123         case 5:
124             if(manager.checkID(input)) {
125                 return true;
126             }
127             return false;
128
129         default:
130             for(int index=0; index<bank.size(); index++) {
131                 int check=bank.get(index).searchID(input,1);
132                 if(check==1) {
133                     currentbank=bank.get(index);
134                     return true;
135                 }
136             }
137             break;
138         }
```

Static Analysis

Code Inspector

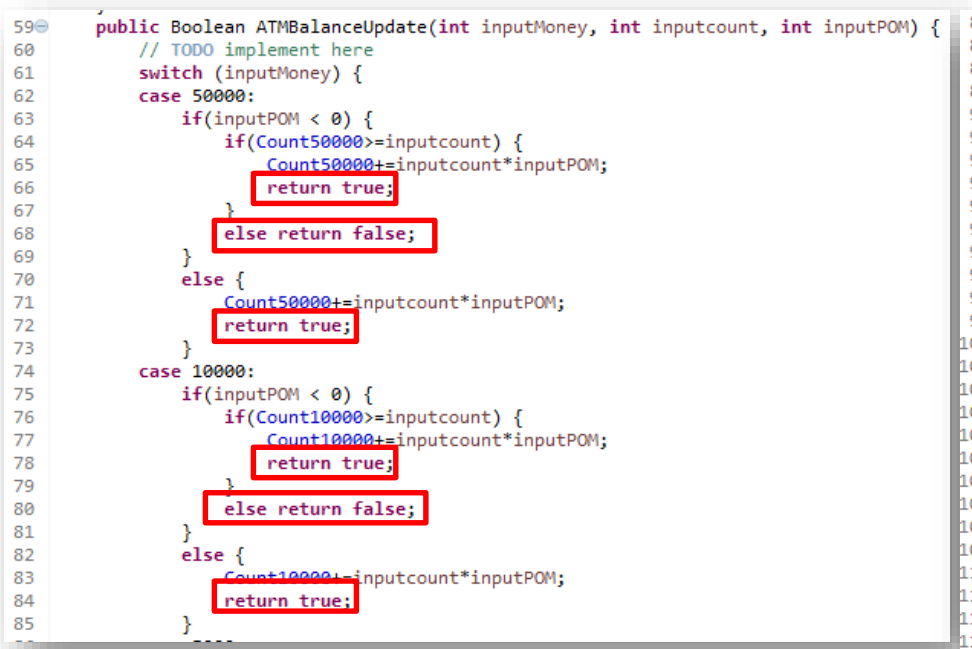
- 생성자 내에서 final이나 private이 아닌 (override 가능한) 메소드를 호출하지 말아야 한다

```
35 :    public MainFrame() {
[JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음
[Sun_05] 생성자의 선언이 메서드 선언보다 코드상 뒤 쪽에 존재함
36 :
37 :            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
[MET05-J] 생성자 내에서 final이나 private이 아닌 메서드를 호출함.
38 :            setBounds(100, 100, 540, 400);
[Sun_24] 숫자 상수를 사용함(for loop의 counter로 -10이나 0, 1을 사용하는 경우 제외)
[MET05-J] 생성자 내에서 final이나 private이 아닌 메서드를 호출함.
39 :            contentPane = new JPanel();
40 :            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
[Sun_24] 숫자 상수를 사용함(for loop의 counter로 -10이나 0, 1을 사용하는 경우 제외)
41 :            setContentPane(contentPane);
[MET05-J] 생성자 내에서 final이나 private이 아닌 메서드를 호출함.
42 :            contentPane.setLayout(null);
[MET05-J] 생성자 내에서 final이나 private이 아닌 메서드를 호출함.
