Final Presentation

-5팀-
이지영 (팀장)
강민호
김정연
유경원

INDEX

01 Spec Review (2nd) – Stage 1001

- ‘직관적인 UI를 제공하고 사용하기 편해야 한다’는 구체적이지 못했던 표현을 ‘벤치마킹’ 으로 더 명확하게 파악할 수 있는 표현으로 수정 요청함
- 데이터 지연은 필연적이므로 달성하지 못하는 목표 대신 더 명확한 표현으로 수정 요청함
- Java 버전 표시 요청함

- Non Functional Requirements
  - 직관적인 UI를 제공하고 사용하기 편해야 한다.
  - Java 사용
  - 네트워크 통신 중 데이터 누락이 없어야 한다.
  - 단독망 사용
  - 네트워크 통신 중 Latency 가 없어야 한다.

=> 수정 확인 완료

- Non Functional Requirements
  - 시중에 존재하는 fresh store 자판기를 벤치마크하여 UI를 제공한다.

  - Java 사용(version : JDK 1.8)
  - 네트워크 통신 중에 투명성과 무결성이 제공된다.
  - 단독망 사용.

01 Spec Review (2nd) – Stage 1003

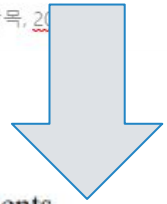
- 0.1초, 1초로 통일되지 않았던 통신 응답 시간을 통일시킴

- 자판기 간 통신 응답 시간이 1초 이내로 수행되어야 한다.
  데이터를 확인하는 시간이 1초 이내로 수행되어야 한다.

3.3 Performance Requirements

- Network Message들의 전송속도는 **0.1s**만에 이루어 지고 전송간 오류가 없어야 한다.
- 사용자가 **screen**에 입력 후 해당 기능이 실행되는데 걸리는 시간은 **0.1s** 이내여야 한다.

SRS 3.3 Performance Requirements 항목, 2



- Performance Requirements
  - 자판기간 통신 응답 시간이 1초 이내로 수행되어야 한다.
  - 데이터를 확인하는 시간이 1초 이내로 수행되어야 한다.

=> 수정 확인 완료

01 Spec Review (2nd) – Stage 1004

- Stock, Count 등 사실상 같은 의미이나 용어가 하나로 통일되지 않았었음.
- 용어의 정의를 item의 수량으로 명백하게 표현 요청

- 1004.Record Term in Glossary(p.7)

Stock	재고
Count	Item의 수량

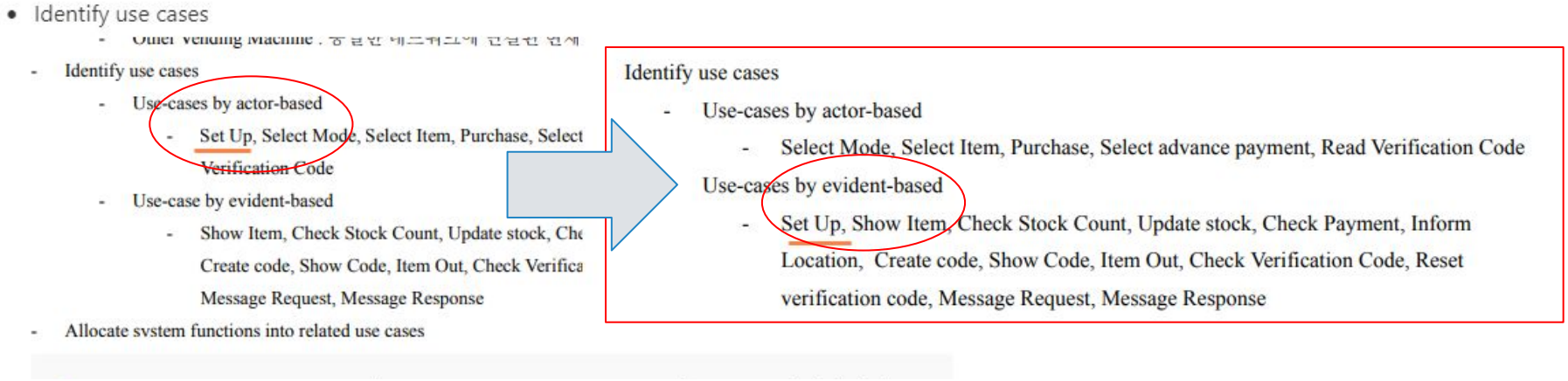
1004. Record Terms in Glossary

Term	Description	Remarks
Item	자판기에 판매되는 음료	
Purchase	결제 유무	
User	분산 자판기를 사용하는 사용자	
Message	네트워크 메시지	
Msg	Message의 약어	
Verification Code	선구매 인증번호	
Count	item의 수량	
Mode	사용자에게 제공하는 구매 유형.	
VM	Vending Machine	
ID	자판기를 구별하기 위한 고유의 아이디	
Broadcast	자신을 제외한 모든 자판기에게 message를 요청하는 과정	

=> 수정 확인 완료

01 Spec Review (2nd) – Stage 1006

- 최초 프로그램 실행시 동작되는 UseCase, 'Set up'은 Hidden임에도 evident-based 카테고리가 아닌 actor-based 카테고리에 들어가 있었음.
- Use case's'인데 s가 누락된 오타가 있었음.

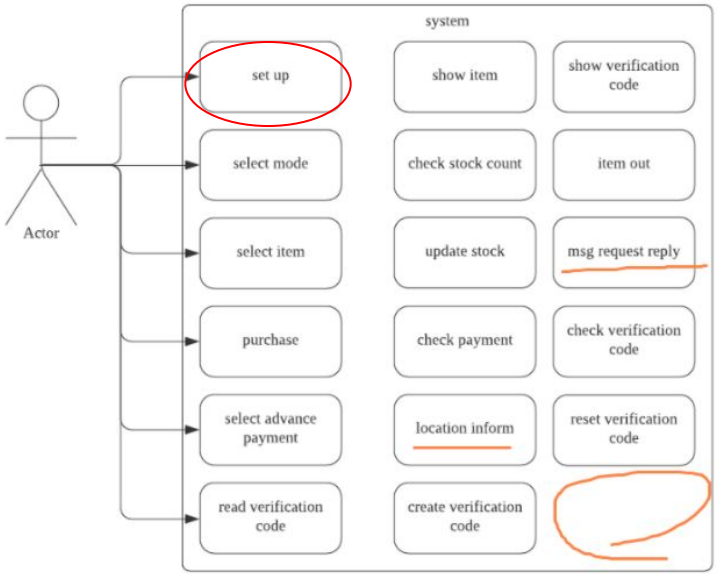


=> 수정 확인 완료

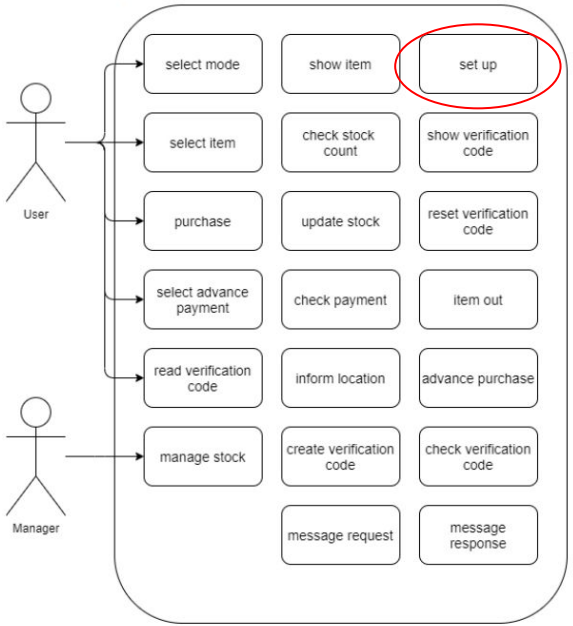
01 Spec Review (2nd) – Stage 1006

- Define System Boundary에는 Manager, User Actor가 따로 있었으나 한가지만 표현됨.
- Set up은 hidden UseCase라 actor가 없어야 하는데 있는 것으로 표시됨.
- 누락된 UseCase들이 있었음.

- use case diagram



- Draw a use case diagram



=> 수정 확인 완료

01 Spec Review (2nd) – Stage 1009

- Function name으로 'Set up all'을 써야하는 데 Usecase명 'Set up' 을 적음 -> 수정 확인
- 6번 Negative Test Case가 무엇을 의미하는지 알 수 없어 명확히 수정 요청 -> 수정 확인

- Functional Requirement Test Case

  "Set up"는 Usecase 명입니다. Function명은 Set up all이므로 p.9에 정의된 Fucntion 이름으로 수정하셔야합니다.

  - No.6 Check Stock Count
    - Negative Test Case
      - 재고가 있는데 없다고 파악하는 경우, 다른 자판기의 재고 수를 요청한다

6	Check Stock Count	1. 기본: 실제 재고와 일치하는 값으로 저장해 다음단계를 진행한다.	1. 재고가 있는데 없다고 파악하는 경우 : 다른 자판기의 재고 수를 요청한다.

  Q. 이게 무슨 뜻인가요?
  A. local에 재고가 없고 remote에 재고가 있는 경우를 나타냅니다.

  "자판기에 재고가 없지만, 다른 자판기에 재고가 있는 경우"로 수정 부탁드립니다.

6	Check Stock Count	1. 기본: 실제 재고와 일치하는 값으로 저장해 다음단계를 진행한다.	1. 현재 자판기에 재고가 없지만, 다른 자판기에 재고가 있는 경우 : : 다른 자판기의 재고 수를 요청한다.

=> 수정 확인 완료

01 Spec Review (2nd) – Stage 2030

이전문서와의 통일 및 모호한 표현 수정

- 13. Create Verification Code (p.10)

Cross Reference	System Functions: R2.10, R2.6 Use Case: "Advance Purchase"
Pre-Requisites	선택된 음료에 대한 결제가 완료됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 새로운 선구매 코드를 생성한다.

Cross Reference	System Functions: R2.10, R2.6 Use Case: "Advance Purchase"
Pre-Requisites	선택된 음료에 대한 선결제가 완료됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 새로운 선구매 코드를 생성한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

현장구매와 선구매 결제의 명확한 구분 필요 => 수정 확인 완료

- 12. Inform Location (p.9)

Cross Reference	System Functions: R2.9, R2.8, R4.1 Use Case: "Select Advance Payment", "Message Request"
Pre-Requisites	음료가 선택됨

Pre-Requisites	위치보기를 누름 자판기의 정보를 알고 있음
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 음료를 갖고있는 자판기의 위치를 보여준다.

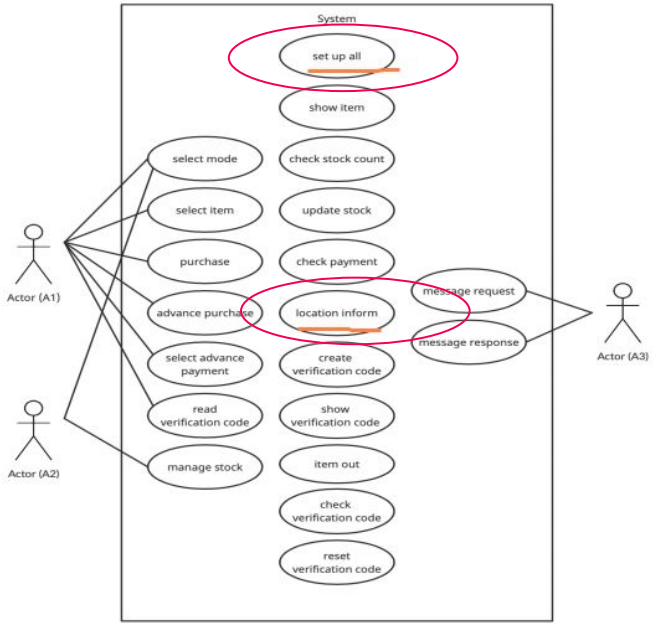
Pre-Requisites의 오류 => 수정 확인 완료

01 Spec Review (2nd) – Stage 2030

Use case 명칭오류

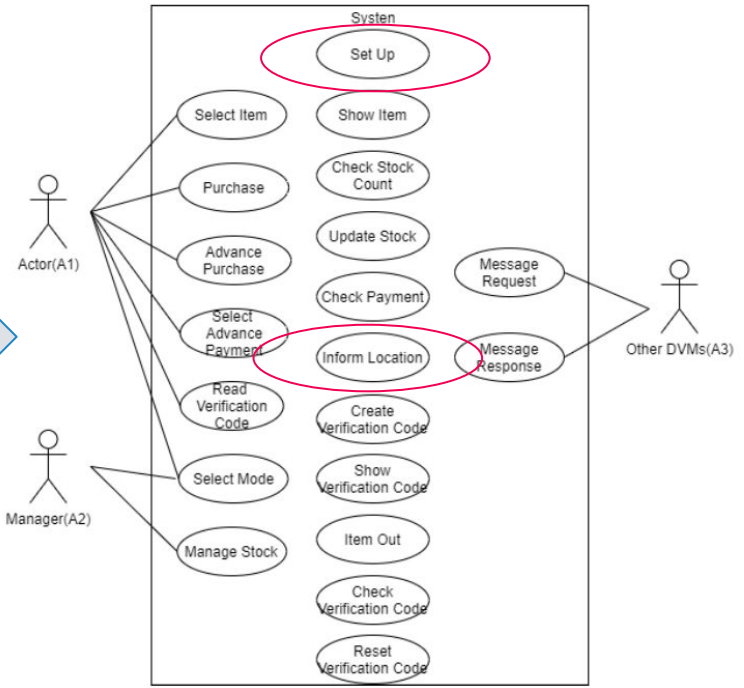
Use case 명칭과 function의 이름이 혼동되어 사용되고 있는 부분
=> 수정 확인 완료

2132. Refine Use Case Diagrams



* (A1): User, (A2): Manager, (A3): Other DVMs

2132. Refine Use Case Diagrams



* (A1): User, (A2): Manager, (A3): Other DVMs

01 Spec Review (2nd) – Stage 2030

수정되지 않은 사항들

- **2133. Define System Sequence Diagrams (p.16)**

  2131. Define Essential Use Cases 항목의 수정사항에 맞게 노트 내용 수정 부탁드립니다.

앞에서 수정한 내용에 맞게 2, 3, 7, 11, 18번 UseCase에 대한 시퀀스 다이어그램 노트 수정 필요

- 13. Create Verification Code (p.10)

Cross Reference	System Functions: R2.10, R2.6 Use Case: "Advance Purchase"
Pre-Requisites	선택된 음료에 대한 결제가 완료됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 새로운 선구매 코드를 생성한다.

