V&V Second Cycle

Spec Review & System Test & Static Analysis

Team 8

김하림
신윤섭
이승민
정인욱

1. Spec Review

V8C #	Review
297	수정 확인
298	수정 확인
299	변수 이름을 바꾸는 대신 변수 설명을 바꾼 것을 확인
300	수정 확인
301	수정 확인
302	수정 확인
303	수정 확인
304	메시지를 받고 처리 후 삭제하며, DVM 대수의 제약 사항으로 128MB를 넘길 일이 없다고 판단함
305	수정 확인
306	수정 확인
307	수정 확인
308	수정 확인
309	수정 확인
310	수정 확인
311	수정 확인
312	수정 확인
313	수정 확인
314	해당 표현 삭제 확인
315	해당 표현 삭제 확인
316	수정 확인
317	수정 확인
318	수정 확인
319	수정 확인
320	수정 확인
321	수정 확인
322	수정 확인

323	수정 확인
324	수정 확인
325	수정 확인
326	수정 확인
327	수정 확인
328	수정 확인
329	수정 확인
330	수정 확인
331	수정 확인
332	수정 확인
333	수정 확인
334	수정 확인
335	수정 확인
336	수정 확인
337	수정 확인
338	수정 확인
339	수정 확인
340	구현에 맞게 수정 확인
341	수정 확인
342	수정 확인
343	수정 확인
344	수정 확인
345	수정 확인
346	수정 확인
347	수정 확인
348	수정 확인
349	수정 확인
350	수정 확인
351	수정 확인

352	수정 확인
353	수정 확인
354	수정 확인
355	수정 확인
356	수정 확인
357	수정 확인
358	수정 확인
359	수정 확인
360	수정 확인
361	수정 확인
362	수정 확인
363	수정 확인
364	수정 확인
365	수정 확인
366	수정 확인
367	수정 확인
368	수정 확인
369	수정 확인
370	수정 확인
371	개발팀과 논의 후 개정안 삭제 완료 (‘재고 수량 업데이트’ 용어 유지)
372	수정 확인
373	수정 확인
374	수정 확인
375	보충 -> 보충, 삭제로 수정 확인
376	위치정보 요구 메시지로 수정 확인
377	Alternative Courses of Events 수정 확인
378	Pre-Requisites 사용자 삭제 확인
379	1번 단계 삭제 확인
380	22개의 use case 확인

381	Notification 삭제 확인
382	수정 확인
383	목록 -> 명칭으로 수정 확인
384	수정 확인
385	수정 확인
386	수정 확인
387	Domain model에서 확인
388	수정 확인
389	수정 사항 누락 - Stage2030_v4 34p use case 4 name
390	수정 확인
391	재고 수량 업데이트 용어 유지 합의
392	안내 문구 수정 확인
393	id 정보 추가 확인
394	수정 확인
395	수정 확인
396	수정 확인
397	합산 표현 삭제 확인
398	재고 수량 업데이트 용어 유지 합의
399	구체적인 표현으로 수정 확인
400	수정 확인
401	재고 수량 업데이트 용어 유지 합의

105개의 Spec 수정 사항 중 104개 수정 확인

2. System Test

2.1. Brute Force Test

Index	Test	Result
1	홀/짝 월에 따른 일 수	P

2	윤년 2월의 일 수 (2024년 2월)	P
3	유통기한 지난 것의 등록 가능 여부	P
4	결제 방법 선택화면에서 선택한 음료정보가 제대로 출력되는지	P
5	자판기가 없을 때 안내 메시지 출력 후 이전 화면으로 복귀	P
6	선결제 인증코드 6자리 이상 입력 시도	P
7	잘못된 인증코드 입력 시도	P
8	사용한 인증코드 다시 입력 시도	P
9	관리자 코드 입력 시도(관리자 화면으로 잘 넘어가는지)	P
10	3분 이상 입력이 없으면 잠금화면으로 이동하는지 시도	P
11	3분 이상 입력이 없어 초기화가 된 후에 음료 구매가 정상적으로 이루어지는지 시도	P
12	취소 또는 나가기 버튼을 눌러 초기화면으로 돌아갔을 경우 정상적으로 구매가 이루어지는지 시도	P
13	관리자 재고화면에서 음료를 보이는 화면 너머까지 추가할 경우 제일 마지막 음료가 잘 보이는지 테스트	P
14	결제 실패 시 오류 메시지를 출력한 후 메인화면으로 복귀 잘 되는지	P
15	모든 자판기 목록이 뜨는지 확인	P
16	구매 가능한 자판기 목록에 재고가 있는 자판기가 모두 뜨는지 확인	P
17	날짜가 변경되었을 때 유통기한이 지난 재고가 삭제되는지 확인	F

총 17개의 Test Case 중 16개의 Test Case를 통과하여 94.12%이다.

2.2. Category Partitioning Test

Category Partitioning Test의 경우 화면, 입력, 데이터라는 세 가지 그룹으로 나누고 각 그룹별로 카테고리를 나누어서 검증했다. 입력의 경우 사용자의 터치 이벤트와 타 DVM의 메시지라는 두 가지 카테고리가 존재하며 데이터의 경우 다양한 세부적인 카테고리가 존재한다.
첫번째 System Test에서 달라진 점은 3가지 이다. 첫번째, Message카테고리의 Value를 3가지로 줄이고 single 속성을 부여하였다. 두번째, Action Category에 재고 추가,삭제 Value를 추가하였다. 세번째, Data그룹에 선결제 여부 Category를 추가하였다.

Group	Category	Index	Value	Reference
View	View	1	잠금 화면	

		2	음료 선택 화면	
		3	재고 없음 안내 화면	
		4	재고 있는 자판기 목록 화면	
		5	결제 방법 선택 화면	
		6	카드 결제 화면	
		7	간편 결제 화면	
		8	음료 출력 화면	
		9	생성된 인증코드 출력 화면	
		10	인증코드 입력 화면	
		11	재고 목록 화면	
		12	재고 관리 화면	
		13	유통기한 입력 화면	
		14	에러 화면	
Input	Message	15	Msg type 1: 재고 확인 요청	[single]
		16	Msg type 4: 주소 요청	[single]
		17	Msg type 3: 선결제 확인	[single]
	Action	18	CANCEL	[3][4][5][6][7][10][11][12][13]
		19	3분 이상 입력주지 않기	[2][3][4][5][10][11][12][13]
		20	확인 버튼 입력하기	[1][3][10]
		21	음료 선택하기	[2]
		22	인증코드 화면으로 전환하기	[2]
		23	결제 수단 선택	[5]
		24	결제 정보 입력하기	[6][7]
		25	재고 추가	[12]
		26	재고 삭제	[12]
Data	음료의 재고	27	사용자가 선택한 음료의 재고가 0일 때	[2]
		28	사용자 또는 관리자가 선택한 음료의	[2][12]

			재고가 1개 이상일 때	
	인증코드 종류	29	관리자 코드일 때 (111111)	[10]
		30	사용자 코드일 때	[10]
		31	유효하지 않은 코드일 때	[10]
	하나의 DVM에서 재고가 있는 음료 종류의 수	32	0~6개 일 때	[12]
		33	7개 일 때	[12]
		34	8~20개 일 때	[12]
	결제 정보	35	유효한 결제 정보일 때	[6][7]
		36	유효하지 않은 결제 정보일 때	[6][7][14]
	선결제 여부	37	선결제가 아닐 때	[6][7]
		38	선결제일 때	[6][7]

All test cases: 14 × 3 × 9 × 2 × 3 × 3 × 2 × 2= **27,216**

2.3. TSL Generator

데이터는 대체로 화면과 관계되어 있는 경우가 많아 TSL Generator에서 이에 대한 의존성 처리를 하여 가지치기된 경우의 수를 도출하였다. Category Partitioning Test로부터 총 **27,216** 개의 Test Case를 획득했고, TSL Generator에서 의존성을 처리하기 위해 제약을 걸어 Test Case의 수를 **59(0.22%)**개로 감소시켰다.