43 :
```

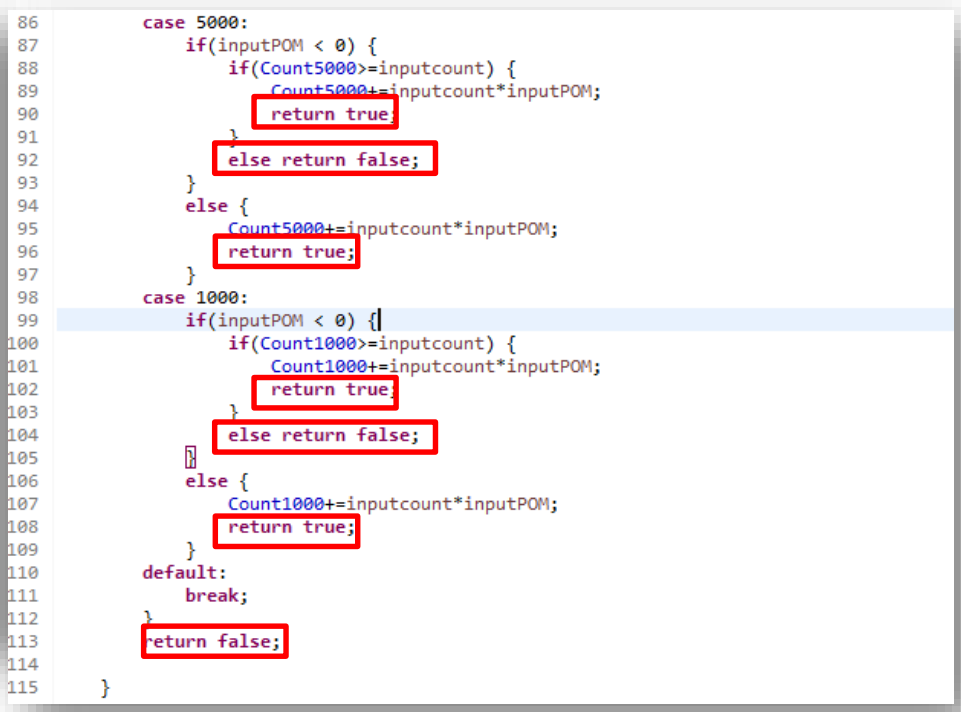
Static Analysis

Code Inspector

- 한 메소드 내에서 과다한 exit point (50라인 정도의 한 함수 내에서 12개의 exit point)



```
59  public Boolean ATMBalanceUpdate(int inputMoney, int inputcount, int inputPOM) {
60      // TODO implement here
61      switch (inputMoney) {
62      case 50000:
63          if(inputPOM < 0) {
64              if(Count50000>=inputcount) {
65                  Count50000+=inputcount*inputPOM;
66                  return true;
67              }
68              else return false;
69          }
70          else {
71              Count50000+=inputcount*inputPOM;
72              return true;
73          }
74      case 10000:
75          if(inputPOM < 0) {
76              if(Count10000>=inputcount) {
77                  Count10000+=inputcount*inputPOM;
78                  return true;
79              }
80              else return false;
81          }
82          else {
83              Count10000+=inputcount*inputPOM;
84              return true;
85          }
```

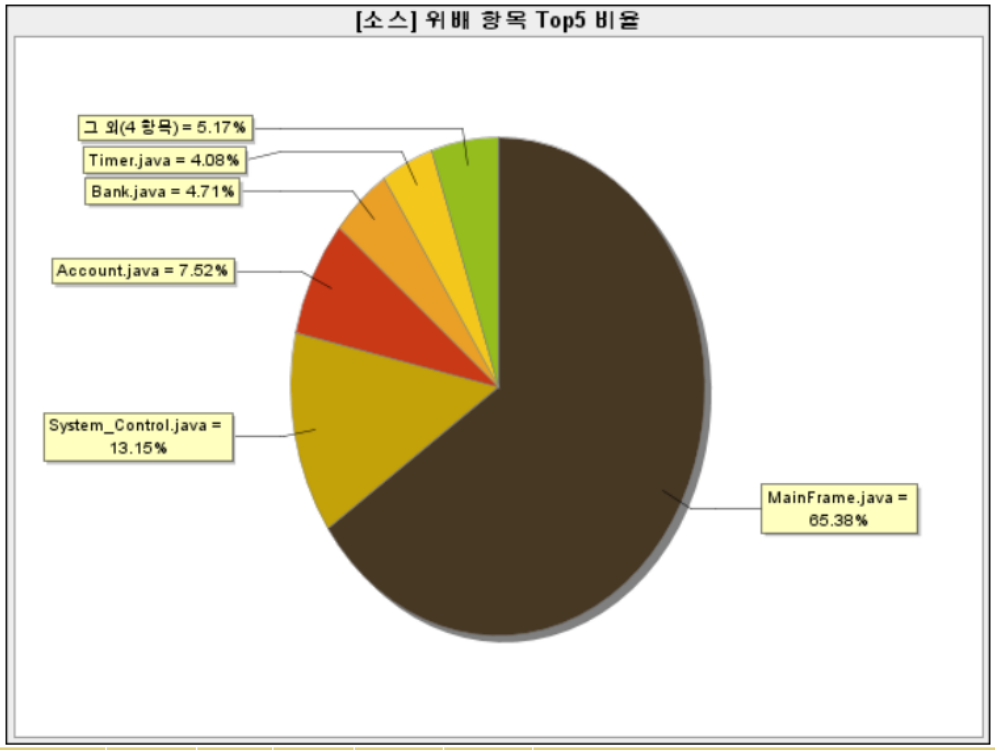