  - Pre-Requisites

    보다 정확하게 "선택된 음료에 대한 선결제가 완료됨."으로 수정 부탁드립니다.

  - Typical Courses of Events

    1. (S): "새로운 선구매 코드를 생성한다." → "6자리 난수의 선구매 코드를 생성한다." 로 수정해야 합니다. OOPT 1000 문서에서 이미 언급된 내용이기 때문입니다.

Stage 1000에서 선구매 코드는 6자리로 결정하였으므로 구체화하여 작성을 요청하였으나 수정되지 않음.

01 Spec Review (2nd) – Stage 2030

수정되지 않은 사항들

- 5. Select Item (p.5)
  - Typical Courses of Events

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A1): 사용자가 구매하고자 하는 음료를 선택한다. 2. (S): 선택한 음료에 대해 "Check Stock Count"를 실행한다. 3-1. (S): 재고가 현 재자판기에 있으면 "Purchase"를 실행한다. 3-2.1. (S): 재고가 다른 자판기에 있으면 "Select Advance Payment"를 실행한다.

3-1 부분을 3으로 두고, 3-2 내용은 또 다른 성공 시나리오이므로 Alternative Course of Events로 이동시키는 것이 좋습니다.

성공 시나리오 분기점이 있어 Alternative Course of Events로 분리해 넣는 것이 좋다고 하였으나 변동 없음.

01 Spec Review – Stage 2040

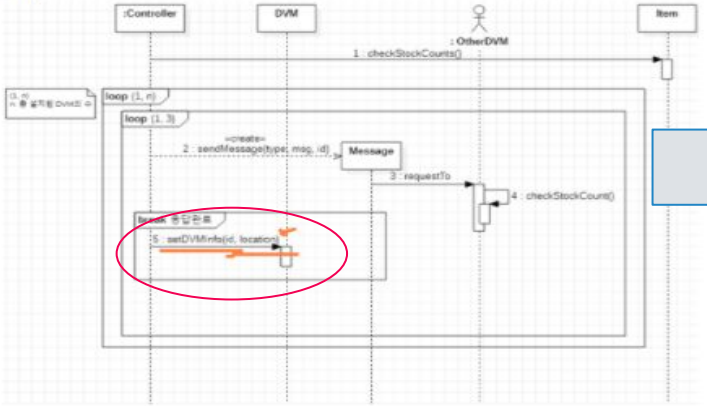
Interaction Diagram에서 메소드의 매개변수가 누락됨

2143. Define Interaction Diagrams

- (1) Set up(pg 19)

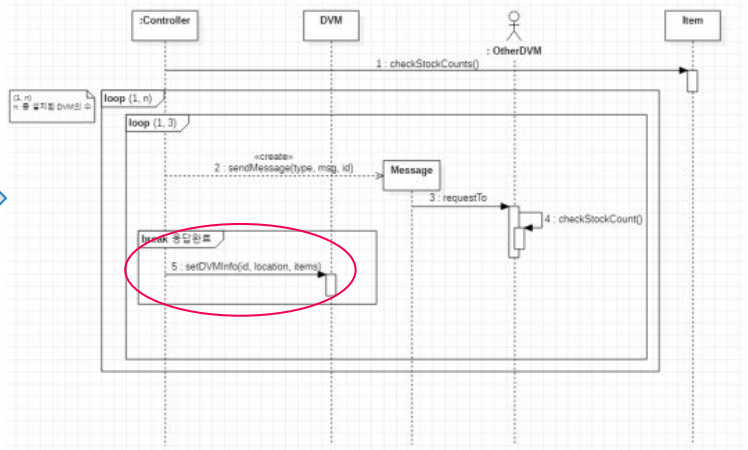
2143. Define Interaction Diagrams

(1) Set Up



2143. Define Interaction Diagrams

(1) Set Up



"setDVMInfo" 오퍼레이션은 DVM의 id, location 그리고 item의 count 항목을 업데이트하는 것이므로 매개변수로 item도 같이 넣어야 합니다.
⇒ "setDVMInfo(id, location)" → setDVMInfo(id, location, item)

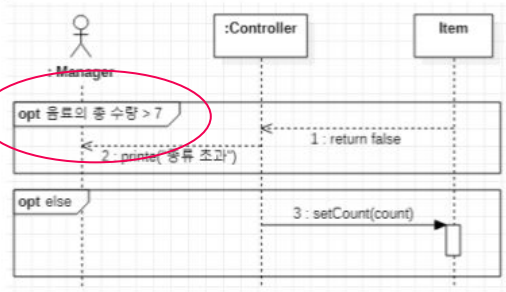
매개변수 누락 => 수정 확인 완료

01 Spec Review – Stage 2040

음료의 총 수량이 음료의 종류를 나타내는 건지 전체 음료의 개수를 의미하는 건지 명확하지 않았음.

- (7) Update Stock (pg 22)

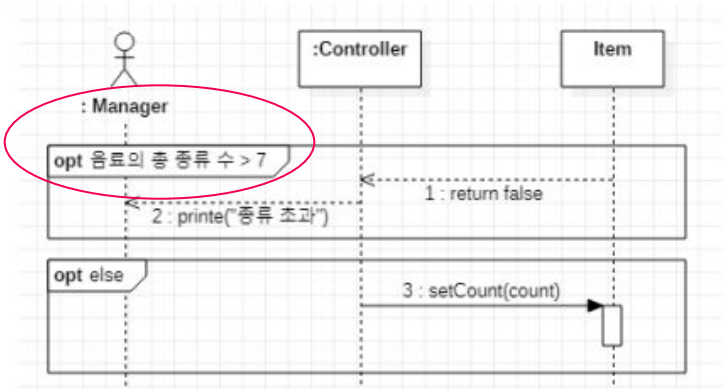
(7) Update Stock



Q. 음료의 총 수량이 7초과일경우 종류 초과 라는 에러 메시지를 생성한다고 되어있는데 "음료의 총 수량"이라는 의미는 음료의 종류가 7가지를 넘어간 경우인가요/음료의 재고의 수가 7개를 넘어간 경우인가요?
A. 음료의 종류의 가짓수가 7개를 초과한 경우입니다.

음료의 종류의 가짓수가 7개 초과한 경우라고 의미가 분명하게 나타나게 수정바랍니다.

(7) Update Stock



의미전달이 정확하지 않은 부분 => 수정 확인 완료

01 Spec Review – Stage 2040

수정된 UI 반영 : 남은 시간은 더 이상 표시하지 않게 됨.

- 7) Show Verification Code(pg 18)

7) Show Verification Code

Q. 남은 시간에 관한 정의는 이전 문서에는 없었는데 UI에서는 표현이 되어 있었으나 추후 문서에서는 수정되었습니다.

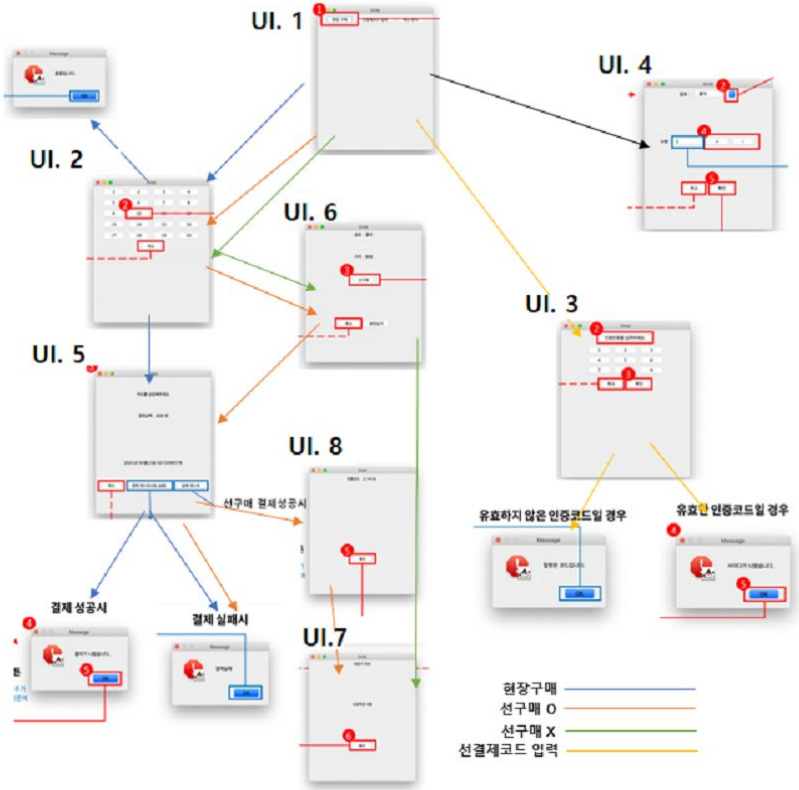
7) Show Verification Code

UI 변경 => 수정 확인 완료

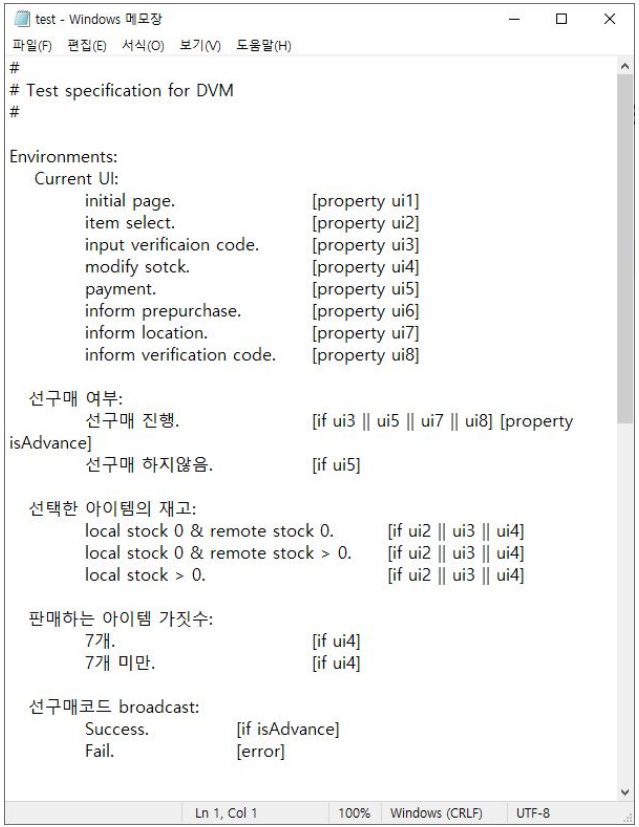
02 Functional Testing Macro

매크로 제작

- 1차 검증 이후 새로 작성된 코드에 맞게 매크로를 다시 제작하였음
- 다음 5가지 flow가 자동화 됨
  1. 파랑색 화살표에 해당하는 flow (현장구매)
  2. 빨간색 화살표에 해당하는 flow (선구매)
  3. 노란색 화살표에 해당하는 flow (선구매코드입력)
  4. 아래 그림에는 없는 flow (품절)
  5. 아래 그림에는 없는 flow (재고 관리)



02 System Testing (2nd) - CPT

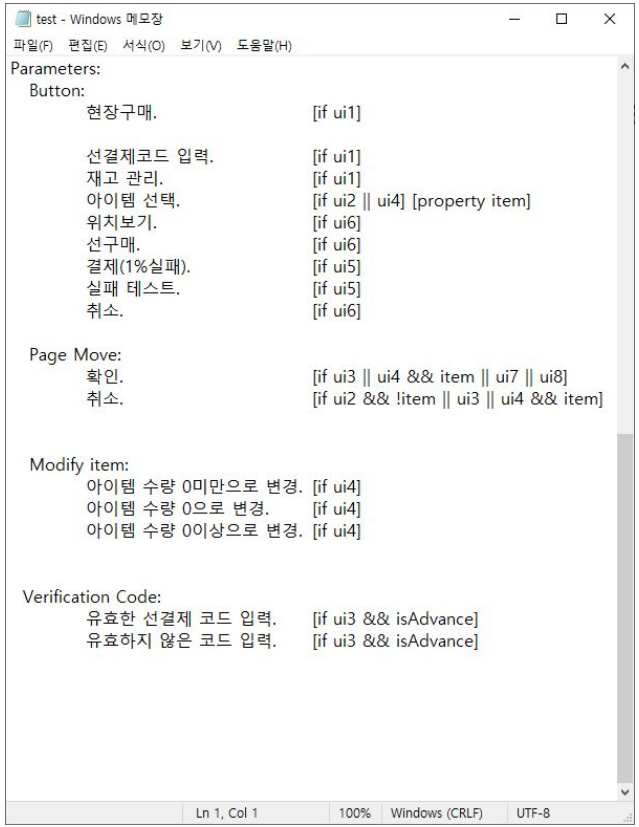


test - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
#
# Test specification for DVM
#