2.3.1. Input: DVM

```
Environments:
        Stub configuration:
                multiple stubs.

Parameters:
        View:
                잠금화면.                       [property P1]
                음료 선택 화면.                 [property P2]
                재고 없음 안내 화면.            [property P3]
```

재고 있는 자판기 목록 화면. [property P4]
결제 방법 선택 화면. [property P5]
카드 결제 화면. [property P6]
간편 결제 화면. [property P7]
음료 출력 화면. [property P8]
생성된 인증코드 출력 화면. [property P9]
인증코드 입력 화면. [property P10]
재고 목록 화면. [property P11]
재고 관리 화면. [property P12]
유통기한 입력 화면. [property P13]
에러 화면. [property P14]

Message:
Msg type 1 재고 확인 요청. [single]
Msg type 4 주소 요청.[single]
Msg type 3 선결제 확인. [single]

Action:
CANCEL. [if P3 || P4 || P5 || P6 || P7 || P10 || P11 || P12 || P13]
3분 이상 입력 주지 않기. [if P2 || P3 || P4 || P5 || P10 || P11 || P12 || P13]
확인 버튼 입력하기. [if P1 || P3]
음료 선택하기. [if P2]
인증코드 화면으로 전환하기. [if P2]
결제 수단 선택. [if P5]
결제 정보 입력하기. [if P6 || P7]
재고 추가. [if P12]
재고 삭제. [if P12]

음료의 재고:
사용자가 선택한 음료의 재고가 0일 때. [if P2]
사용자 또는 관리자가 선택한 음료의 재고가 1개 이상일 때. [if P2 || P12]

인증코드 종류:
관리자 코드일 때. [if P10]
사용자 코드일 때. [if P10]
유효하지 않은 코드일 때. [if P10]

하나의 DVM에서 재고가 있는 음료 종류의 수:
0~6개 일 때. [if P12]
7개 일 때. [if P12]
8~20개 일 때. [if P12]

결제 정보:
유효한 결제 정보일 때. [if P6 || P7]
유효하지 않은 결제 정보일 때. [if P6 || P7 || P14]

선결제 여부:
선결제가 아닐 때. [if P6 || P7]
선결제일 때. [if P6 || P7]

2.3.2. Output: DVM.tsl

```
Test Case 1             <single>
  Message :  Msg type 1 재고 확인 요청


Test Case 2             <single>
  Message :  Msg type 4 주소 요청


Test Case 3             <single>
  Message :  Msg type 6 음료 판매 확인


Test Case 4             (Key = 1.1.0.3.0.0.0.0.0.)
  Stub configuration                    :  multiple stubs
  View                                  :  잠금화면
  Message                               :  <n/a>
  Action                                :  확인 버튼 입력하기
  음료의 재고                           :  <n/a>
  인증코드 종류                         :  <n/a>
  하나의 DVM에서 재고가 있는 음료 종류의 수 :  <n/a>
  결제 정보                             :  <n/a>
  선결제 여부                           :  <n/a>


// 중략


Test Case 59            (Key = 1.14.0.0.0.0.0.1.0.)
  Stub configuration                    :  multiple stubs
  View                                  :  에러 화면
  Message                               :  <n/a>
  Action                                :  <n/a>
  음료의 재고                           :  <n/a>
  인증코드 종류                         :  <n/a>
  하나의 DVM에서 재고가 있는 음료 종류의 수 :  <n/a>
  결제 정보                             :  유효하지 않은 결제 정보일 때
  선결제 여부                           :  <n/a>
```

2.3.3. Test Result

TC #	Key	Input	Expected Result	Result
1	<single>	A 자판기가 B 자판기로부터 재고	B 자판기가 A 자판기로부터 재고 응답	P

		확인 요청 메시지를 수신함.	메시지를 수신하여 재고가 있다면 자판기 목록에 A자판기를 포함하여 출력되어야 함.	
2	<single>	A 자판기가 B 자판기로부터 주소 요청 메시지를 수신함.	B 자판기가 A 자판기로부터 주소 응답 메시지를 수신하여 재고가 있다면 자판기 목록에 A자판기를 포함하여 출력되어야 함.	P
3	<single>	A 자판기가 B 자판기로부터 선결제 확인 메시지를 수신함.	B 자판기에서 출력된 인증번호를 A 자판기 인증번호 입력 화면에서 입력 시 B 자판기에서 선택한 음료가 출력 되어야 함.	P
4	1.1.0.3.0.0.0.0.0.	잠금화면에서 화면에 아무 입력(클릭)을 줌.	음료 선택 화면으로 전환되어야 함.	P
5	1.2.0.2.1.0.0.0.0.	음료 선택 화면에서 3분간 입력을 주지 않음.	잠금 화면으로 전환되어야 함.	P
6	1.2.0.2.2.0.0.0.0.	음료 선택 화면에서 3분간 입력을 주지 않음.	잠금 화면으로 전환되어야 함.	P
7	1.2.0.4.1.0.0.0.0.	음료 선택 화면에서 DVM 내 재고가 1개 이상인 음료를 선택함.	결제 수단 입력 화면으로 전환되어야 함.	P
8	1.2.0.4.2.0.0.0.0.	음료 선택 화면에서 DVM 내 재고가 0개인 음료를 선택함.	재고 없음 안내 화면으로 전환되어야 함.	P
9	1.2.0.5.1.0.0.0.0.	음료 선택 화면에서 ‘선결제 인증번호’ 버튼을 선택함.	인증번호 입력 화면으로 전환되어야 함.	P
10	1.2.0.5.2.0.0.0.0.	음료 선택 화면에서 ‘선결제 인증번호’ 버튼을 선택함.	인증번호 입력 화면으로 전환되어야 함.	P
11	1.3.0.1.0.0.0.0.0.	재고 없음 안내 화면에서 취소 버튼을 선택함.	음료 선택 화면으로 전환되어야 함.	P

12	1.3.0.2.0.0.0.0.0.	재고 입력 화면에서 3분간 입력을 주지 않음.	잠금 화면으로 전환되어야 함.	P
13	1.3.0.3.0.0.0.0.0.	재고 없음 안내 화면에서 '구매 가능한 자판기 안내' 버튼을 선택함.	재고 있는 자판기 목록 화면으로 전환되어야 함.	P
14	1.4.0.1.0.0.0.0.0.	재고 있는 자판기 목록 화면에서 취소 버튼을 선택함.	음료 선택 화면으로 전환되어야 함.	P
15	1.4.0.2.0.0.0.0.0.	재고 있는 자판기 목록 화면에서 3분 이상 입력을 주지 않음.	잠금 화면으로 전환되어야 함.	P
16	1.5.0.1.0.0.0.0.0.	결제 방법 선택 화면에서 취소 버튼을 누름	음료 선택 화면으로 전환되어야 함	P
17	1.5.0.2.0.0.0.0.0.	결제 방법 선택 화면에서 3분 이상 입력을 주지 않음	음료 선택 화면으로 전환되어야 함	P
18	1.5.0.6.0.0.0.0.0.	1. 결제 방법 선택 화면에서 카드 결제를 선택함 2. 결제 방법 선택 화면에서 간편 결제를 선택함	1. 카드 결제 화면으로 전환되어야 함 2. 간편 결제 화면으로 전환되어야 함	P
19	1.6.0.1.0.0.0.1.1.	카드 결제 화면에서 유효한 결제 정보이며 선결제가 아닐 때 취소 버튼을 선택함	음료 선택 화면으로 전환되어야 함	P
20	1.6.0.1.0.0.0.1.2.	카드 결제 화면에서 유효한 결제 정보이며 선결제일 때 취소 버튼을 선택함	음료 선택 화면으로 전환되어야 함	P
21	1.6.0.1.0.0.0.2.1.	카드 결제 화면에서 유효하지 않 결제 정보이며 선결제가 아닐 때 취소 버튼을 선택함	음료 선택 화면으로 전환되어야 함	P
22	1.6.0.1.0.0.0.2.2.	카드 결제 화면에서 유효하지 않은 결제	음료 선택 화면으로 전환되어야 함	P