```
86      case 5000:
87          if(inputPOM < 0) {
88              if(Count5000>=inputcount) {
89                  Count5000+=inputcount*inputPOM;
90                  return true;
91              }
92              else return false;
93          }
94          else {
95              Count5000+=inputcount*inputPOM;
96              return true;
97          }
98      case 1000:
99          if(inputPOM < 0) {
100             if(Count1000>=inputcount) {
101                 Count1000+=inputcount*inputPOM;
102                 return true;
103             }
104             else return false;
105         }
106         else {
107             Count1000+=inputcount*inputPOM;
108             return true;
109         }
110     default:
111         break;
112     }
113     return false;
114
115 }
```

Static Analysis

Code Inspector

- 전체 소스 코드 중 65%의 비중을 차지하는 클래스에서 대부분의 위배 발생
- 주로 변수의 적절치 못한 선언과, 메소드 내부에서의 흐름(exit point, switch)

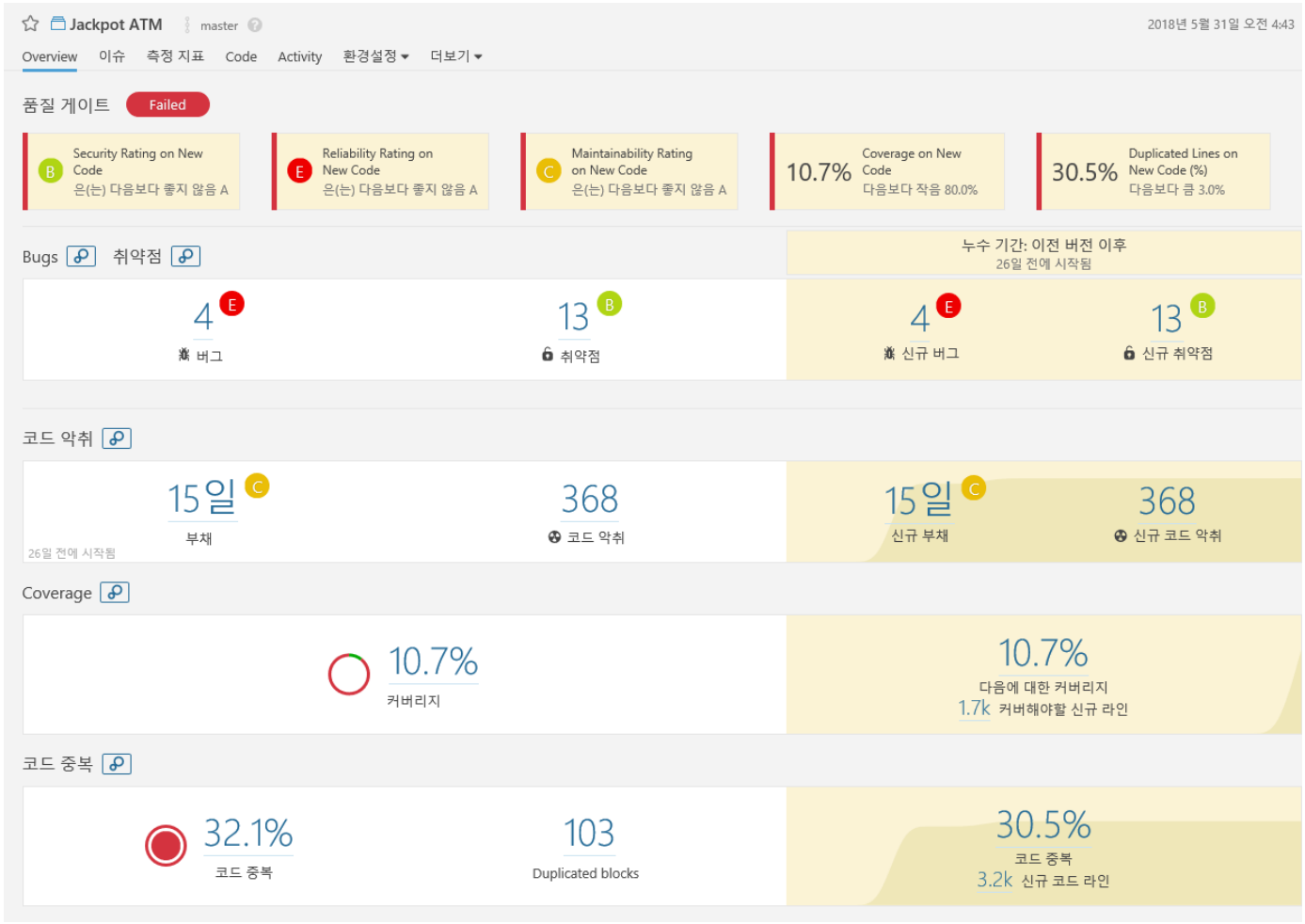


Class	Lines (2318)	위배 (1742)
MainFrame	1.5k (65%)	1139 (65%)
System_ Control	289 (12%)	229 (13%)