Environments:
    Current UI:
            initial page.                   [property ui1]
            item select.                    [property ui2]
            input verificaion code.         [property ui3]
            modify sotck.                   [property ui4]
            payment.                        [property ui5]
            inform prepurchase.             [property ui6]
            inform location.                [property ui7]
            inform verification code.       [property ui8]

    선구매 여부:
            선구매 진행.                    [if ui3 || ui5 || ui7 || ui8] [property
isAdvance]
            선구매 하지않음.                [if ui5]

    선택한 아이템의 재고:
            local stock 0 & remote stock 0.         [if ui2 || ui3 || ui4]
            local stock 0 & remote stock > 0.       [if ui2 || ui3 || ui4]
            local stock > 0.                        [if ui2 || ui3 || ui4]

    판매하는 아이템 가짓수:
            7개.                            [if ui4]
            7개 미만.                       [if ui4]

    선구매코드 broadcast:
            Success.            [if isAdvance]
            Fail.               [error]
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8



test - Windows 메모장

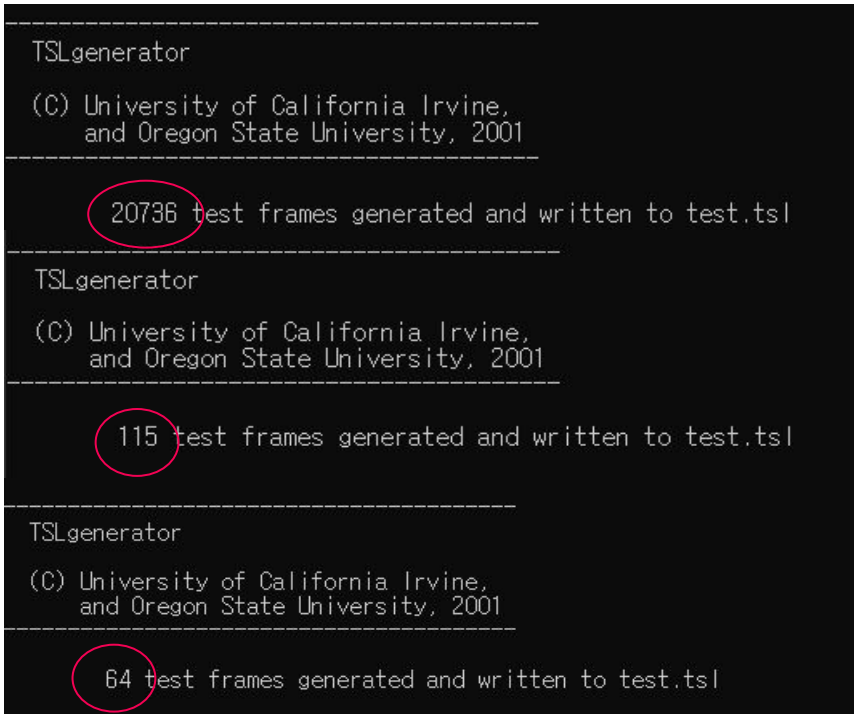
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Parameters:
  Button:
            현장구매.                     [if ui1]

            선결제코드 입력.              [if ui1]
            재고 관리.                    [if ui1]
            아이템 선택.                  [if ui2 || ui4] [property item]
            위치보기.                     [if ui6]
            선구매.                       [if ui6]
            결제(1%실패).                 [if ui5]
            실패 테스트.                  [if ui5]
            취소.                         [if ui6]

  Page Move:
            확인.                         [if ui3 || ui4 && item || ui7 || ui8]
            취소.                         [if ui2 && !item || ui3 || ui4 && item]


  Modify item:
            아이템 수량 0미만으로 변경.   [if ui4]
            아이템 수량 0으로 변경.       [if ui4]
            아이템 수량 0이상으로 변경.   [if ui4]


  Verification Code:
            유효한 선결제 코드 입력.      [if ui3 && isAdvance]
            유효하지 않은 코드 입력.      [if ui3 && isAdvance]
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

02 System Testing (2nd) - CPT

TSL Generator 사용하여 Test Case 생성



Test Case 생성

Property constraint 적용전 ->20736개

Property constraint 적용-> 115개

Property constraint 추가, error marking 적용 -> 64개

02 System Testing (2nd) - CPT

Constraint 적용 & CPT Result

Num	Description	result
1	선구매코드가 broadcast되지 않은 경우	pass
2	초기화면에서 현장구매 버튼을 선택한경우 음료 선택화면으로 진입	pass
3	초기화면에서 선결제코드 입력 버튼을 선택한경우 인증번호 입력화면으로 진입	pass
4	초기화면에서 재고관리 버튼을 선택한경우 음료 선택화면으로 진입	pass
5	아이템선택화면에서 local stock과 remote stock 모두 재고가 없는 아이템 선택한경우 품절입니다 메시지창 생성	pass
6	아이템선택화면에서 local stock 0 remote stock 1 이상인경우 선구매화면 진입	pass
7	아이템선택화면에서 local stock 1이상인경우 결제화면으로 진입	pass
8	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가0으로 떨어진 상태에서 유효한 선결제 코드를 입력후 확인선택	Fail 선택한 음료의 재고가 없음에도 선결제코드를 입력하면 음료 배출
9	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가0으로 떨어진 상태에서 유효하지 않은 선결제코드 입력하고 확인선택	pass
10	선구매 진행후 선구매코드가 broadcast되고 선택	pass
	한 아이템의 전체 재고가0으로 떨어진 상태에서 유효한 선결제 코드를 입력후 취소선택	
11	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가0으로 떨어진 상태에서 유효하지 않은 선결제코드 입력하고 취소선택	pass
12	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 현재자판기의 재고가 0으로 떨어진 경우 유효한 선결제코드를 입력하고 확인 선택	Fail 선택한 음료의 재고가 없음에도 선결제코드를 입력하면 음료 배출
13	선구매 진행후 선택한 선구매코드가 broadcast되고 아이템의 현재자판기의 재고가 0으로 떨어진 경우 유효하지 않은 선결제코드를 입력하고 확인 선택	pass
14	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 현재자판기의 재고가 0으로 떨어진 경우 유효한 선결제코드를 입력하고 취소 선택	pass
15	선구매 진행후 선택한 선구매코드가 broadcast되고 아이템의 현재자판기의 재고가 0으로 떨어진 경우 유효하지 않은 선결제코드를 입력하고 취소 선택	pass
16	선구매 진행후 선택한 선구매코드가 broadcast되고 아이템의 현재자판기의 재고가 1이상인 경우 유효한 선결제코드를 입력하고 확인 선택	pass
17	선구매 진행후 선택한 선구매코드가 broadcast되고 아이템의 현재자판기의 재고가 1이상인 경우 유효하지 않은 선결제코드를 입력하고 확인 선택	pass
18	선구매 진행후 선택한 선구매코드가 broadcast되고 아이템의 현재자판기의 재고가 1이상인 경우 유효한 선결제코드를 입력하고 취소 선택	pass
19	선구매 진행후 선택한 선구매코드가 broadcast되고 아이템의 현재자판기의 재고가 1이상인 경우 유효하지 않은 선결제코드를 입력하고 취소 선택	pass
20	재고관리에서 판매하고 있는 아이템이 7가지인경우 전체재고가 0인아이템의 수량을 0미만으로 변경하는경우	pass
21	재고관리에서 판매하고 있는 아이템이 7가지인경우 전체재고가 0인아이템의 수량을 0으로 변경하는경우	pass
22	재고관리에서 판매하고 있는 아이템이 7가지인경	Fail
	우 전체재고가 0인아이템의 수량을 0이상으로 변경하는경우	1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
23	재고관리에서 판매하고 있는 아이템이 7가지인경우 전체재고가 0인아이템의 수량을 0미만으로 설정하고 취소선택하는 경우	pass
24	재고관리에서 판매하고 있는 아이템이 7가지인경우 전체재고가 0인아이템의 수량을 0으로 설정하고 취소선택하는 경우	pass
25	재고관리에서 판매하고 있는 아이템이 7가지인경우 전체재고가 0인아이템의 수량을 0이상으로 설정하고 취소선택하는 경우	pass
26	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 전체 재고가 0인 아이템의 수량을 0미만으로 변경하는 경우	pass
27	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 전체 재고가 0인 아이템의 수량을 0으로 변경하는 경우	pass
28	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 전체 재고가 0인 아이템의 수량을 0이상으로 변경하는 경우	Fail 1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
29	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 전체 재고가 0인 아이템의 수량을 0미만으로 설정하고 취소 선택	pass
30	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 전체 재고가 0인 아이템의 수량을 0으로 설정하고 취소 선택	pass
31	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 전체 재고가 0인 아이템의 수량을 0이상으로 설정하고 취소 선택	pass
32	재고관리에서 판매하고 있는 아이템이 7가지 인경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0미만으로 변경하는 경우	pass
33	재고관리에서 판매하고 있는 아이템이 7가지 인경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0으로 변경하는 경우	pass

test case 56개 pass

02 System Testing (2nd) - CPT

Constraint 적용 & CPT Result

34	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0이상으로 변경하는 경우	Fail 1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
35	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0미만으로 설정하고 취소 선택	pass
36	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0으로 설정하고 취소 선택	pass
37	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0이상으로 설정하고 취소 선택	pass
38	재고관리에서 판매하고 있는 아이템이 7미만 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0미만으로 변경하는 경우	pass
39	재고관리에서 판매하고 있는 아이템이 7가지 미만인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0으로 변경하는 경우	pass
40	재고관리에서 판매하고 있는 아이템이 7미만 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0이상 변경하는 경우	Fail 1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
41	재고관리에서 판매하고 있는 아이템이 7미만 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0미만으로 설정하고 취소선택	pass
42	재고관리에서 판매하고 있는 아이템이 7미만 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0으로 설정하고 취소선택	pass
43	재고관리에서 판매하고 있는 아이템이 7미만 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0이상으로 설정하	pass

	고 취소선택	
44	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 미만으로 변경하는 경우	pass
45	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 으로으로 변경하는 경우	pass
46	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 이상으로 변경하는 경우	Fail 1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
47	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 미만으로 설정하고 취소 선택	pass
48	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 으로 설정하고 취소 선택	pass
49	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 이상으로 설정하고 취소 선택	pass
50	재고관리에서 판매하고 있는 아이템이 7가지 일 때 현재자판기에 재고가 있는 아이템의 수량을 0 미만으로 변경하는 경우	pass
51	재고관리에서 판매하고 있는 아이템이 7가지 미만 일때 현재자판기에 재고가 있는 아이템의 수량을 0으로 변경하는 경우	pass
52	재고관리에서 판매하고 있는 아이템이 7가지 미만 일때 현재자판기에 재고가 있는 아이템의 수량을 0이상으로 변경하는 경우	Fail 1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
53	재고관리에서 판매하고 있는 아이템이 7가지 미만 일때 현재자판기에 재고가 있는 아이템의 수량을 0미만으로 설정하고 취소하는 경우	pass
54	재고관리에서 판매하고 있는 아이템이 7가지 미만 일때 현재자판기에 재고가 있는 아이템의 수량을 0으로 설정하고 취소하는 경우	pass
55	재고관리에서 판매하고 있는 아이템이 7가지 미만 일때 현재자판기에 재고가 있는 아이템의 수량을 0이상으로 설정하고 취소하는 경우	pass

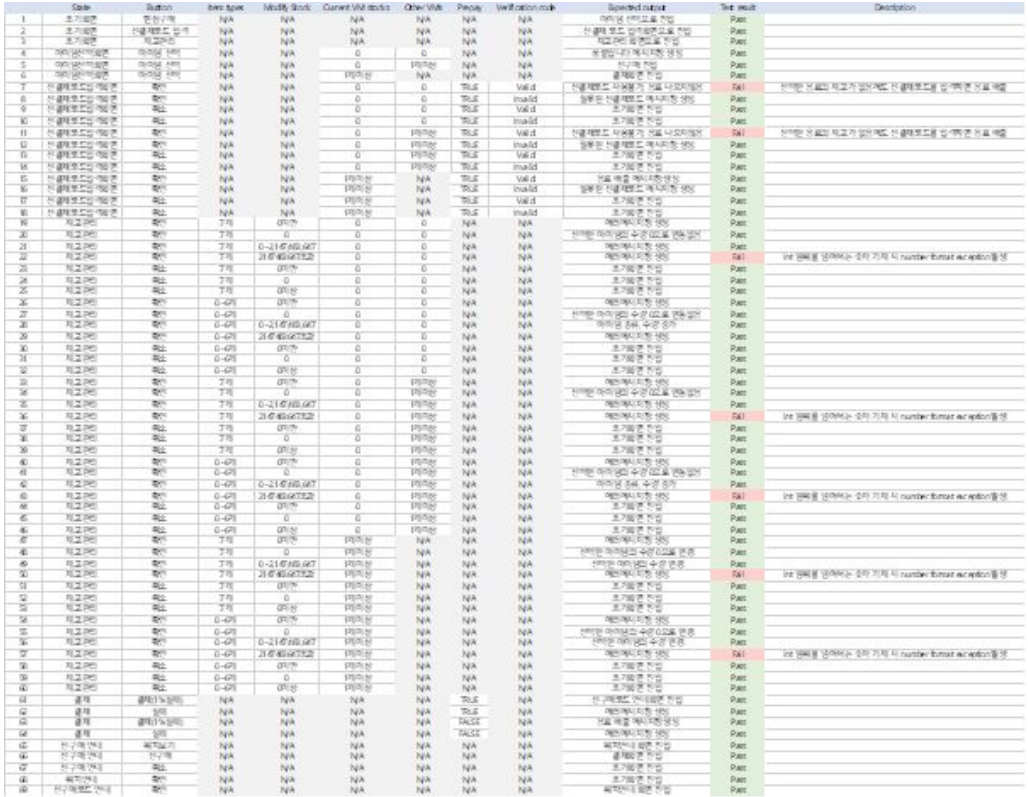
56	선구매후 결제(1%실패)를 진행하는 경우	pass
57	선구매후 실패테스트를 진행하는 경우	pass
58	선구매 하지않고 결제(1%실패)를 진행하는 경우	pass
59	선구매 하지않고 실패테스트를 진행하는 경우	pass
60	선구매 안내에서 위치보기를 선택하는 경우	pass
61	선구매 안내에서 선구매를 선택하는 경우	pass
62	선구매 안내에서 취소를 선택하는 경우	pass
63	위치안내 화면에서 확인을 선택하는 경우	pass
64	선구매코드 안내 화면에서 확인을 선택하는 경우	pass

56/64 = 87.5% 성공

주로 int 값을 입력할 때 최대치를 정해두지 않아 오버플로우가 발생한 부분에서 Fail이 일어남.