		정보이며 선결제일 때 취소 버튼을 선택함		
23	1.6.0.7.0.0.0.1.1.	카드 결제 화면에서 유효한 결제 정보이며 선결제가 아닐 때 결제 정보를 입력함	음료 출력 화면으로 전환되어야 함	P
24	1.6.0.7.0.0.0.1.2.	카드 결제 화면에서 유효한 결제 정보이며 선결제일 때 결제 정보를 입력함	음료 출력 화면으로 전환되어야 함	P
25	1.6.0.7.0.0.0.2.1.	카드 결제 화면에서 유효하지 않은 결제 정보이며 선결제가 아닐 때 결제 정보를 입력함	에러 화면으로 전환 후 결제 실패 문구를 출력해야 함	P
26	1.6.0.7.0.0.0.2.2.	카드 결제 화면에서 유효하지 않은 결제 정보이며 선결제일 때 결제 정보를 입력함	에러 화면으로 전환 후 결제 실패 문구를 출력해야 함	P
27	1.7.0.1.0.0.0.1.1.	간편 결제 화면에서 유효한 결제 정보이며 선결제가 아닐 때 취소 버튼을 입력함	음료 선택 화면으로 전환해야 함	P
28	1.7.0.1.0.0.0.1.2.	간편 결제 화면에서 유효한 결제 정보이며 선결제일 때 취소 버튼을 입력함	음료 선택 화면으로 전환해야 함	P
29	1.7.0.1.0.0.0.2.1.	간편 결제 화면에서 유효하지 않은 결제 정보이며 선결제가 아닐 때 취소 버튼을 입력함	음료 선택 화면으로 전환해야 함	P
30	1.7.0.1.0.0.0.2.2.	간편 결제 화면에서 유효하지 않은 결제 정보이며 선결제일 때 취소 버튼을 입력함	음료 선택 화면으로 전환해야 함	P
31	1.7.0.7.0.0.0.1.1.	유효한 결제 정보이며 선결제가 아닐 때 결제 정보를 입력함	결제가 이루어지고 음료 출력창으로 이동해야 함	P

32	1.7.0.7.0.0.0.1.2.	유효한 결제 정보이며 선결제일 때 결제 정보를 입력함	결제가 이루어지고 해당 음료에 대한 인증번호가 출력되어야 함	P
33	1.7.0.7.0.0.0.2.1.	유효한 결제 정보가 아니며 선결제가 아닐 때 결제 정보를 입력함	간편 결제 실패 화면으로 전환되어야 함	P
34	1.7.0.7.0.0.0.2.2.	유효한 결제 정보가 아니며 선결제일 때 결제 정보를 입력함	간편 결제 실패 화면으로 전환되어야 함	P
35	1.8.0.0.0.0.0.0.0.	음료 결제 성공 / 선결제 인증번호 입력	결제한 음료 출력 화면이 10초간 출력되어야 함	P
36	1.9.0.0.0.0.0.0.0.	선결제(카드/간편) 성공	선결제한 음료에 대해 생성된 인증번호가 10초동안 출력되어야 함	P
37	1.10.0.1.0.1.0.0.0.	인증코드 입력화면에서 관리자 코드 입력 후 Cancel버튼을 클릭한다.	메인화면으로 전환되야 함	P
38	1.10.0.1.0.2.0.0.0.	인증코드 입력화면에서 사용자 코드 입력 후 Cancel버튼을 클릭한다.	메인화면으로 전환되야 함	P
39	1.10.0.1.0.3.0.0.0.	인증코드 입력화면에서 유효하지 않은 코드 입력 후 Cancel버튼을 클릭한다.	메인화면으로 전환되야 함	P
40	1.10.0.2.0.1.0.0.0.	인증코드 입력화면에서 관리자 코드 입력 후 3분 이상 입력을 주지 않는다.	대기화면으로 전환되야 함	P
41	1.10.0.2.0.2.0.0.0.	인증코드 입력화면에서 사용자 코드 입력 후 3분 이상 입력을 주지 않는다.	대기화면으로 전환되야 함	P
42	1.10.0.2.0.3.0.0.0.	인증코드 입력화면에서 유효하지 않은 코드 입력 후 3분 이상 입력을 주지 않는다.	대기화면으로 전환되야 함	P
43	1.11.0.1.0.0.0.0.0.	재고목록 화면에서 cancel을 눌렀을 경우	메인화면으로 전환되야 함	P

44	1.11.0.2.0.0.0.0.0.	재고 목록 화면에서 3분 이상 대기할 경우	대기화면으로 전환되야 함	P
45	1.12.0.1.2.0.1.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 0~6개 일때 CANCEL 버튼을 누른다.	Real Use Cases의 Exceptional Courses of Events기준 : 관리자 메뉴 첫 화면으로 전환되야 함	P
46	1.12.0.1.2.0.2.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 7개 일때 CANCEL 버튼을 누른다.	관리자 메뉴 첫 화면으로 복귀되어야 함	P
47	1.12.0.1.2.0.3.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 8~20개 일때 CANCEL 버튼을 누른다.	음료가 7종류 이상 추가 되어서는 안됨.	P
48	1.12.0.2.2.0.1.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 0~6개 일때 3분 이상 아무런 입력도 주지 않는다.	대기화면으로 전환되야 함	P
49	1.12.0.2.2.0.2.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 7개 일때 3분 이상 아무런 입력도 주지 않는다.	대기화면으로 전환되야 함	P

50	1.12.0.2.2.0.3.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 8~20개 일 때 3분 이상 아무런 입력도 주지 않는다.	음료가 7종류 이상 추가 되어서는 안됨.	P
51	1.12.0.8.2.0.1.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 0~6개 일때 재고를 추가한다.	재고가 1개 증가함.	P
52	1.12.0.8.2.0.2.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 7개 일때 재고를 추가한다.	재고가 1개 증가함.	P
53	1.12.0.8.2.0.3.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 8~20개 일때 재고를 추가한다.	음료가 7종류 이상 추가 되어서는 안됨.	P
54	1.12.0.9.2.0.1.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 0~6개 일때 재고를 삭제한다.	재고가 1개 감소함.	P
55	1.12.0.9.2.0.2.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가	재고가 1개 감소함.	P

		1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 7개 일때 재고를 삭제한다.		
56	1.12.0.9.2.0.3.0.0.	재고 관리 화면에서 사용자 또는 관리자가 선택한 음료의 재고가 1개 이상이고 하나의 DVM에서 재고가 있는 음료 종류의 수가 8~20개 일때 재고를 삭제한다.	음료가 7종류 이상 추가 되어서는 안됨.	P
57	1.13.0.1.0.0.0.0.0.	유통기한 입력화면에서 CANCEL 버튼을 누른다.	item 목록 출력 화면으로 복귀 되어야 함.	P
58	1.13.0.2.0.0.0.0.0.	유통기한 입력화면에서 3분 이상 아무런 입력도 주지 않는다.	대기화면으로 전환되어야 함.	P
59	1.14.0.0.0.0.0.1.0.	유효하지 않은 결제 정보를 갖고 있을 때 결제를 시도 한다.	결제 실패 에러화면이 보여야 함.	P

총 59개의 Test Case 중 59개의 Test Case를 통과하여 **100%**이다.

3. Static Analysis

3.1. Checkstyle

소프트웨어를 개발하는 모든 과정에서 들어가는 비용 중 많은 비용이 유지보수하는 데에 쓰인다. 코딩 컨벤션을 정하면 소스 코드를 유지보수 하는 것에 도움이 된다.
Checkstyle은 코딩 컨벤션을 정의하고 검사해주는 정적 분석 도구이다. Checkstyle로 DVM 프로젝트를 검사한 결과는 다음과 같다.

3.1.1. Pre-modification

CheckStyle Audit

Designed for use with CheckStyle and Ant.

Summary	
Files	**Errors**
22	2530

총 22개의 파일에서 2530개의 에러가 발생했다. 첫 적용시에는 Google Style Guide를 그대로 사용해서 javadoc comment, import 문 사전 순 정렬 조건과 같이 프로젝트에서 적절하지 않은 검사가 포함되어 있었다. import 문 사전 순 정렬의 경우 intellij에서 제공하는 코드 정렬의 기준과 달라 프로젝트에서 적합하지 않다고 판단하였다. 따라서 프로젝트의 config/checkstyle/checkstyle.xml를 수정하여 해당 검사들을 삭제하였다.
그 결과 프로젝트에서 검출된 에러들은 다음과 같다.