Account	153 (7%)	131 (7%)

Static Analysis

SonarQube

- 신뢰성, 보안성, 유지보수성, Coverage, 중복 등을 알린다



Static Analysis

SonarQube

- SM 팀의 JUnit Test 결과 100% success (JUnit report)

Test Summary

18	0	0	0.305s
tests	failures	ignored	duration

100%
successful

Packages **Classes**

Class	Tests	Failures	Ignored	Duration	Success rate
DataBase.BankTest	2	0	0	0.070s	100%
DataBase.ManagerTest	2	0	0	0.011s	100%
System.System_ControlTest	8	0	0	0.171s	100%
User.AccountTest	6	0	0	0.053s	100%

Static Analysis

SonarQube

JACOCO
Java Code Coverage

- Coverage: 10%

	Coverage	Uncovered lines	Uncovered conditions
Test/src/main/java/View/Main_controller.java	0.0%	12	–
Test/src/main/java/View/MainFrame.java	0.0%	1,187	382
Test/src/main/java/System/MoneyCount.java	9.8%	34	21
Test/src/main/java/System/System_Control.java	17.5%	131	90
Test/src/main/java/DataBase/Bank.java	47.8%	33	27
Test/src/main/java/User/Card.java	50.0%	4	3
Test/src/main/java/System/Timer.java	51.6%	37	7
Test/src/main/java/User/Account.java	52.4%	51	9
Test/src/main/java/DataBase/Manager.java	100%	0	0