02 System Testing (2nd) - CPT

Data Specification



56/64 = 87.5% 성공

주로 int 값을 입력할 때 최대치를 정해두지 않아 오버플로우가 발생한 부분에서 Fail이 일어남.

02 System Testing (2nd) - BFT

- 3학년 개발팀(B1)이 정의한 기능(OOPT 2063)을 구체화하고 실제 프로그램 실행하며 test case 추가
- 1차 검증에서 테스트했던 불필요한 항목 1개 제거

Use case	No	Test case	1th Test Result	2th Test Result
Set up	1-1	시스템이 시작되었을때 다른 DVM의 ID를 받아왔는지 확인	PASS	PASS
	1-2	시스템이 시작되었을때 다른 DVM의 재고를 받아왔는지 확인	PASS	PASS
	1-3	시스템이 시작되었을 때 자판기의 재고가 업데이트 되었는지 확인	PASS	PASS
Select mode	2-1	USER가 선택한 모드에 따라 정해진 메뉴가 실행되는지 확인	PASS	PASS
Manage Stock	3-1	Manager가 갱신한 재고 정보가 적용되는지 확인	PASS	FAIL
	3-2	Manager가 총 8가지 이상의 음료를 갱신할 경우 에러가 나오는지 확인	PASS	PASS
Show item	4-1	모든 음료가 나오는지 확인	PASS	PASS
	4-2	User가 선택한 음료가 선택되는지 확인	FAIL	PASS
Check stock count	5-1	User가 선택한 음료에 대해 현재 자판기에서 판매 가능한지 확인	PASS	PASS
	5-2	현재 자판기에서 판매 가능하다면 Purchase를 정상적으로 실행하는지 확인	PASS	PASS
	5-3	통신을 통해 다른 자판기로부터 재고를 받아오는지 확인	PASS	PASS
	5-4	다른 자판기로부터 재고가 있다는 응답을 받으면, "Select Advance Payment"를 정상적으로 실행하는지 확인	PASS	PASS
Update stock	6-1	음료 재고정보를 주어진 값에 맞게 최신화 하는지 확인	FAIL	PASS
Purchase	7-1	결제가 성공했을 때 음료 재고를 맞게 줄이는지 확인	PASS	PASS
	7-2	결제가 성공했을 해당 음료를 배출하는지 확인	PASS	PASS
	7-3	결제가 실패했을 결제 실패 메시지를 보여주는지 확인	PASS	PASS
Advance purchase	8-1	선구매결제에 성공했을 때, 선구매코드를 받아오는지 확인	PASS	PASS
	8-2	선구매 결제에 성공했을 때, 받아온 선구매코드를 "Message Request"통해 "Broadcast"하는지 확인	FAIL	PASS
	8-3	선구매 결제에 성공했을 때, 해당 음료가 있는 자판기에 대해 "Inform Location"을 실행하는지 확인	PASS	PASS
	8-4	선구매 결제에 성공했을 때, 생성된 선구매코드를 보여주는지 확인	PASS	PASS
	8-5	선구매 결제에 실패했을 때, 결제 실패 메시지를 보여주는지 확인	PASS	PASS
	8-6	선구매 결제에 실패했을 때, 감소시켰던 재고 수량을 다시 증가시키는지 확인	FAIL	PASS
	8-7	선구매 진행 후 선구매 코드를 현재 자판기에 저장하는지 확인	PASS	PASS
	8-8	선구매 코드 입력 결제 시, 해당 item의 재고가 줄어드는지 확인	FAIL	PASS
Check payment	9-1	결제 결과를 받아오는지 확인	PASS	PASS
	9-2	정상적인 결제 카드가 아니라면 "잘못된 카드"라는 에러 메시지를 보여주는지 확인	PASS	PASS
Select advance payment	10-1	선구매를 선택한 음료에 대해 "Message Request"로 재고를 받아오는지 확인	PASS	PASS
	10-2	선구매결정한 음료의 재고가 남아있다면 "Message Request"로 재고를 감소시키는지 확인	FAIL	PASS
	10-3	재고를 감소시킨 후 "Advance Purchase"를 실행하는지 확인	FAIL	PASS
	10-4	선구매 결정한 음료의 재고가 없다면 품절메시지를 보여주고, "Select Mode"로 돌아가는지 확인	PASS	PASS
	10-5	위치확인 모드를 선택한 음료에 대해 "Message Reauest"로 재고를 받아오는지 확인	PASS	PASS
	10-6	위치확인 모드를 선택한 음료의 재고가 남아있다면 해당 음료에 대해 "Inform Location"을 실행하는지 확인	PASS	PASS
	10-7	위치확인 모드를 선택한 음료의 재고가 남아있지 않다면 품절메시지를 보여주고, "Select Mode"로 돌아가는지 확인	PASS	PASS
	10-8	선구매 진행을 위한 결제시 다른 자판기들의 재고가 바로 줄어드는지 확인	FAIL	PASS
	10-9	선결제 후 선구매 코드 입력시 해당 item의 재고가 줄어드는지 확인	FAIL	PASS
Inform location	11-1	음료를 갖고있는 자판기에 대해 위치를 보여주는지 확인	PASS	PASS
Create verification code	12-1	음료가 선택되었을 때, 새로운 선구매 코드를 생성하는지 확인	PASS	PASS
	12-2	"Create Verification Code"를 통해 생성된 선구매 코드를 보여주는지 확인	PASS	PASS
	12-3	선결제 코드가 여러종류의 음료에 중복되어 생성되지 않는지 확인	FAIL	PASS
Item out	13-1	음료가 선택되어있고, 결제가 완료되었거나 유효한 선구매 코드를 입력했을때만 음료를 제공하는지 확인	FAIL	PASS
	13-2	현재 자판기에 재고가 없는데 선구매 코드가 있다는 이유로 item out이 되는지 확인	FAIL	PASS
Read verification code	14-1	User가 입력한 선구매 코드를 "Check Verification Code"에 전달하고 그 결과를 받는지 확인	PASS	PASS
	14-2	기록된 코드가 유효하다면 "Item Out"을 실행하는지 확인	PASS	FAIL
	14-3	"Item Out"을 실행한 후에 "Reset Verification Code"를 실행하는지 확인	PASS	PASS
Check verification code	15-1	시스템에 입력된 코드가 존재하는지 확인	PASS	PASS
Reset verification code	16-1	시스템에서 입력된 코드를 정상적으로 삭제하는지 확인	PASS	PASS
Message request	17-1	보내고자하는 메시지 종류를 수신 대상에게 잘보내는지 확인	FAIL	PASS
	17-2	수신된 메시지에 따라 명시된 응답을 전송한 DVM에 정상적으로 보내는지 확인	PASS	PASS
	17-3	알 수 없는 종류에 메시지를 수신한 경우 알 수 없음을 응답하는지 확인	PASS	PASS
전체 Test Case : 49개			Pass : 47개	

- 49개의 test case 중에서 47개의 test case pass => 96% Pass
- 1차 검증 70%에 비해, 2차 검증은 26% 향상

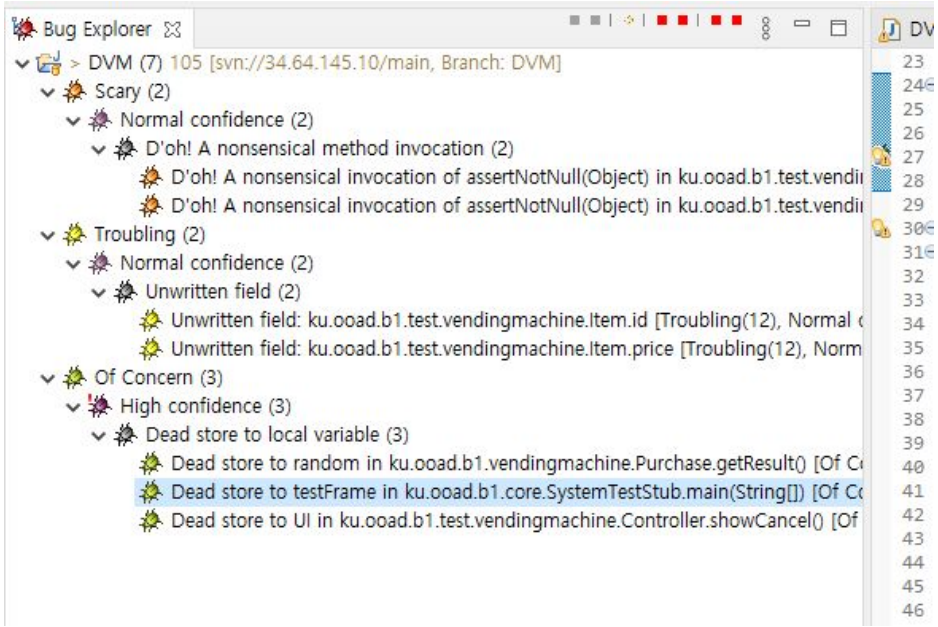
02 System Testing (2nd) - BFT

FAIL된 항목 2개의 Input & Environment

- No. 3-1
  - FAIL 사유
    - 음료의 종류가 7가지에서 6가지로 바뀌었을 때 새로운 종류의 음료 재고의 추가가 안됨
  - FAIL이 발생하는 Input & Environment
    - 콜라 1개, 사이다 0개 (음료 7가지) → 콜라 0개, 사이다 0개 (음료 6가지) → 콜라 0개, 사이다 1개 (음료 7가지) 가 가능해야 하는데, 사이다 추가 불가능
- No. 14-2
  - FAIL 사유
    - 재고가 0인 음료에 대하여, 선결제 구매 코드를 생성한 뒤, 재고를 추가하면 재고를 추가한 VM에서도 선결제 구매 코드를 통해 구매할 수 있어야 하는데 불가능
  - FAIL이 발생하는 Input & Environment
    - VM0에 콜라 0개, VM1에 콜라 2개 있다. → VM0에서 콜라 구매를 시도하여 선결제코드 생성 후 broadcast → VM0의 선결제코드 목록은 비어있으나, VM1의 선결제코드 목록은 콜라의 코드가 추가됨 → VM0에서 콜라의 재고를 추가함 → VM0에서도 선결제 코드를 통해 콜라를 구매할 수 있어야 하나, 구매 불가능

03 Static Analysis - Bugs

FindBugs로 버그 측정 결과

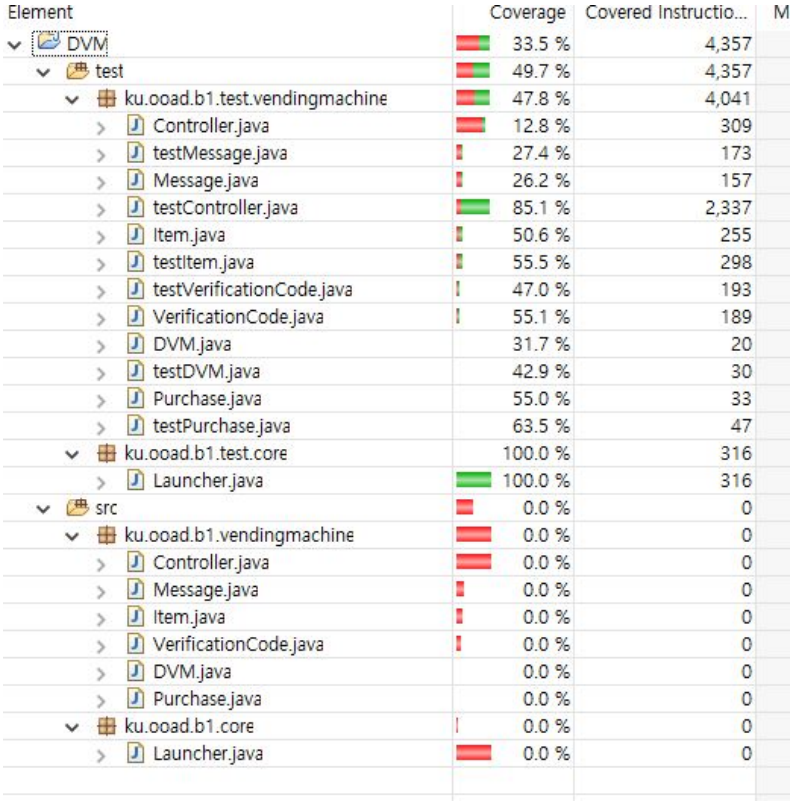


Fatal한 Error는 없으며 전부 만들어졌으나 사용하지 않은 필드, 변수가 있어 측정된 code smell이다.