Index	Property	Description
1	Indentation	들여쓰기 레벨이 2칸으로 설정했는데 프로젝트에서는 4칸으로 코드를 작성하여 발생함
2	Package naming convention	자바에서 패키지 명명 규칙은 ASCII 문자에 포함되어 있는 소문자로 사용해야 함
3	Class naming convention	자바에서 클래스 이름은 명사이어야 하며, 복합 명사일 경우 각 단어의 첫 글자는 대문자이어야 함. 클래스 이름은 간단하고 명시적으로 작성해야 함. 완전한 단어를 사용하고 두 문자어와 약어는 피하도록 함. 단, 약어가 URL, HTML과 같이 보편적으로 사용되는 경우 괜찮음
4	Variable naming convention	자바에서 변수 이름의 첫 번째 문자는 소문자로 시작하고, 각각의 내부 단어의 첫 번째 문자는 대문자로 시작해야 함 이름이 짧지만 의미 있어야 함 이름의 선택은 그 변수의 사용 의도를 알아낼 수 있도록 의미적이어야 함. 단, 임시 변수와 버릴 변수일 경우는 제외함.
5	Method naming convention	자바에서 메소드의 이름은 동사여야 하며, 복합 단어의 경우 첫 단어는 소문자로 시작하고 그 이후로 나오는 단어의 첫 문자는 대문자로 사용해야 함.
6	Whitespace around	중괄호 뒤에는 공백이 있어야 함
7	Import statement	import 문에서 '.*'와 같은 형태는 피해야 함

		import 문 사이 개행이 없어야 함
8	Operator rule	연산이 길어질 경우 연산자는 구문의 앞에 위치해야 함

전체적으로 프로젝트의 패키지, 클래스, 변수 등 네이밍 컨벤션이 지켜지지 않은 부분에서 가장 많은 에러가 검출되었다. 들여쓰기 레벨을 2로 설정되어 있으나 대부분 들여쓰기를 4칸으로 사용한 부분 또한 상당수 검출되었다. 그 외 중괄호 { 뒤 주석을 작성하거나 import 문에서의 규칙 등에서 검출되었다. 그러나 import 문과 관련된 규칙 중 '.*'의 형태를 사용해야 하는 규칙은 Controller 클래스의 경우 GUI 관련 처리를 도맡아서 하기 때문에 'gui.*'와 같이 사용하는 것이 적절할 수 있어 false alarm으로 보인다.
전반적으로 코딩 컨벤션이 통일되어 있지 않아서 가독성이 좋지 않은 상태로 판단하였다. 지라 이슈로 등록 후 해당 항목에 대해 수정을 권고하였다.

3.1.2. Post-modification

CheckStyle Audit

Designed for use with CheckStyle and Ant.

Summary	
Files	**Errors**
22	121

총 에러 수가 **95.4%(2530 -> 121)** 감소하여 코딩 컨벤션 관련 상당 부분 개선되었음을 확인하였다. 남은 에러들은 CNumber, K, 등 스펙에 정의되어 있는 클래스, 변수, 메소드 명명에 대한 것으로 중요성이 낮은 부분이며, 개발팀과 협의하여 수정하지 않는 것으로 합의하였다. Checkstyle 에러들을 수정한 후 코딩 컨벤션이 잘 지켜지고 있는 것을 확인했다.

3.2. PMD

Problems found

#	File	Line	Problem
1	/home/runner/work/ctip/ctip/src/main/java/gui/AddItemMenu.java	18	Classes implementing Serializable should set a serialVersionUID
2	/home/runner/work/ctip/ctip/src/main/java/gui/AddItemMenu.java	20	Found non-transient, non-static member. Please mark as transient or provide accessors.
3	/home/runner/work/ctip/ctip/src/main/java/gui/AddItemMenu.java	28	Found non-transient, non-static member. Please mark as transient or provide accessors.
4	/home/runner/work/ctip/ctip/src/main/java/gui/AddItemMenu.java	29	Found non-transient, non-static member. Please mark as transient or provide accessors.

Total(162개) = GUI (101개) + Logic (61개) 에서 GUI는 제외하고 분석을 하였다.
PMD는 Summary를 해주고 있지 않고 발견된 문제들을 단순히 나열만 하고 있기 때문에, 분석을 위해 따로 표를 작성하였다.

#	Problem	Priority	Number
1	Found non-transient, non-static member. Please mark as transient or provide accessors.	Medium (3)	1

2	Assigning an Object to null is a code smell. Consider refactoring.	Medium (3)	3
3	Avoid using Literals in Conditional Statements	Medium (3)	36
4	Ensure that resources like this BufferedReader object are closed after use	Medium (3)	2
5	Ensure that resources like this ObjectInputStream object are closed after use	Medium (3)	1
6	Ensure that resources like this PrintWriter object are closed after use	Medium (3)	3
7	Found 'DD'-anomaly for variable.	Low (5)	9
8	Found 'DU'-anomaly for variable.	Low (5)	5
9	Non-static initializers are confusing	Medium (3)	1
Total			**61**

그 중 logic에서 발견된 문제 유형은 총 9개이며, 실질적으로 문제 4, 5, 6은 CloseResource 문제 7, 8은 DataflowAnomalyAnalysis 로 묶이기 때문에 크게 6개의 문제 유형이 발견되었다.
PMD는 가장 심각한 1부터 가상 심각하지 않은 5까지 우선순위를 매긴다. 이번에 발견된 우선순위는 3과 5이며 심각한 위반 사항은 없음을 알 수 있었다.

P #	Problem Name	File, Line	Description
1	BeanMembers ShouldSerialize	logic/CNumber.java, 12	member 변수에 transient, static을 붙여주거나 get/set 함수가 필요함.
2	NullAssignment	logic/Controller.java, 34 logic/Controller.java, 549 logic/Payment.java, 59	변수를 선언하고 있는 상황이 아닐 때, 변수에 null을 할당하는 것은 일반적으로 잘못된 형식임.
3	AvoidLiteralsInIf Condition	logic/Controller.java, 144 logic/Controller.java, 220 logic/Controller.java, 221 logic/Controller.java, 254 logic/Controller.java, 277 logic/Controller.java, 283 logic/Controller.java, 310 logic/Controller.java, 314 logic/Controller.java, 357 logic/Controller.java, 400 logic/Controller.java, 467 logic/Controller.java, 491	조건문에서 하드 코딩된 리터럴을 사용하는 것은 좋지 않음.

		logic/Message.java, 35 logic/Message.java, 38 logic/Message.java, 48 logic/Message.java, 58 logic/Message.java, 69 logic/Message.java, 78 logic/Message.java, 89 logic/MessageQueue.java, 76 logic/MessageQueue.java, 79 logic/MessageQueue.java, 82 logic/MessageQueue.java, 85 logic/MessageQueue.java, 88 logic/MessageQueue.java, 91 logic/MessageQueue.java, 94 logic/MessageQueue.java, 153 logic/MessageQueue.java, 159 logic/MessageQueue.java, 190 logic/MessageQueue.java, 194 logic/MessageQueue.java, 196 logic/MessageQueue.java, 201 logic/MessageQueue.java, 205 logic/MessageQueue.java, 207 logic/MessageQueue.java, 211 logic/Title.java, 21	
4	CloseResource	logic/MessageQueue.java, 49 logic/MessageQueue.java, 126	BufferedReader를 사용하고 close를 하지 않았음.
5	CloseResource	logic/MessageQueue.java, 52	ObjectInputStream를 사용하고 close를 하지 않았음.
6	CloseResource	logic/MessageQueue.java, 50 logic/MessageQueue.java, 53 logic/MessageQueue.java, 127	PrintWriter를 사용하고 close를 하지 않았음.
7	DataflowAnomalyAnalysis	logic/Controller.java, 232-235 logic/Controller.java, 235-235 logic/MessageQueue.java, 34-37 logic/MessageQueue.java, 35-38 logic/MessageQueue.java, 47-63 logic/MessageQueue.java, 48-56 logic/MessageQueue.java, 125-133 logic/MessageQueue.java, 218-225 logic/MessageQueue.java, 225-225	(수정 필수 x) 함수 내에서 정의한 변수가 재정의가 된 것은 좋은 것은 아니나 버그까진 아님.
8	DataflowAnomalyAnalysis	logic/Controller.java, 470-505 logic/Controller.java, 489-505 logic/MessageQueue.java, 50-122 logic/MessageQueue.java, 129-174 logic/MessageQueue.java, 187-253	(수정 필수 x) 함수 내에서 정의한 변수가 함수 내에서 사용되지 않았음. 삭제하거나 변수를 사용하는 것을 권장.