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
View		0%		0%	273	273	1,199	1,199	82	82	47	47
System		28%		15%	89	105	202	281	18	27	0	3
User		54%		60%	13	26	55	110	4	11	0	2
DataBase		62%		43%	19	33	33	81	3	8	0	2
Total	7,377 of 8,229	10%	539 of 599	10%	394	437	1,489	1,671	107	128	47	54

Static Analysis

Findbugs

- 크게 심각하지 않은 Warning이 34개

FindBugs Result

Warnings Trend

All Warnings	New this build	Fixed Warnings
34	34	0

Summary

Total	High Priority	Normal Priority	Low Priority
34	0	0	34

Details

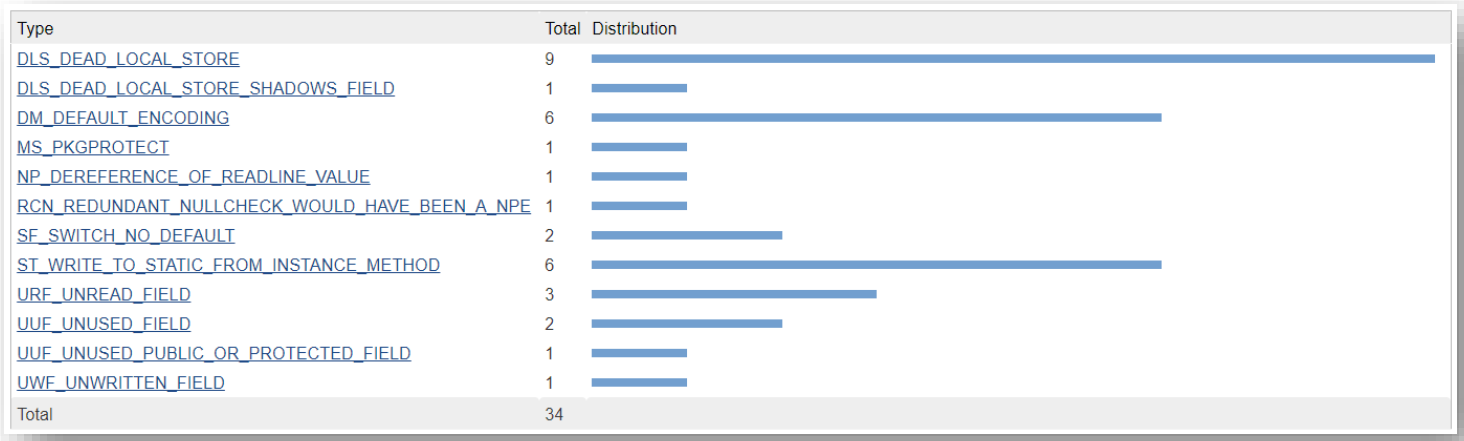
Packages | Files | People | Categories | Types | Warnings | Origin | Details | New

Package	Total	Distribution
DataBase	2	
System	8	
User	7	
View	17	
Total	34	

Static Analysis

Findbugs

Warnings(#)	Description
9	선언 후 사용되지 않는 변수
6	byte로 받아서 String으로 형변환
6	정적 변수를 클래스에서 선언 후, 내부 메소드에서 write

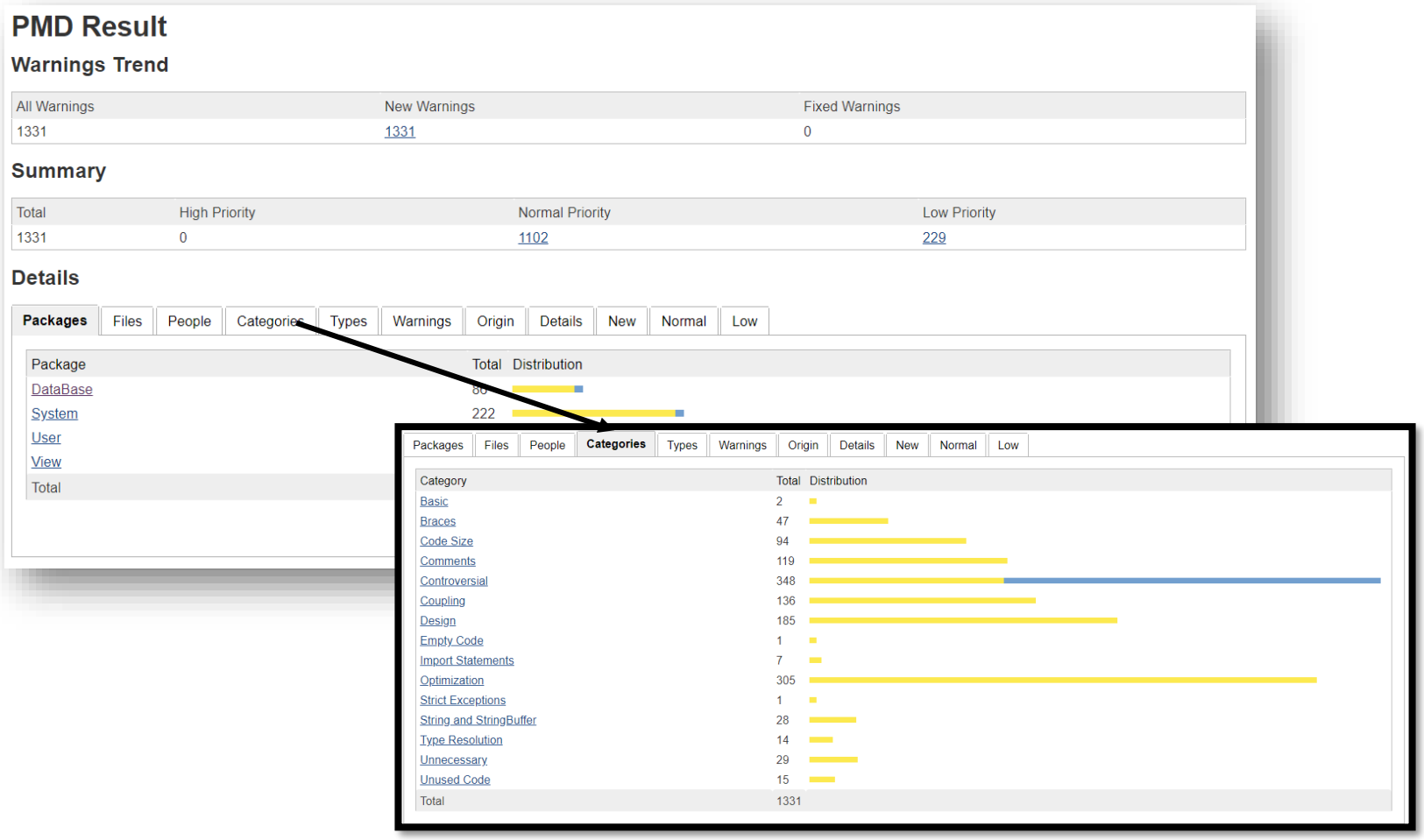


Type	Total	Distribution
DLS_DEAD_LOCAL_STORE	9	
DLS_DEAD_LOCAL_STORE_SHADOWS_FIELD	1	
DM_DEFAULT_ENCODING	6	
MS_PKGPROTECT	1	
NP_DEREFERENCE_OF_READLINE_VALUE	1	