Scary(주의)에는 assertNotNull이란 메소드에 인자가 String이 와서 검출되었다. Java에선 String이 Object이기 때문에 큰 문제는 아니다.

03 Static Analysis - Code Coverage

MC/DC Coverage 측정 결과



Element	Coverage	Covered Instructio...	M
DVM	33.5 %	4,357	
test	49.7 %	4,357	
ku.ooad.b1.test.vendingmachine	47.8 %	4,041	
Controller.java	12.8 %	309	
testMessage.java	27.4 %	173	
Message.java	26.2 %	157	
testController.java	85.1 %	2,337	
Item.java	50.6 %	255	
testItem.java	55.5 %	298	
testVerificationCode.java	47.0 %	193	
VerificationCode.java	55.1 %	189	
DVM.java	31.7 %	20	
testDVM.java	42.9 %	30	
Purchase.java	55.0 %	33	
testPurchase.java	63.5 %	47	
ku.ooad.b1.test.core	100.0 %	316	
Launcher.java	100.0 %	316	
src	0.0 %	0	
ku.ooad.b1.vendingmachine	0.0 %	0	
Controller.java	0.0 %	0	
Message.java	0.0 %	0	
Item.java	0.0 %	0	
VerificationCode.java	0.0 %	0	
DVM.java	0.0 %	0	
Purchase.java	0.0 %	0	
ku.ooad.b1.core	0.0 %	0	
Launcher.java	0.0 %	0	

테스트를 위한 클래스와 메인 프로그램 실행에 대한 구분이 안되있어 실질적인 커버리지는 0% 였으며, 유닛테스트 폴더내의 커버리지도 49.7%로 상당히 미달됨을 확인할 수 있었습니다.

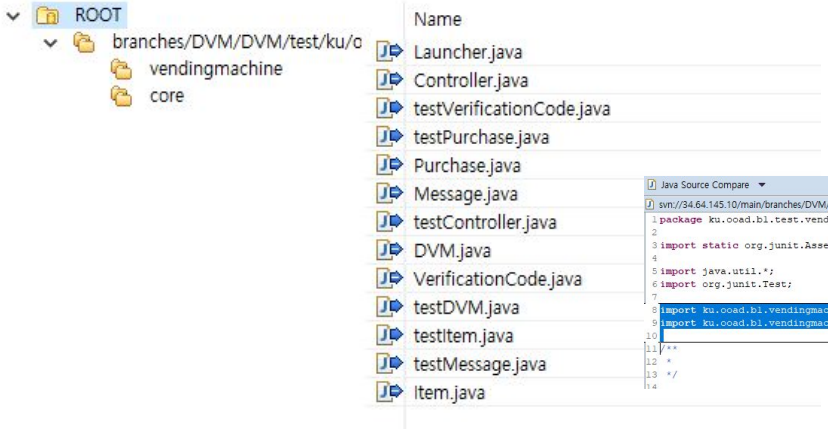
03 Static Analysis - Code Coverage

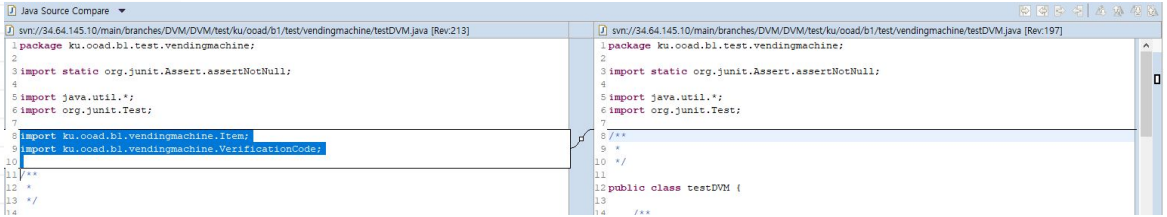
MC/DC Coverage 측정 결과

DVM

R.	D.	...	A.	Comment
	2...	1	c...	[no comme...
	2...	...	c...	코드 커버리...
	2...	2	c...	junit4로 변...
	2...	1	t...	[no comme...
	2...	1	t...	[no comme...
	2...	2	t...	[no comme...
	2...	1	t...	[no comme...
	2...	1	t...	bb
	2...	4	t...	[no comme...

코드 커버리지 계산을 위한 수정



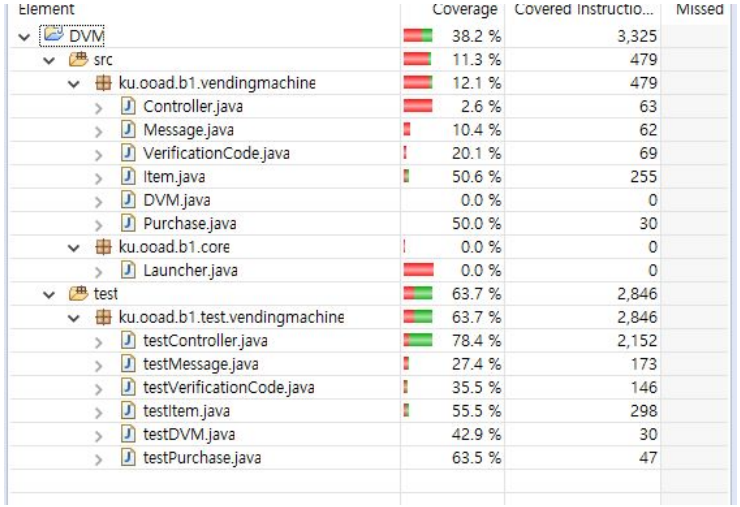


실질적인 커버리지를 올리기위하여 메인클래스들과 겹치게 생성되있는 테스트폴더내의 클래스를 모두 삭제하고

동시에 메인과 유닛테스트클래스를 연결해주는 임포트를 추가하였습니다.

03 Static Analysis - Code Coverage

MC/DC Coverage 측정 결과

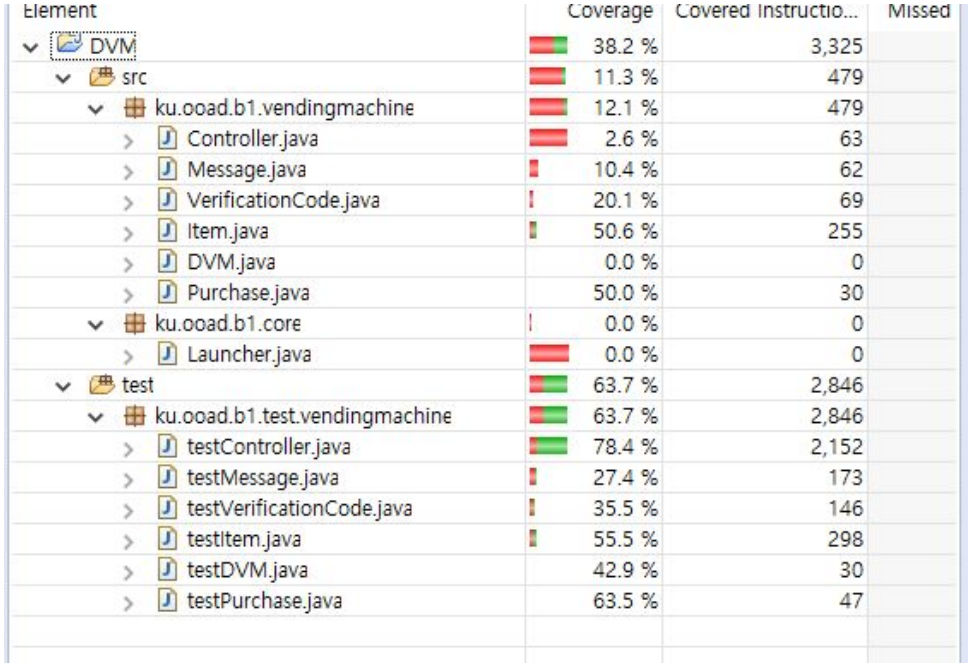


Element	Coverage	Covered Instructio...	Missed
DVM	38.2 %	3,325	
src	11.3 %	479	
ku.ooad.b1.vendingmachine	12.1 %	479	
Controller.java	2.6 %	63	
Message.java	10.4 %	62	
VerificationCode.java	20.1 %	69	
Item.java	50.6 %	255	
DVM.java	0.0 %	0	
Purchase.java	50.0 %	30	
ku.ooad.b1.core	0.0 %	0	
Launcher.java	0.0 %	0	
test	63.7 %	2,846	
ku.ooad.b1.test.vendingmachine	63.7 %	2,846	
testController.java	78.4 %	2,152	
testMessage.java	27.4 %	173	
testVerificationCode.java	35.5 %	146	
testItem.java	55.5 %	298	
testDVM.java	42.9 %	30	
testPurchase.java	63.5 %	47	

앞선 솔루션으로 달라진 커버리지는
다음과같이 전체 커버리지가 38.2%, 메인클래스들의
커버리지가 11.3%로 증가한 것을 확인할 수 있습니다.

03 Static Analysis - Code Coverage

Decision/Branch Coverage 측정 결과



Element	Coverage	Covered Instructio...	Missed
DVM	38.2 %	3,325	
src	11.3 %	479	
ku.ooad.b1.vendingmachine	12.1 %	479	
Controller.java	2.6 %	63	
Message.java	10.4 %	62	
VerificationCode.java	20.1 %	69	
Item.java	50.6 %	255	
DVM.java	0.0 %	0	
Purchase.java	50.0 %	30	
ku.ooad.b1.core	0.0 %	0	
Launcher.java	0.0 %	0	
test	63.7 %	2,846	
ku.ooad.b1.test.vendingmachine	63.7 %	2,846	
testController.java	78.4 %	2,152	
testMessage.java	27.4 %	173	
testVerificationCode.java	35.5 %	146	
testItem.java	55.5 %	298	
testDVM.java	42.9 %	30	
testPurchase.java	63.5 %	47	

MC/DC Coverage를 시행한후 같은 조건에서 어디까지 커버가 되는지 확인하였는데,

기존에 생성되있었던 유닛테스트 함수들이 모두 미흡하여, 테스트함수끼리의 호출과 불필요한 절차지향식 코드, 객체지향적이지 못하여 테스트함수가 생성되지 못한 점들이 많아 커버리지를 최대한 올리지 못하였습니다.

03 Static Analysis - Code Coverage

Decision/Branch Coverage 측정 결과

[Code Coverage-Decision/Branch Coverage] 측정 결과

in list 이슈 현황

LABELS

심각도 보통 중요 +

Description Edit

코드커버리지가 40퍼를 넘지 못하기때문에 최소한 50퍼는 넘게 수정부탁드립니다.
일부기능을 함수화 하여 test클래스들에서 호출하는 방식을 차용하시길 바랍니다.

Attachments

coverage_3.png ↗
Added 6 hours ago - Comment - Delete - Edit
Make cover

coverage_2.png ↗
Added 6 hours ago - Comment - Delete - Edit
Make cover

coverage_1.png ↗
Added 6 hours ago - Comment - Delete - Edit
Make cover

Add an attachment

이를 3학년팀에 트렐로로 이슈를 등록하여 수정하게끔 요청 진행

Launcher.java

```
24      List<Item> items = new ArrayList<>();
25      List<DVM> dvms = new ArrayList<>();
26      for(int i = 0; i < 20; i++) {
27          items.add(null);
28      }
29      for(int i = 0; i < 5; i++) {
30          dvms.add(null);
31      }
32
33      /* 재고 설정 */
34      for(int i = 0; i < 5; i++) {          함수화 필요함 >> JUnit Test 추가
35          int totalCount = 0;
36          Random rnd = new Random();
37          while(totalCount < 7) {
38              int cur = rnd.nextInt(20);
39              if(items.get(cur) != null) {
40                  if(items.get(cur).getCount(i) != null) {
41                      continue;
42                  }
43                  else {
44                      //다른자판기에 있는거임
45                      int count = rnd.nextInt(9)+1;
46                      Item item = items.get(cur);
47                      item.setCount(i, count);
48                      totalCount++;
49                      System.out.println(i + "에 " + Item.names[cur] + " 을(를) " + count + "만큼 설정");
50                  }
51              } else {
52                  int count = rnd.nextInt(9)+1;
53                  Item item = new Item(cur, count, Item.prices[cur], i);
54                  /* FOR STUB */
55                  items.set(cur, item);
56                  System.out.println(i + "에 " + Item.names[cur] + " 을(를) " + count + "만큼 설정");
57                  /* FOR STUB */
58                  totalCount++;
59              }
60          }
61      }
62      /* 재고 설정 */
63      for(int i = 0; i < dvms.size(); i++) {
64          dvms.set(i, new DVM(i, locations[i], items));
65      }
```

03 Static Analysis - Code Coverage

Condition Coverage 측정 결과 - 1) 특정 배열을 비웠을때의 Coverage

```
public static String names[] = {"클라","사이다", "환타", "스프라이트", "펩시",
        "마운틴 듀", "파워 에이드", "2프로", "포카리 스웨트", "게토레이",
        "아이시스", "삼다수", "레쓰비", "T.O.P", "비타500",
        "데자와", "솔의 눈","아침햇살", "ZICO", "맥콜"};

public static Integer prices[] = {};
```

Element	Coverage	Covered Instructio...	Missed Instructions
DVM	34.4 %	2,956	5,626
src	11.3 %	478	3,756
ku.ooad.b1.vendingmachine	12.1 %	478	3,487
Controller.java	2.6 %	63	2,333
Message.java	10.2 %	61	538
VerificationCode.java	20.1 %	69	274
Item.java	50.6 %	255	249
DVM.java	0.0 %	0	63
Purchase.java	50.0 %	30	30
ku.ooad.b1.core	0.0 %	0	269
Launcher.java	0.0 %	0	269
test	57.0 %	2,478	1,870
ku.ooad.b1.test.vendingmachine	57.0 %	2,478	1,870
testController.java	72.3 %	1,971	754
testMessage.java	27.4 %	173	458
testItem.java	25.4 %	111	326
testVerificationCode.java	35.5 %	146	265
testDVM.java	42.9 %	30	40
testPurchase.java	63.5 %	47	27

위와 같이 가격들을 비우고 장소들을 비웠을 때에 대한 Coverage를 재보았는데

왼쪽과 같은 커버리지를 확인할 수 있었고, 이에 대한 코드를 분석 진행함.