9	NonStaticInitializer	logic/CNumber.java, 14	초기화 블록이 생성자가 생성될 때마다 실행됨. 틀린 구문은 아니지만 거의 사용 되지 않고 혼란을 줄 수 있음. static으로 수정하거나 삭제하는 것을 권장.

문제의 이름과 문제의 발생 위치, 문제가 발생한 이유 및 수정 방향을 작성하여 개발팀에서 수정할 수 있도록 정리하였다.
PMD Source Code Analyzer Project에서 정리한 Error Prone는 총 98개로 그 중 발견된 것은 6개밖에 없으며, 전반적으로 발견된 위반 사항의 우선 순위가 높지 않은 점과 수정하는데 큰 시간이 소요되지 않을 것이라 생각되어 코드 작성 능력이 준수하다고 평가를 내렸다.

3.3. Spotbugs

Summary

Warning Type	Number
Bad practice Warnings	8
Internationalization Warnings	4
Performance Warnings	2
Dodgy code Warnings	21
Total	**35**

210531 생성된 report 기준 총 35개의 경고가 발생하였다. 그 중 logic 패키지에서 32개, gui 패키지에서 3개의 경고가 발생했다.
gui 패키지에서 발생한 경고는 제외하였다.

- High Priority Warnings : 4개
- Medium Priority Warnings : 3개

- Bad Practice Warnings

Index	Warning	Code	Warning 발생 이유
1	NM_CONFUSING: Confusing method names	CNumber.getDvmID() & Payment.getDVMId()	대문자 여부만 다른 함수이름 때문에 발생하였다.
2	NM_CONFUSING: Confusing method names	CNumber.setDvmID(int) & Payment.setDVMId(int)	대문자 여부만 다른 함수이름 때문에 발생하였다.
3	NM_FIELD_NAMING_CONVENTION:	CNumber.DvmID	변수 이름은 소문자로 시작해야한다.

	Field names should start with a lowercase letter		
4,5	RV_RETURN_VALUE_IGNORED_BAD_PRACTICE: Method ignores exceptional return value	MessageQueue.java :[line 75, 195, 206, 212]	offer함수의 return값을 검사하여 오류가 발생하였는지 확인이 필요하다.

- Internationalization Warnings

Index	Warning	Code	Warning 발생 이유
6~9	DM_DEFAULT_ENCODING: Reliance on default encoding	MessageQueue.java :[line 65, 66, 135, 136]	생성자 인자에 인코딩 방식을 추가로 전달해야 시스템 종속적인 인코딩이 시행되지 않는다. `new InputStreamReader(socket.getInputStream(),StandardCharsets.UTF_8)`

- Performance Warnings

Index	Warning	Code	해결방법
10	SBSC_USE_STRINGBUFFER_CONCATENATION: Method concatenates strings using + in a loop	CNumber.java :[line 70]	Loop안에서 String concatenate를 할 경우에는 StringBuffer을 사용하여 append하고 Loop 종료 후에 String으로 변환하는 것이 좋다. 다만 Loop를 3번만 도는 것으로 확인되어 이를 꼭 수정할 필요는 없을 것으로 생각된다.
11	SIC_INNER_SHOULD_BE_STATIC_ANON: Could be refactored into a named static inner class	Title.java :[line 48]	Outer Class의 reference를 이용하지 않아 익명 inner class보다 static inner class를 따로 선언하여 사용하는 것이 좋다. 하지만 IntelliJ에서 제안하는대로 `Comparator.comparingInt(Item::getExpirationDate)` 로 코드를 바꾸는 것이 더 좋아보인다.

- Dodgy Code Warnings

Index	Warning	Code	해결방법
12~30	BC_UNCONFIRMED_CAST: Unchecked/unconfirmed cast	Controller.java: [line 339, 349, 169, 520, 133, 91, 254,278,271,258, 228,245,264 214,284, 185, 444, 478, 460, 73, 427, 528, 301, 391, 381]	JFrame객체를 gui의 각 Class로 cast할 때 castable한지 확인하는 부분이 존재하지 않아 발생한 경고이다.
31, 32	NP_DEREFERENCE_OF_READLINE_VALUE: Dereference of the result of readLine() without nullcheck	MessageQueue.java :[line 68, 147]	readLine()의 결과에 대해 null checking이 필요하다.

- 경고 타입별로 표를 나누고 각 경고별로 경고 내용, 발생한 코드 위치, 상세 설명을 작성하였다.
- 경고 중 높은 우선순위의 경고와 중간 우선순위의 경고는 바탕색을 넣어 표시하였다.
- 다수의 경고가 오류값을 return할 수 있는 return value를 검사하지 않거나, Cast할 시 castable을 검사하지 않아 발생하였다. 오류 상황에 대처할 수 있는 코드도 작성해야함을 유의해야한다.

Metrics

2105 lines of code analyzed, in 37 classes, in 2 packages.

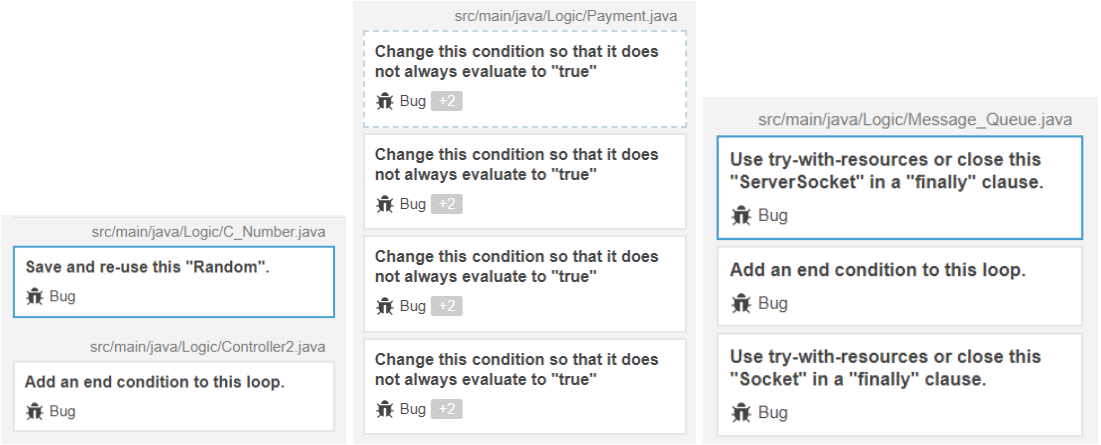
Metric	Total	Density*
High Priority Warnings	4	1.90
Medium Priority Warnings	3	1.43
Low Priority Warnings	28	13.30
Total Warnings	**35**	**16.63**

(Defects per Thousand lines of non-commenting source statements)*

- 전체적으로 코드 라인 수에 비한 Warning Density가 16.63으로 적은 편이어서 coding rule을 잘 지키며 코드를 작성했다고 생각한다.

3.4. SonarQube

개발 과정에서 10개의 버그와 수백개의 코드 스멜이 발생했다. 검출한 버그의 종류는 총 4개가 발생했으며 While 문에 종료 조건을 추가하거나 조건문이 항상 true인 경우 등 수정할 필요가 있는 버그였다. 해당 부분은 개발팀이 수정을 완료한 상태이다.