RCN_REDUNDANT_NULLCHECK_WOULD_HAVE_BEEN_A_NPE	1	
SF_SWITCH_NO_DEFAULT	2	
ST_WRITE_TO_STATIC_FROM_INSTANCE_METHOD	6	
URF_UNREAD_FIELD	3	
UUF_UNUSED_FIELD	2	
UUF_UNUSED_PUBLIC_OR_PROTECTED_FIELD	1	
UWF_UNWRITTEN_FIELD	1	
Total	34	

Static Analysis

PMD

- 총 1331개 중 Normal Priority 1102, Low Priority 229



PMD Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
1331	1331	0

Summary

Total	High Priority	Normal Priority	Low Priority
1331	0	1102	229

Details

Packages | Files | People | Categories | Types | Warnings | Origin | Details | New | Normal | Low

Package	Total	Distribution
DataBase	80	
System	222	
User		
View		
Total		

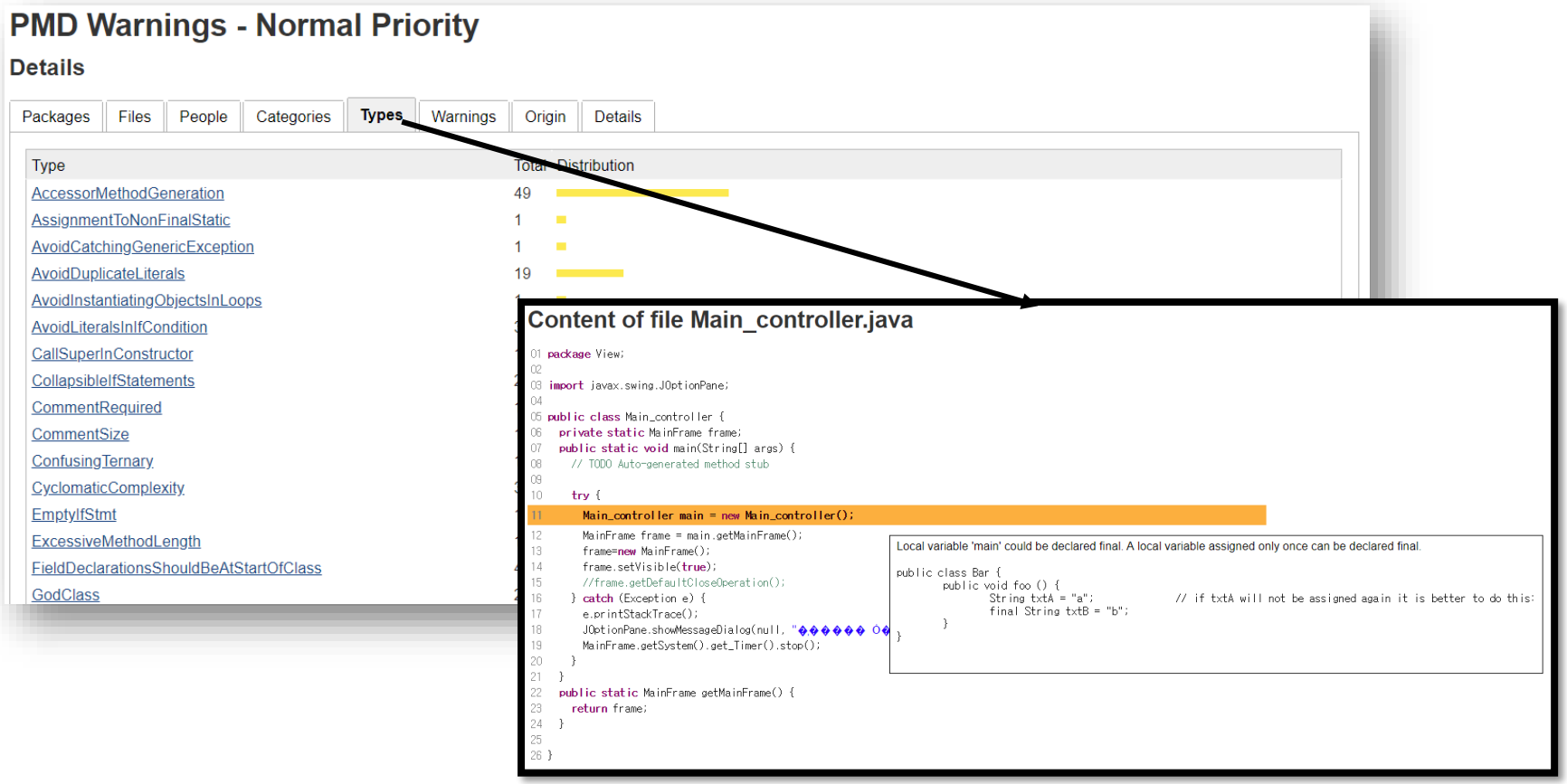
Packages | Files | People | Categories | Types | Warnings | Origin | Details | New | Normal | Low

Category	Total	Distribution
Basic	2	
Braces	47	
Code Size	94	
Comments	119	
Controversial	348	
Coupling	136	
Design	185	
Empty Code	1	
Import Statements	7	
Optimization	305	
Strict Exceptions	1	
String and StringBuffer	28	
Type Resolution	14	
Unnecessary	29	
Unused Code	15	
Total	1331	

Static Analysis

PMD