03 Static Analysis - Code Coverage

Condition Coverage 측정 결과 - 1) 특정 배열을 비웠을때의 Coverage

```
386         String[] alpha = {"0", "1", "2", "3", "4"};
387         assertThat(dvms, is(alpha));
388
389         if(dvms != null) {
390             String[] locations = {"신공학관 1층", "새천년관 1층", "학생회관 1층", "법학관 1층", "도서관 1층"};
391             for(int i = 0; i < dvms.length; i++) {
392                 Integer dvmId = Integer.parseInt(dvms[i]);
393                 dvmLocations += locations[dvmId];
394
395                 if(i < dvms.length-1)
396                     dvmLocations += ",";
397             }
398         } else {
399             dvmLocations = "없음";
400         }
401         assertNotNull(dvmLocations);
402
403         //      System.out.println("dvm location: " + this.dvm.get(dvm_id).getLocation());
404         locationSet.setText(dvmLocations);
405         ok = new JButton("확인");
406
407         underLayer1.add(locate);
408         underLayer2.add(locationSet);
409         underLayer3.add(ok);
410
411         showLocation.add(underLayer1);
412         showLocation.add(underLayer2);
413         showLocation.add(underLayer3);
414
```

배열이 비어있음에 따라 적절하게 커버리지가 빠지는것을 볼 수 있었음.

```
384         }
385
386         String[] alpha = {"0", "1", "2", "3", "4"};
387         assertThat(dvms, is(alpha));
388
389         if(dvms != null) {
390             String[] locations = {};
391             for(int i = 0; i < dvms.length; i++) {
392                 Integer dvmId = Integer.parseInt(dvms[i]);
393                 dvmLocations += locations[dvmId];
394
395                 if(i < dvms.length-1)
396                     dvmLocations += ",";
397             }
398         } else {
399             dvmLocations = "없음";
400         }
401         assertNotNull(dvmLocations);
402
403         //      System.out.println("dvm location: " + this.dvm.get(dvm_id).getLocation());
404         locationSet.setText(dvmLocations);
405         ok = new JButton("확인");
406
407         underLayer1.add(locate);
408         underLayer2.add(locationSet);
409         underLayer3.add(ok);
410
411         showLocation.add(underLayer1);
412         showLocation.add(underLayer2);
413         showLocation.add(underLayer3);
414
415         UI.add(showLocation);
```

03 Static Analysis - Code Coverage

Condition Coverage 측정 결과 - 1) 특정 배열을 비웠을때의 Coverage

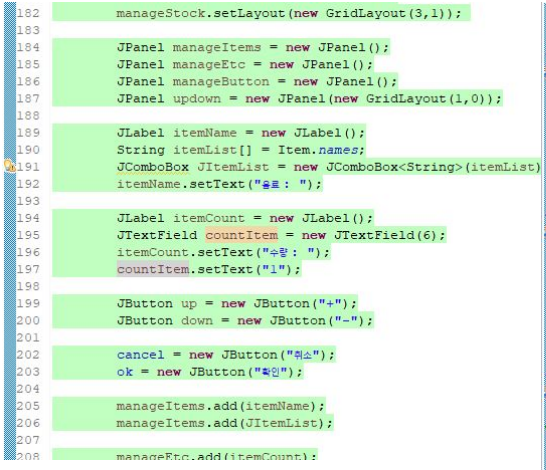
```
23
24    public static Integer prices[] = {900, 900, 700, 700, 800,
25            700, 1200, 900, 1200, 1200,
26            600, 800, 500, 1200, 500,
27            600, 600, 1500, 700, 700};
28
29    public testItem() {
30        this.id = new Random().nextInt(5);
31        Integer dvmId = new Random().nextInt(5);
32        Integer count = new Random().nextInt(9);
33        Integer price = prices[id];
34        setName(testItem.names[id]);
35
36        this.price=price;
37        if(this.price == 0)
38            this.price = testItem.prices[id];
39
40        if(this.count == null)
41            this.count=new HashMap<Integer, Integer>();
42        this.count.put(dvmId, count);
43    }
44
```

특정 Condition에 따라 명확하게 커버리지가 변화하는것을 볼 수 있었으나 일부 코드들이 메소드화가 안되있어서 Condition에 따라 이후의 커버리지가 전혀 되지 않고 있음을 확인할 수 있었음.

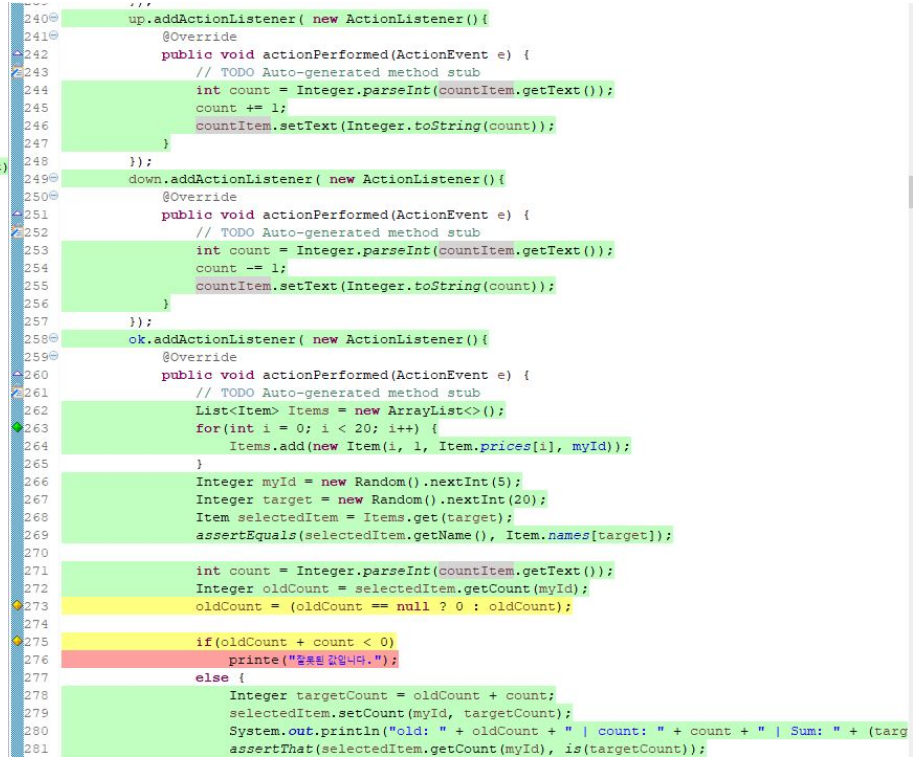
```
23
24    public static Integer prices[] = {};
25
26    public testItem() {
27        this.id = new Random().nextInt(5);
28        Integer dvmId = new Random().nextInt(5);
29        Integer count = new Random().nextInt(9);
30        Integer price = prices[id];
31        setName(testItem.names[id]);
32
33        this.price=price;
34        if(this.price == 0)
35            this.price = testItem.prices[id];
36
37        if(this.count == null)
38            this.count=new HashMap<Integer, Integer>();
39        this.count.put(dvmId, count);
40    }
41
42    /**
43     *
44     */
```

03 Static Analysis - Code Coverage

Condition Coverage 측정 결과 - 2) 수량을 int범위 이상으로 인풋하였을 경우의 Coverage



```
182         manageStock.setLayout(new GridLayout(3,1));
183
184         JPanel manageItems = new JPanel();
185         JPanel manageEtc = new JPanel();
186         JPanel manageButton = new JPanel();
187         JPanel updown = new JPanel(new GridLayout(1,0));
188
189         JLabel itemName = new JLabel();
190         String itemList[] = Item.names;
191         JComboBox JItemList = new JComboBox<String>(itemList)
192         itemName.setText("물품 : ");
193
194         JLabel itemCount = new JLabel();
195         JTextField countItem = new JTextField(6);
196         itemCount.setText("수량 : ");
197         countItem.setText("1");
198
199         JButton up = new JButton("+");
200         JButton down = new JButton("-");
201
202         cancel = new JButton("취소");
203         ok = new JButton("확인");
204
205         manageItems.add(itemName);
206         manageItems.add(JItemList);
207
208         manageEtc.add(itemCount);
```



```
240         up.addActionListener( new ActionListener(){
241             @Override
242             public void actionPerformed(ActionEvent e) {
243                 // TODO Auto-generated method stub
244                 int count = Integer.parseInt(countItem.getText());
245                 count += 1;
246                 countItem.setText(Integer.toString(count));
247             }
248         });
249         down.addActionListener( new ActionListener(){
250             @Override
251             public void actionPerformed(ActionEvent e) {
252                 // TODO Auto-generated method stub
253                 int count = Integer.parseInt(countItem.getText());
254                 count -= 1;
255                 countItem.setText(Integer.toString(count));
256             }
257         });
258         ok.addActionListener( new ActionListener(){
259             @Override
260             public void actionPerformed(ActionEvent e) {
261                 // TODO Auto-generated method stub
262                 List<Item> Items = new ArrayList<>();
263                 for(int i = 0; i < 20; i++) {
264                     Items.add(new Item(i, 1, Item.prices[i], myId));
265                 }
266                 Integer myId = new Random().nextInt(5);
267                 Integer target = new Random().nextInt(20);
268                 Item selectedItem = Items.get(target);
269                 assertEquals(selectedItem.getName(), Item.names[target]);
270
271                 int count = Integer.parseInt(countItem.getText());
272                 Integer oldCount = selectedItem.getCount(myId);
273                 oldCount = (oldCount == null ? 0 : oldCount);
274
275                 if(oldCount + count < 0)
276                     printe("잘못된 값입니다.");
277                 else {
278                     Integer targetCount = oldCount + count;
279                     selectedItem.setCount(myId, targetCount);
280                     System.out.println("old: " + oldCount + " | count: " + count + " | Sum: " + (targ
281                     assertThat(selectedItem.getCount(myId), is(targetCount));
```

03 Static Analysis - Code Coverage

Condition Coverage 측정 결과 - 2) 수량을 int범위 이상으로 인풋하였을 경우의 Coverage

```
178    @Test
179    public void updateStockInfo() {
180        JFrame UI = new JFrame();
181        JPanel manageStock = new JPanel();
182        manageStock.setLayout(new GridLayout(3,1));
183
184        JPanel manageItems = new JPanel();
185        JPanel manageEtc = new JPanel();
186        JPanel manageButton = new JPanel();
187        JPanel updown = new JPanel(new GridLayout(1,0)
188
189        JLabel itemName = new JLabel();
190        String itemList[] = Item.names;
191        JComboBox JItemList = new JComboBox<String>(it
192        itemName.setText("음료 : ");
193
194        JLabel itemCount = new JLabel();
195        JTextField countItem = new JTextField(6);
196        itemCount.setText("수량 : ");
197        countItem.setText("10000000000000000000000000")
198
199        JButton up = new JButton("+");
200        JButton down = new JButton("-");
201
```

testControll... | testItem.java | testPurchas... | testMessage.... | testVerific... | testDVM.java

```
240        up.addActionListener( new ActionListener(){
241            @Override
242            public void actionPerformed(ActionEvent e) {
243                // TODO Auto-generated method stub
244                int count = Integer.parseInt(countItem.getText());
245                count += 1;
246                countItem.setText(Integer.toString(count));
247            }
248        });
249        down.addActionListener( new ActionListener(){
250            @Override
251            public void actionPerformed(ActionEvent e) {
252                // TODO Auto-generated method stub
253                int count = Integer.parseInt(countItem.getText());
254                count -= 1;
255                countItem.setText(Integer.toString(count));
256            }
257        });
258        ok.addActionListener( new ActionListener(){
259            @Override
260            public void actionPerformed(ActionEvent e) {
261                // TODO Auto-generated method stub
262                List<Item> Items = new ArrayList<>();
263                for(int i = 0; i < 20; i++) {
264                    Items.add(new Item(i, 1, Item.prices[i], myId));
265                }
266                Integer myId = new Random().nextInt(5);
267                Integer target = new Random().nextInt(20);
268                Item selectedItem = Items.get(target);
269                assertEquals(selectedItem.getName(), Item.names[target
270
271                int count = Integer.parseInt(countItem.getText());
272                Integer oldCount = selectedItem.getCount(myId);
273                oldCount = (oldCount == null ? 0 : oldCount);
274
275                if(oldCount + count < 0)
276                    printe("잘못된 값입니다.");
277                else {
278                    Integer targetCount = oldCount + count;
279                    selectedItem.setCount(myId, targetCount);
280                    System.out.println("old: " + oldCount + " | count: " + count + " | Sum: " + (targetCou
281                    assertThat(selectedItem.getCount(myId), is(targetCount));
```