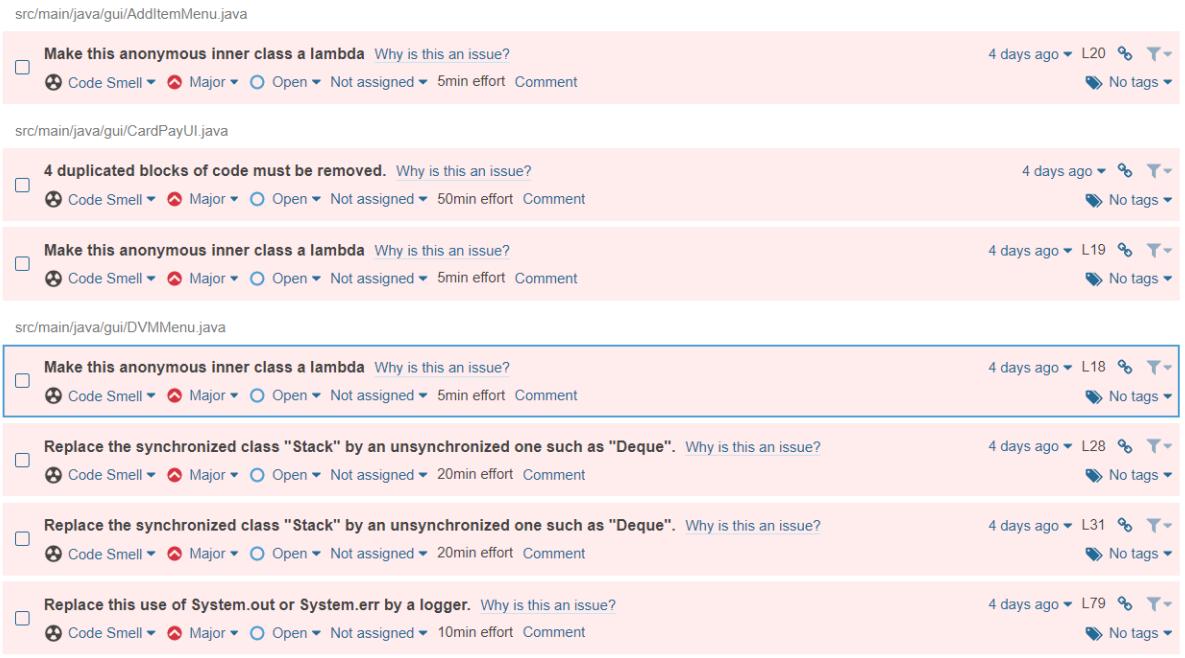


Checkstyle, SonarQube의 버그를 수정하는 과정에서 코드 스멜 역시 137개로 줄어들었다. 코드 스멜은 5개의 타입(Blocker, Critical, Major, Minor, Info)으로 구분된다. 그 중 프로젝트에서 검출된 타입은 Critical, Major, Minor, Info이다.

Critical로 분류된 코드 스멜은 모두 Cognitive Complexitiy에 관한 것이었다. 기능에 대한 에러는 없어 선택적으로 수정하도록 결정했다.

[Critical]

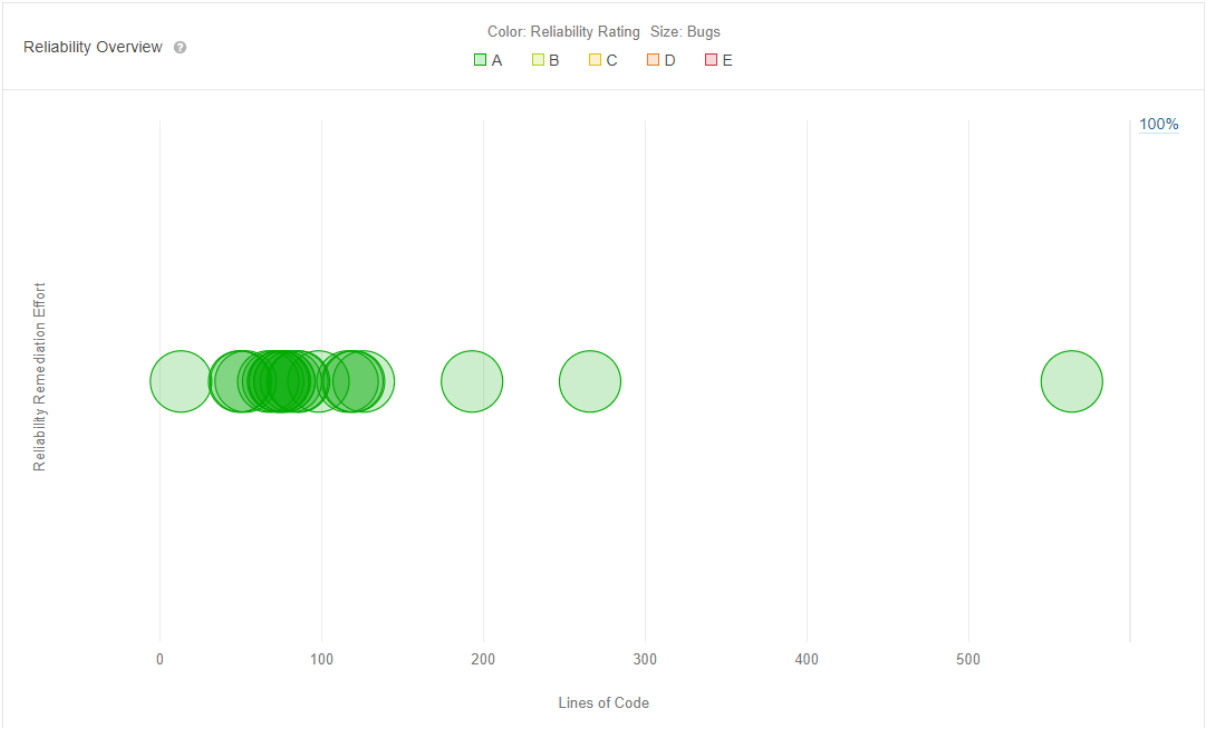
Major로 분류된 코드 스멜은 익명 클래스를 람다식으로 수정, 중복 코드 제거, 로그를 logger로 표현하는 등 코딩 스타일에 대한 내용이었다. Stack의 경우, 동기화가 필요하지 않으면 Deque을 쓰는 것을 권장하고 있는데 이 부분은 반영하는 것이 좋다고 판단한다.



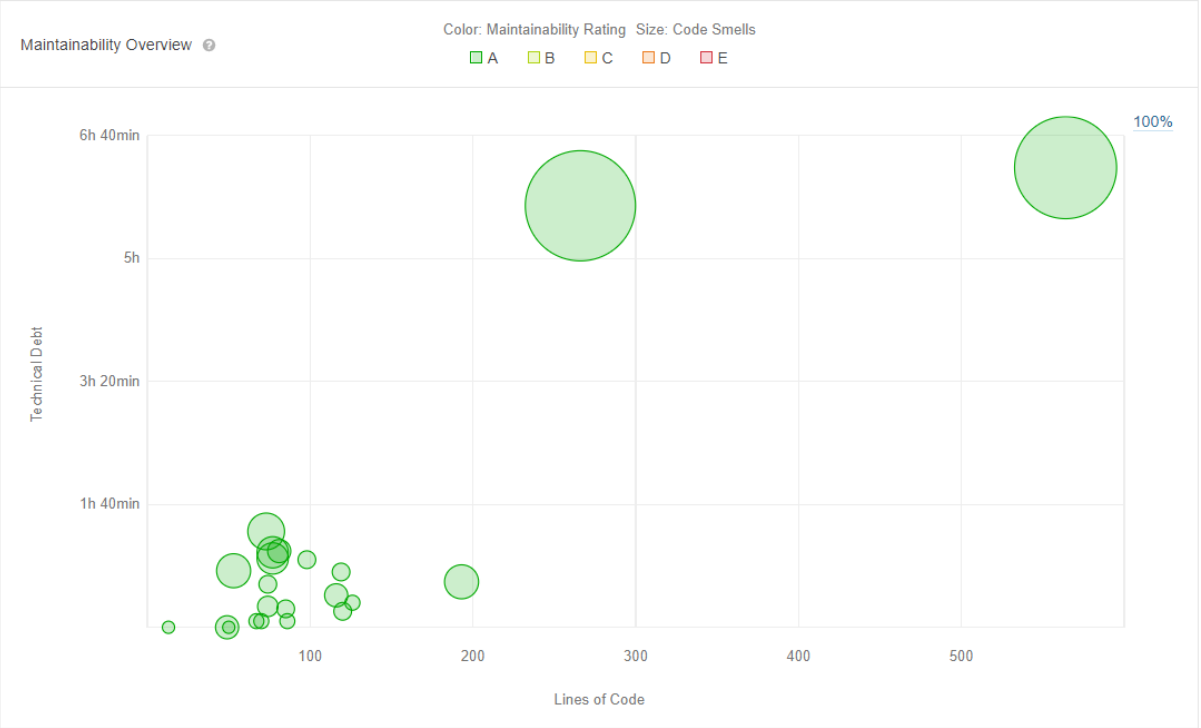
[Major]