- Normal Priority 중에서 가장 많이 발생한 Warning: 지역 변수의 선언과 관련
- 이 밖에도 빈번하게 발생한 문제들은 대부분 지역/전역 변수의 사용과 형변환



Static Analysis

Comparison

Software	Result
Code Inspector	변수의 선언, 메소드 내부에서 매끄러운 흐름
FindBugs	변수의 선언, 형변환
PMD	변수의 선언, 주석, exit point
SonarQube	Coverage, 보안성, 코드 중복 등을 한 눈에 볼 수 있다

Static Analysis

Comparison

심각도	Code Inspector	FindBugs	PMD
매우 높음	47 (2%)		
높음	513 (29%)	0 (0%)	0 (0%)
보통		0 (0%)	1102 (82%)
낮음	169 (9%)	34 (100%)	229 (17%)
매우 낮음	1013 (58%)		
총 위배 수	1742	34	1331

- Code Inspector: 상세하고 세분화 되게 이슈들을 하나하나 점검 해 보고 싶을 때
- FindBugs: 빠르게 꼭 필요한 결과 몇 개 만을 보면서 큰 Warning들을 잡아내고 싶을 때.
- PMD: FindBugs에 비해 Warning의 이름이 더욱 직관적이고, 코드의 경향이나 특징을 잡아내고자 할 때 편리하다.

Overall

- 점점 발전해 나가는 Specification과 소프트웨어

1st System Testing
- Specification Review
- Category-partition Testing
- Pairwise Testing
- Brute Force Testing

2nd System Testing
- Specification Review
- Category-partition & Pairwise Testing
- Brute Force Testing

Static Analysis
- Code Inspector
- SonarQube
- FindBugs
- PMD

예외 처리 등 기본 기능에서의 문제

- 추가적인 기능(Jackpot)에서의 문제
- 코딩에서 매끄럽지 못한 메모리 사용

THANK YOU