Element	Coverage	Covered Instructio...	Missed Instructions
DVM	35.3 %	3,072	5,63
src	11.1 %	469	3,76
ku.ooad.b1.vendingmachine	11.8 %	469	3,49
Controller.java	2.6 %	63	2,33
Message.java	10.2 %	61	53
VerificationCode.java	20.1 %	69	27
Item.java	48.8 %	246	25
DVM.java	0.0 %	0	6
Purchase.java	50.0 %	30	3
ku.ooad.b1.core	0.0 %	0	26
Launcher.java	0.0 %	0	26
test	58.3 %	2,603	1,86
ku.ooad.b1.test.vendingmachine	58.3 %	2,603	1,86
testController.java	69.5 %	1,909	83
testMessage.java	27.4 %	173	45
testVerificationCode.java	35.5 %	146	26
testItem.java	55.5 %	298	23
testDVM.java	42.9 %	30	4
testPurchase.java	63.5 %	47	2

03 Static Analysis - Code Coverage

Statement Coverage 측정 결과 - 1) updateStockInfo

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
DVM	7.6 %	662	8,040	8,702
test	9.2 %	413	4,055	4,468
ku.ooad.b1.test.vendingmachine	9.2 %	413	4,055	4,468
testController.java	15.0 %	413	2,332	2,745
testMessage.java	0.0 %	0	631	631
testItem.java	0.0 %	0	537	537
testVerificationCode.java	0.0 %	0	411	411
testPurchase.java	0.0 %	0	74	74
testDVM.java	0.0 %	0	70	70
src	5.9 %	249	3,985	4,234
ku.ooad.b1.vendingmachine	6.3 %	249	3,716	3,965
Controller.java	0.0 %	0	2,396	2,396
Message.java	0.0 %	0	599	599
VerificationCode.java	0.0 %	0	343	343
Item.java	49.4 %	249	255	504
DVM.java	0.0 %	0	63	63
Purchase.java	0.0 %	0	60	60
ku.ooad.b1.core	0.0 %	0	269	269
Launcher.java	0.0 %	0	269	269

해당 statement가 제일 함수화가 덜 되어있었고 절차지향적이게 사용하고 있었기에 커버리지를 어느 컨트롤러 또는 클래스를 제외하고 하여야될지 의문이었습니다.

03 Static Analysis - Code Coverage

Statement Coverage 측정 결과 - 2) showMode

```
313     * @param mode
314     */
315    public void showMode() {
316        JFrame UI = new JFrame();
317        JPanel showMode = new JPanel();
318        showMode.setLayout(new FlowLayout());
319
320        JButton goShowItem = new JButton("현장 구매");
321        JButton goReadVcode = new JButton("선결제코드 입력");
322        JButton goManageStock = new JButton("재고 관리");
323
324        showMode.add(goShowItem);
325        showMode.add(goReadVcode);
326        showMode.add(goManageStock);
327
328        UI.add(showMode);
329        UI.setVisible(true);
330
331        boolean goshowClicked = false;
332        boolean goreadClicked = false;
333        boolean gomanageClicked = false;
334
335        goShowItem.addActionListener( new ActionListener(){
336            @Override
337            public void actionPerformed(ActionEvent e) {
338                testController.getInstance(myId).selectMode();
339            }
340        });
341        goReadVcode.addActionListener( new ActionListener(){
342            @Override
343            public void actionPerformed(ActionEvent e) {
344                testController.getInstance(myId).selectMode();
345            }
346        });
347        goManageStock.addActionListener( new ActionListener(){
348            @Override
349            public void actionPerformed(ActionEvent e) {
350                testController.getInstance(myId).selectMode();
351            }
352        });
353    }
```

testController.showMode (2021. 5. 31. 오후 10:17:53)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
DVM	0.0 %	0	8,702	8,702
test	0.0 %	0	4,468	4,468
ku.ooad.b1.test.vendingmachine	0.0 %	0	4,468	4,468
testController.java	0.0 %	0	2,745	2,745
testMessage.java	0.0 %	0	631	631
testItem.java	0.0 %	0	537	537
testVerificationCode.java	0.0 %	0	411	411
testPurchase.java	0.0 %	0	74	74
testDVM.java	0.0 %	0	70	70
src	0.0 %	0	4,234	4,234
ku.ooad.b1.vendingmachine	0.0 %	0	3,965	3,965
Controller.java	0.0 %	0	2,396	2,396
Message.java	0.0 %	0	599	599
Item.java	0.0 %	0	504	504
VerificationCode.java	0.0 %	0	343	343
DVM.java	0.0 %	0	63	63
Purchase.java	0.0 %	0	60	60
ku.ooad.b1.core	0.0 %	0	269	269
Launcher.java	0.0 %	0	269	269

테스트 클래스에는 테스트하는 함수만이 존재해야되나 의미없는 함수가 존재하였습니다. 해당 함수를 커버리지를 조사하였을때는 0퍼로 수렴하는것을 확인하였습니다.

03 Static Analysis - Code Coverage

Statement Coverage 측정 결과 - 3) showLocation

```
356     * @param dvm
357     * @return
358     */
359    @Test
360    public void showLocation() { // (DVM dvm)
361        //JFrame showLocation2 = new JFrame();
362        JFrame UI = new JFrame();
363        JPanel showLocation = new JPanel();
364        setTitle("showLocation");
365        setSize(370,450);
366        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
367
368        //Container c = getContentPane();
369        showLocation.setLayout(new GridLayout(0,1));
370
371        JPanel underLayer1 = new JPanel(new FlowLayout());
372        JPanel underLayer2 = new JPanel(new FlowLayout());
373        JPanel underLayer3 = new JPanel(new FlowLayout());
374
375        JLabel locate = new JLabel("자판기 위치");
376        JLabel locationSet = new JLabel();
377        String dvmLocations = null;
378        String dvm_ids = "0^1^2^3^4^";
379        String[] dvms = null;
380        if(dvm_ids != null) {
381            if(dvmLocations == null)
382                dvmLocations = "";
383            dvms = dvm_ids.split("\\^");
384        }
385
386        String[] alpha = {"0", "1", "2", "3", "4"};
387        assertThat(dvms, is(alpha));
388
389        if(dvms != null) {
390            String[] locations = {"신공학관 1층", "새천년관 1층", "학생회관 1층", "법학관 1층", "도서관 1층"};
391            for(int i = 0; i < dvms.length; i++) {
392                Integer dvmId = Integer.parseInt(dvms[i]);
393                dvmLocations += locations[dvmId];
394
395                if(i < dvms.length-1)
396                    dvmLocations += ",";
397            }
```

testController.showLocation (2021. 5. 31. 오후 10:18:53)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
DVM	3.1 %	269	8,433	8,702
src	0.0 %	0	4,234	4,234
ku.ooad.b1.vendingmachine	0.0 %	0	3,965	3,965
Controller.java	0.0 %	0	2,396	2,396
Message.java	0.0 %	0	599	599
Item.java	0.0 %	0	504	504
VerificationCode.java	0.0 %	0	343	343
DVM.java	0.0 %	0	63	63
Purchase.java	0.0 %	0	60	60
ku.ooad.b1.core	0.0 %	0	269	269
Launcher.java	0.0 %	0	269	269
test	6.0 %	269	4,199	4,468
ku.ooad.b1.test.vendingmachine	6.0 %	269	4,199	4,468
testController.java	9.8 %	269	2,476	2,745
testMessage.java	0.0 %	0	631	631
testItem.java	0.0 %	0	537	537
testVerificationCode.java	0.0 %	0	411	411
testPurchase.java	0.0 %	0	74	74
testDVM.java	0.0 %	0	70	70

해당 함수는 메인 함수를 전혀 커버하지 않는 유닛테스트 케이스로 보여 수정 요청함.

03 Static Analysis - Code Coverage (final)

Statement Coverage 최종 측정 결과

21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	6	thirdman	dd
21. 6. 8. 오후 5:37	1	thirdman	3rd cycle
21. 6. 8. 오후 5:37	1	thirdman	[no comment]
21. 6. 8. 오후 4:01	8	thirdman	[no comment]
21. 6. 8. 오후 3:37	1	thirdman	[no comment]
21. 6. 8. 오전 3:56	1	thirdman	[no comment]
21. 6. 8. 오전 3:56	1	thirdman	[no comment]
21. 6. 8. 오전 3:55	1	thirdman	[no comment]
21. 6. 7. 오후 11:49	3	thirdman	[no comment]
21. 6. 6. 오후 8:31	1	thirdman	ccc
21. 6. 6. 오후 8:31	1	thirdman	dd
21. 6. 4. 오후 12:18	1	thirdman	bf
21. 6. 4. 오후 12:18	1	thirdman	
21. 6. 4. 오후 12:17	1	thirdman	tt
21. 6. 4. 오후 12:14	1	thirdman	tt
21. 6. 4. 오후 12:14	1	thirdman	tt
21. 6. 4. 오후 12:14	1	thirdman	tt
21. 6. 4. 오후 12:14	1	thirdman	tt
21. 6. 4. 오후 12:14	1	thirdman	tt
21. 6. 4. 오후 12:14	1	thirdman	tt
21. 6. 4. 오후 12:13	1	thirdman	Bug Fixed
21. 6. 4. 오후 12:13	1	thirdman	Bug Fixed
21. 6. 4. 오후 12:13	1	thirdman	Bug Fixed
21. 6. 4. 오후 12:13	1	thirdman	Bug Fixed
21. 6. 4. 오후 12:13	1	thirdman	Bug Fixed
21. 6. 4. 오후 12:13	1	thirdman	Bug Fixed
21. 6. 4. 오후 12:12	6	thirdman	dd
21. 6. 4. 오후 12:12	1	thirdman	Split

3rd cycle

트렐로에 올라온 앞선 Code Coverage 이슈들을 통해서 3학년팀에서 수정

03 Static Analysis - Code Coverage (final)

Statement Coverage 최종 측정 결과

Element	Coverage	Covered Instructio...	M
DVM	33.5 %	4,357	
test	49.7 %	4,357	
ku.ooad.b1.test.vendingmachine	47.8 %	4,041	
Controller.java	12.8 %	309	
testMessage.java	27.4 %	173	
Message.java	26.2 %	157	
testController.java	85.1 %	2,337	
Item.java	50.6 %	255	
testItem.java	55.5 %	298	
testVerificationCode.java	47.0 %	193	
VerificationCode.java	55.1 %	189	
DVM.java	31.7 %		
testDVM.java	42.9 %		
Purchase.java	55.0 %		
testPurchase.java	63.5 %		
ku.ooad.b1.test.core	100.0 %	316	
Launcher.java	100.0 %	316	
src	0.0 %	0	
ku.ooad.b1.vendingmachine	0.0 %	0	
Controller.java	0.0 %	0	
Message.java	0.0 %	0	
Item.java	0.0 %	0	
VerificationCode.java	0.0 %	0	
DVM.java	0.0 %	0	
Purchase.java	0.0 %	0	
ku.ooad.b1.core	0.0 %	0	
Launcher.java	0.0 %	0	

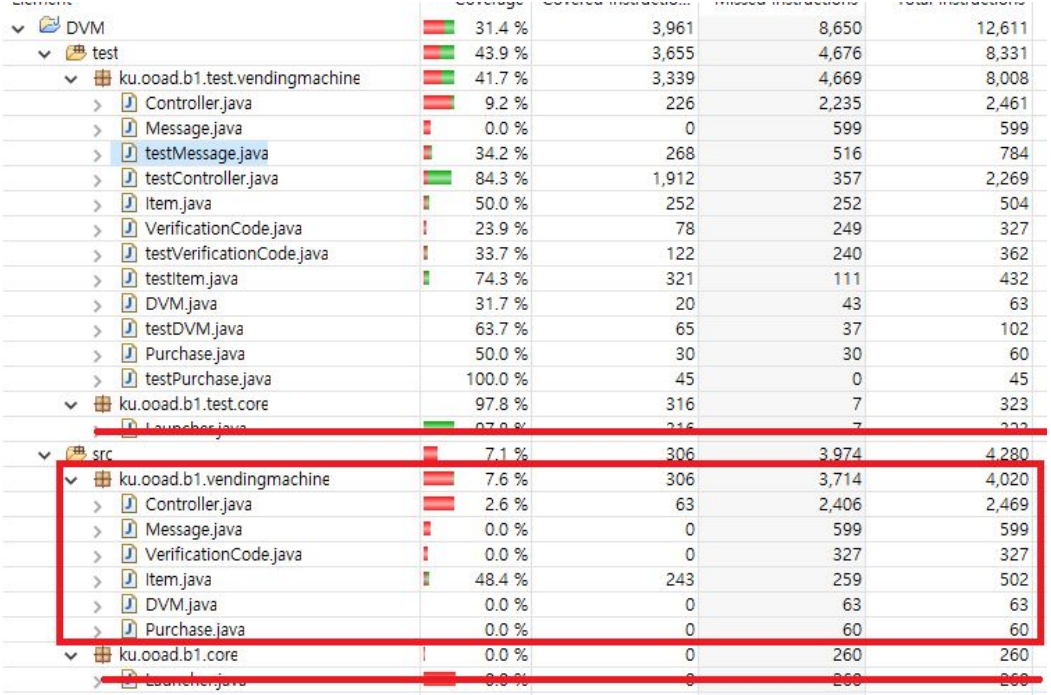
피드백 반영

DVM	31.4 %	3,961	8,650	12,611
test	43.9 %	3,655	4,676	8,331
ku.ooad.b1.test.vendingmachine	41.7 %	3,339	4,669	8,008
Controller.java	9.2 %	226	2,235	2,461
Message.java	0.0 %	0	599	599
testMessage.java	34.2 %	268	516	784
testController.java	84.3 %	1,912	357	2,269
Item.java	50.0 %	252	252	504
VerificationCode.java	23.9 %	78	249	327
testVerificationCode.java	33.7 %	122	240	362
testItem.java	74.3 %	321	111	432
DVM.java	31.7 %	20	43	63
testDVM.java	63.7 %	65	37	102
Purchase.java	50.0 %	30	30	60
testPurchase.java	100.0 %	45	0	45
ku.ooad.b1.test.core	97.8 %	316	7	323
Launcher.java	97.8 %	316	7	323
src	7.1 %	306	3,974	4,280
ku.ooad.b1.vendingmachine	7.6 %	306	3,714	4,020
Controller.java	2.6 %	63	2,406	2,469
Message.java	0.0 %	0	599	599
VerificationCode.java	0.0 %	0	327	327
Item.java	48.4 %	243	259	502
DVM.java	0.0 %	0	63	63
Purchase.java	0.0 %	0	60	60
ku.ooad.b1.core	0.0 %	0	260	260
Launcher.java	0.0 %	0	260	260

메인과 테스트패키지의 구별이 피드백했던대로 분리되지 않음, 하지만 비교적 메인커버리지가 된것으로 판단되었습니다.