Minor로 분류된 코드 스멜은 Checkstyle에서 합의한 내용과 일치하였다. Info로 분류된 코드 스멜을 TODO 코멘트를 삭제하는 것으로 해결할 수 있는 간단한 문제이다.

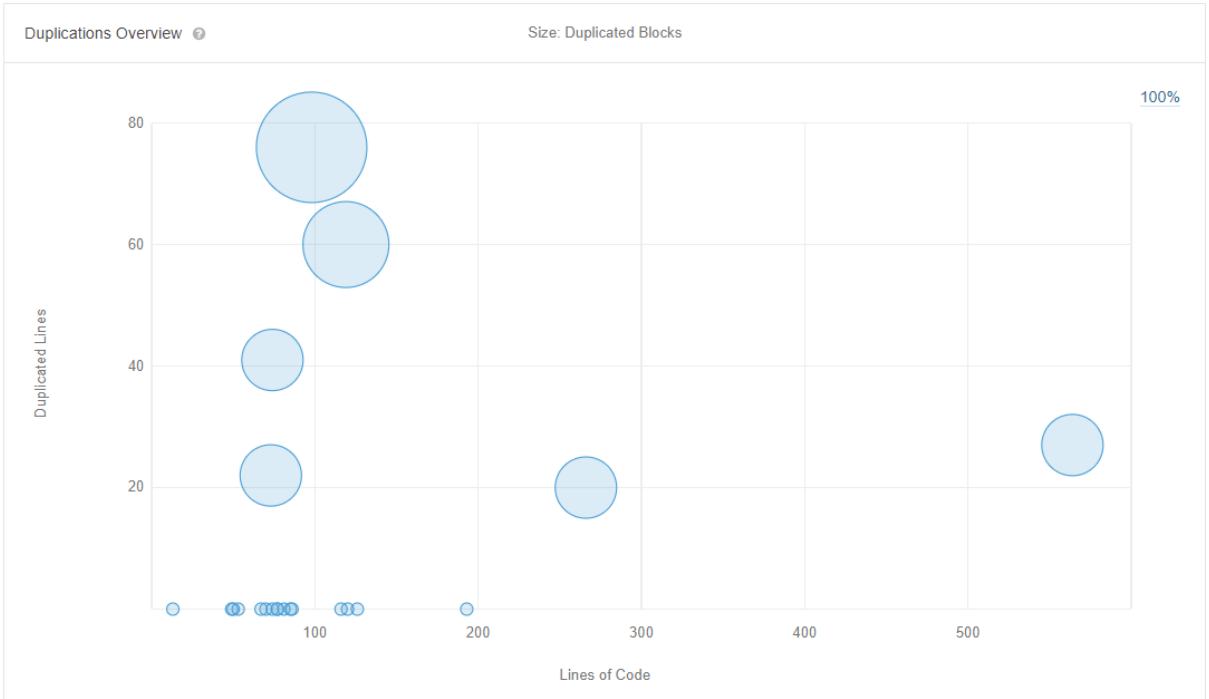
3.4.1. Reliability



3.4.2. Maintainability



3.4.3. Duplication



SonarQube로 실제 버그가 발생할 수 있는 내용들을 검출하여 수정하였으며 그 외에는 특별히 수정할 필요가 없는 false alarm으로 판단하였다. Maintainability와 Duplication 역시 큰 문제가 없어 프로젝트의 퀄리티가 괜찮다고 판단한다.

3.5. JaCoCo

전체 코드(gui package, logic package) 중 logic package에 대해서만 JUnit Test 작성이 이루어졌다. 따라서 logic package에서 수행된 JUnit tests만 분석하였다.

3.5.1. JUnit Test Report

28	0	0	0.101s
tests	failures	ignored	duration

100% successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
CNumberManagerTest	5	0	0	0.004s	100%
CNumberTest	1	0	0	0.055s	100%
ControllerTest	1	0	0	0.001s	100%
DVMTest	4	0	0	0.004s	100%
ItemTest	1	0	0	0.002s	100%
MessageQueueTest	1	0	0	0.003s	100%
MessageTest	7	0	0	0.015s	100%
PaymentTest	3	0	0	0.012s	100%
TitleTest	5	0	0	0.005s	100%

- 총 28개의 Tests가 진행되었다.
- main과 생성자 메소드는 제외하고 getter, setter는 포함하여 logic package의 총 메소드 수는 8+11+42+13+2+23+17+9+12 = 137개이다.
- 총 메소드 수는 137개이지만 Test 수는 28개이다. 전반적으로 테스트가 부족해 보인다.
- 28개의 Tests로 커버되지 않는(예외 처리로 테스트 하기 어려운 부분을 제외한) Methods 수는 69개이다.
- 28개의 Tests 중 assert문이 1번만 사용 된 Tests 수가 13개이다.
- 전반적으로 Test 당 assert문의 수가 적다.

3.5.2. JaCoCo Report

logic

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Controller		13%		0%	127	136	358	398	37	46	0	1
MessageQueue		24%		25%	44	56	136	185	12	19	0	1
Message		81%		57%	11	31	17	85	5	24	0	1
Title		81%		85%	8	20	11	39	6	13	0	1
CNumber		93%		71%	6	16	6	48	2	9	0	1
Payment		89%		100%	3	16	6	41	3	10	0	1
DVM		81%		n/a	4	14	6	26	4	14	0	1
CNumberManager		91%		100%	2	14	4	27	2	12	0	1
Item		100%		n/a	0	3	0	6	0	3	0	1
Title.new Comparator() {...}		100%		n/a	0	2	0	2	0	2	0	1
Total	1,973 of 3,109	36%	247 of 312	20%	205	308	544	856	71	152	0	10

- logic package에서 instruction coverage는 36%, branch coverage는 20%를 기록하였다.

- Controller, MessageQueue에 대부분의 코드가 집중되어 있지만 13% 24%라는 낮은 Coverage를 보였다.
- Controller, MessageQueue를 제외한 Class들에서는 모두 80% 이상의 Coverage를 보였다.

- getter, setter 메소드들(총 36개)을 간단하게 테스트한다면, Coverage가 의미있게 상승할 것이다.

- Instruction 혹은 Branch가 커버되지 않은 method들을 다음과 같이 나열하였다. 개발팀과 논의 후 필요하다고 생각되는 method들에 대해서는 추가적인 JUnit test를 진행할 것이다.

Index	Class	Method	Description(Optional)
1	Controller	manEditItem(int)	
2		reqFindDVM()	
3		cardPay()	
4		smartPay()	
5		inputCNumber()	
6		inputSelect()	
7		selectPayment(int)	
8		returnItem(Title, boolean)	
9		printCNumber(String, int, int)	
10		selectDVM()	
11		manSelectTitle()	
12		infoNoUsableDVM()	
13		reqMainMenu()	
14		manShowItem(int)	
15		infoCNumberError()	
16		manShowTitle()	
17		printSelectPay(int)	
18		infoNoItem()	

19		showInputLine()	
20		showMainMenu()	
21		showUsableDVM()	
22		showCardPay()	
23		showSmartPay()	
24		cancelPay()	
25		setK(JFrame)	
26		setDel(int)	
27		cancelItem()	
28		returnMain()	
29		getK()	
30		getDel()	
31		setTitleList(ArrayList)	
32		setDvmStack(Stack)	
33		setCm(CNumberManager)	
34		setMq(MessageQueue)	
35		getDvmStack()	
36		getMq()	
37	MessageQueue	msgReceive(int)	
38		dequeue()	
39		msgSend(Message)	
40		run()	
41		getIP()	
42		setLoc(int)	
43		setStk(int)	
44		setMsgQueue(Queue)	
45		setStkMsgQueue(Queue)	
46		setLocMsgQueue(Queue)	
47	Message	translate(int, int, int, double,	

		double, int, int, boolean)	
48		setMyId(int)	
49		setXAddress(double)	
50		setYAddress(double)	
51		setBoolData(boolean)	
52	Title	setHold(Integer)	
53		setName(String)	
54		setPrice(int)	
55		setItemList(ArrayList)	
56		updateStock(int, boolean)	재고 오류인 경우에 대해서 테스트 안됨.
57		getName()	
58		getPrice()	
59	CNumber	setTitleId(int)	
60		setDvmID(int)	
61	Payment	setTitleId(int)	
62		setDVMId(int)	
63		setErrorLog(String)	
64	DVM	setCurrentX(double)	
65		setCurrentY(double)	
66		getCurrentX()	
67		getCurrentY()	
68	CNumberManager	setCList(HashMap)	
69		setChCList(HashMap)	

- Test 개수가 메소드 수에 비해 부족하며, 총 assert문 수는 63개로 Test 1개 당 assert 수는 2.25개이다. Test 1개 당 assert 수가 적다고 판단된다.
- Controller, MessageQueue JUnit Test에 주석처리된 Test들이 20개이다. 이를 잘 수정하여 복구하고, getter, setter Test를 간단하게 추가한다면 Coverage가 지금에 비해 높은 수준으로 올라갈 것이 기대된다.

4. Quality Measurement

4.1. Product Quality Factors

4.1.1. Reliability

정해진 스펙대로 작동하는지 Use Case 별 DVM의 수를 조절하면서 각 10회 테스트를 진행하였다.

4.1.1.1. Use case 1 - 음료 리스트 화면 출력

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.2. Use case 2 - 구매할 음료 입력

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.3. Use case 3 - 사용자 선택 취소

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.4. Use case 4 - 사용자 인증번호 입력

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Alter	P	P	P	P	P	P	P	P	P	P	10/10

Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.5. Use case 5 - 재고가 부족한 제품 안내

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.6. Use case 6 - 구매 가능한 자판기 안내

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.7. Use case 7 - 사용자 자판기 선택

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.8. Use case 8 - 결제 수단 목록 출력

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.9. Use case 9 - 결제 수단 결정

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.10. Use case 10 - 간편 결제

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Alter	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.11. Use case 11 - 카드 결제

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Alter	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.12. Use case 12 - 결제 취소

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.13. Use case 13 - 인증번호 생성

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.14. Use case 14 - 인증번호 출력

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.15. Use case 15 - 음료 전달

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.16. Use case 16 - 재고 수량 업데이트

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Alter	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.17. Use case 17 - 관리자 메뉴 출력

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.18. Use case 18 - 재고 관리

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Except	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.19. Use case 19 - 위치 정보 확인

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.20. Use case 20 - 재고 정보 확인

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.21. Use case 21 - 메시지 발신

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10
Alter	P	P	P	P	P	P	P	P	P	P	10/10

4.1.1.22. Use case 22 - 메시지 수신

회수	1	2	3	4	5	6	7	8	9	10	평균
결과	P	P	P	P	P	P	P	P	P	P	10/10

4.1.2. Correctness

스펙과 동일한지에 대한 척도로 전체 Use Case 중 Reliability가 100인 Use Case의 비율로 측정하였다.
22개의 Use Case 중 22개의 Use Case 모두 Reliability가 100%이므로 Correctness는 **100%**이다.

4.1.3. Usability

Index	Description	Result
1	1회 마우스 클릭으로 버튼이 동작하는가?	P

2	재고가 20개 있을 때, 10회 이내의 마우스 스크롤로 스크롤바가 최하단까지 이동하는가?	P
3	화면 이동 시 화면의 크기가 변하지 않는가?	P
4	안내 문구를 읽기 적절한 시간(10초 이상)을 부여하는가?	P
5	선결제 인증번호 화면이 지속적으로 떠있는가?	P
6	선결제 인증번호를 다시 확인할 수 있는가?	F
7	폰트의 크기가 12f 이상인가?	P
8	아무런 입력이 없을 때 잠금화면으로 돌아가는 시간이 너무 짧거나 너무 길지 않은가?	P
9	컴포넌트들이 시각적으로 잘 구분되어 적절하게 보이는가?	P
10	자판기를 이용하는 방법에 대한 설명이 존재하는가?	P
11	버튼 크기가 충분히 큰가?	P
12	결제 성공/실패 입력 버튼이 잘 눌렸는지 확인할 수 있는가?	F
13	재고가 있는 자판기 검색 시간이 짧은가? (10초 이내)	F

13개 중 10개의 항목에서 통과했으므로 Usability는 **77%**이다.

4.1.4. Maintainability

3절에서 분석한 각 Static Analysis 도구별 점수를 측정

- Checkstyle: 5/5
- PMD: 4/5
- Spotbugs: 4/5
- SonarQube: 5/5

기타

- ID, 주소, 관리자코드 하드코딩: 0/20

18 / 40점이므로, Maintainability는 **45%**이다.

4.2. Process Quality Factor

4.2.1. Schedule

각 스테이지 별 기간 내 정상적으로 수행했는지 확인

팀발표 #1	팀발표 #2	팀발표 #3	팀발표 #4	팀발표 #5
OOPT Stage 1000	OOPT Stage 2030	OOPT Stage 2040	OOPT 1st Cycle	OOPT 2nd Cycle
Planning	Analysis	Design	Implementation & Unit Test	Implementation & Unit Test 2

5개의 스테이지 중 4개의 스테이지를 기간 내 정상적으로 수행했으므로 Schedule Visibility는 **80%**이다.

4.3. Total Quality

4.3.1. Project Planning

4.3.1.1 SRS

전체 하위 항목 32개 정상적으로 작성된 항목 32개로 비율을 계산 하였다.
32 / 32 * 100 = 100%

4.3.1.2 SDS

전체 하위 항목 30개 중 정상적으로 작성된 항목 30개로 비율을 계산 하였다.
30 / 30 * 100 = 100%

4.3.1.3 OOPT - 1000

전체 하위 항목 49 개 중 정상적으로 작성된 항목 49개로 비율을 계산 하였다.
49 / 49 * 100 = 100%

4.3.1.4 OOPT - 2000

전체 하위 항목 77개 중 정상적으로 작성된 항목이 76개로 비율을 계산하였다.
76 / 77 * 100 = 98.7%

평균: (100 + 100 + 100 + 98.7) / 4 = 99.7%

기획 Total Quality: 0.2 * 99.7% = **19.94**

4.3.2. Development

개발 40% 중
Reliability(25%), Correctness(25%), Usability(25%), Maintainability(25%)

Reliability : 360 / 360 * 100 = 100%
Correctness : 22 / 22 * 100 = 100%
Usability : 10 / 12 * 100 = 83%
Maintainability : 18 / 40 * 100 = 45%

개발 Total Quality = 0.4 * 82 % = **32.8**

4.3.3. Verification & Validation

테스트 40% 중
Spec Review(25%), System Test(25%), Static Analysis(25%), Unit Test(25%)

Spec Review : 25/25
System Test : 23.5/25
Static Analysis : 25/25
Unit Test : 12.5/25

테스트 Total Quality : 0.4 * 86% = **34.4**

Total Quality :19.94 + 32.8 + 34.4 = **87.14**