03 Static Analysis - Code Coverage (final)

Statement Coverage 최종 측정 결과



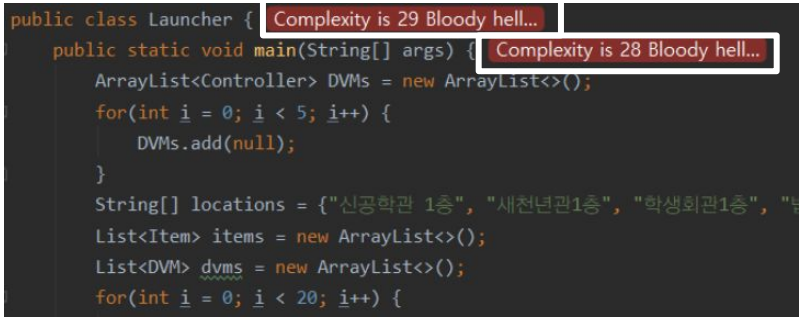
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
DVM	31.4 %	3,961	8,650	12,611
test	43.9 %	3,655	4,676	8,331
ku.ooad.b1.test.vendingmachine	41.7 %	3,339	4,669	8,008
Controller.java	9.2 %	226	2,235	2,461
Message.java	0.0 %	0	599	599
testMessage.java	34.2 %	268	516	784
testController.java	84.3 %	1,912	357	2,269
Item.java	50.0 %	252	252	504
VerificationCode.java	23.9 %	78	249	327
testVerificationCode.java	33.7 %	122	240	362
testItem.java	74.3 %	321	111	432
DVM.java	31.7 %	20	43	63
testDVM.java	63.7 %	65	37	102
Purchase.java	50.0 %	30	30	60
testPurchase.java	100.0 %	45	0	45
ku.ooad.b1.test.core	97.8 %	316	7	323
Launcher.java	97.9 %	316	7	323
src	7.1 %	306	3,974	4,280
ku.ooad.b1.vendingmachine	7.6 %	306	3,714	4,020
Controller.java	2.6 %	63	2,406	2,469
Message.java	0.0 %	0	599	599
VerificationCode.java	0.0 %	0	327	327
Item.java	48.4 %	243	259	502
DVM.java	0.0 %	0	63	63
Purchase.java	0.0 %	0	60	60
ku.ooad.b1.core	0.0 %	0	260	260
Launcher.java	0.0 %	0	260	260

ui 를 출력하는 Launcher.java를 제외하고나서, 메인패키지의 총 커버리지는 7.6%로 알 수 있었습니다.

03 Static Analysis - Cyclomatic Complexity

순환복잡도 측정 툴

- 순환 복잡도를 측정할 수 있는 eclipse plugin들 중 local PC에서 설치 가능한 plugin이 존재하지 X
- 프로젝트 코드를 Intellij 프로젝트로 새로 열어서 진행
- Intellij plugin 중 CodeMetrics라는 툴 사용



계산된 순환복잡도가 출력되는 모습

순환복잡도 줄이는 법

1. 메서드를 분리하고 if문을 줄인다 (https://stackoverflow.com/questions/16418038/how-to-reduce-cyclomatic-complexity)
2. 복잡한 if문에 대하여, 이를 for문과 함께 사용하여 if문을 단순화 시킨다 (https://stackoverflow.com/questions/42224595/how-to-reduce-cyclomatic-complexity-in-a-if-condition)
3. 가능하면 메서드 당 20~30줄을 넘기지 말고, 가능하면 parameter를 메서드에 넘기지 말 것 (https://stackoverflow.com/questions/40218530/how-to-reduce-cyclomatic-complexity-by-refactoring-the-code)

03 Static Analysis - Cyclomatic Complexity

순환복잡도 조절 가이드

* **Launcher.java**

  - 메서드 : public static void main(String[] args)
  - 복잡도가 높은 원인 : 코드가 너무 길 (약 60줄)
  - 솔루션 : 메서드를 분리할 것
  - 기대 효과 : 순환복잡도가 28에서 13으로 줄어들 것으로 예상

* **Message.java**

  - 메서드 : private Map<Integer, String> requestTo()
  - 복잡도가 높은 원인 : 메시지 타입에 따른 action이 분리되어 있지 않음
  - 솔루션 : 메시지 타입에 따른 action들이 이중 if문으로 복잡하므로, 별도의 메서드로 분리
  - 기대효과 : 순환복잡도가 53에서 28로 줄어들 것으로 예상

03 Static Analysis - Cyclomatic Complexity

순환복잡도 조절 가이드

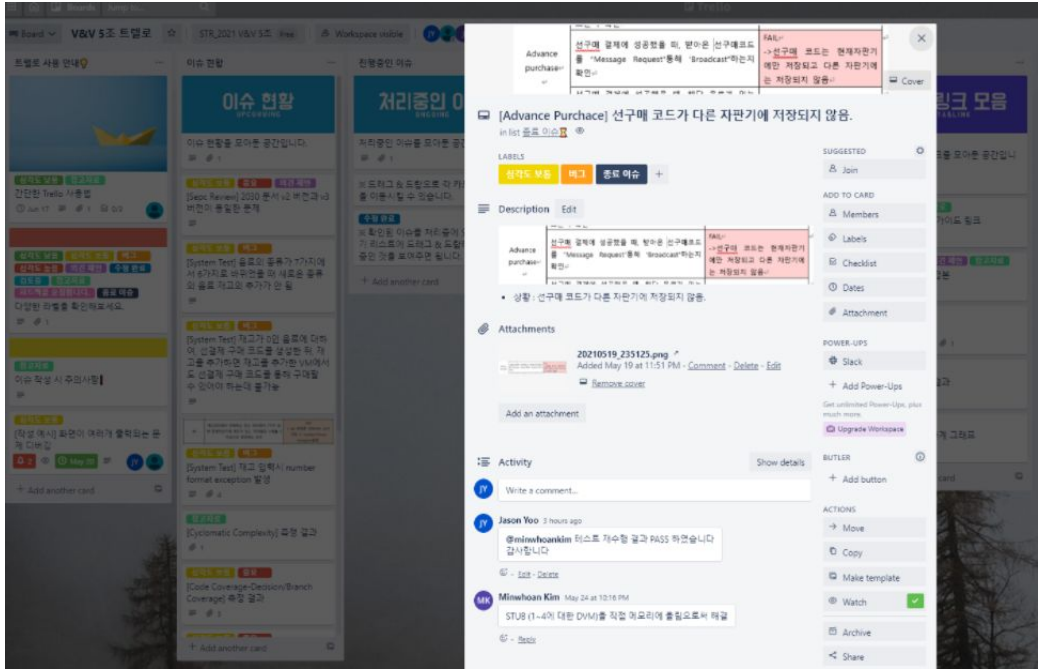
* **VerificationCode.java**

  - 메서드 : public Boolean resetCode(Integer code, Boolean broadcast)
  - 복잡도가 높은 원인 : 불필요한 if문
  - 솔루션 : " if(isDone) break; "라는 코드 제거
  - 기대 효과 : 순환복잡도가 19에서 17으로 줄어들 것으로 예상
  - 비고 : 성능과 복잡도 간에 trade-off가 있어서, 굳이 없애지 않아도 될 것 같긴 함

04 Issue Tracking - Trello

각 Cycle마다 생성된 산출물 검증 후 수정이 필요한 부분 트렐로에 issue로 등록하여 진행 사항 및 피드백 관리

04 Issue Tracking - Trello



1차 검증 때 부여한 일감 -
(spec review 및 system testing)에 대해

3학년 팀과 4학년 팀이

Comment를 통해

Communication한 모습

05 General Review(총평)

* SW Quality Assessment

  - CPT 기준 100% / 100% - - - - - - - - - - - - (가중치 x 40)

  - BFT 기준 100% / 100% - - - - - - - - - - - - - - - (가중치 x 40)

  - Code Coverage 7.6% / 100% - - - - - - - (가중치 x 16)

  - Cyclomatic Complexity 100점 / 100점 - - - - (가중치 x 4)

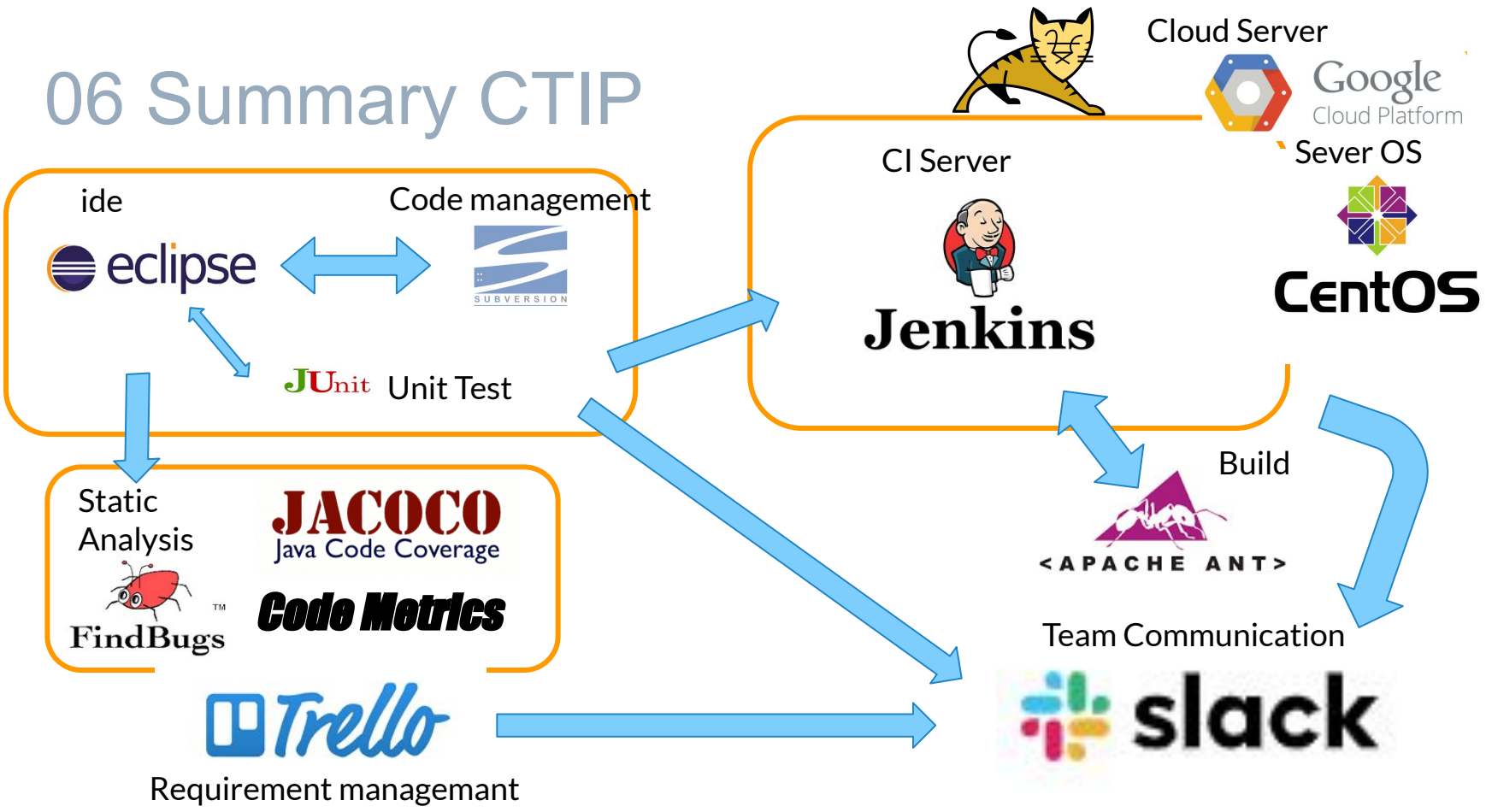
    -> (100 x 40 + 100 x 40 + 7.6 x 16 + 100 x 4) / 100 = 85.2점

    -> 2nd Cycle Quality 77.6점에 비해 7.6점 상승

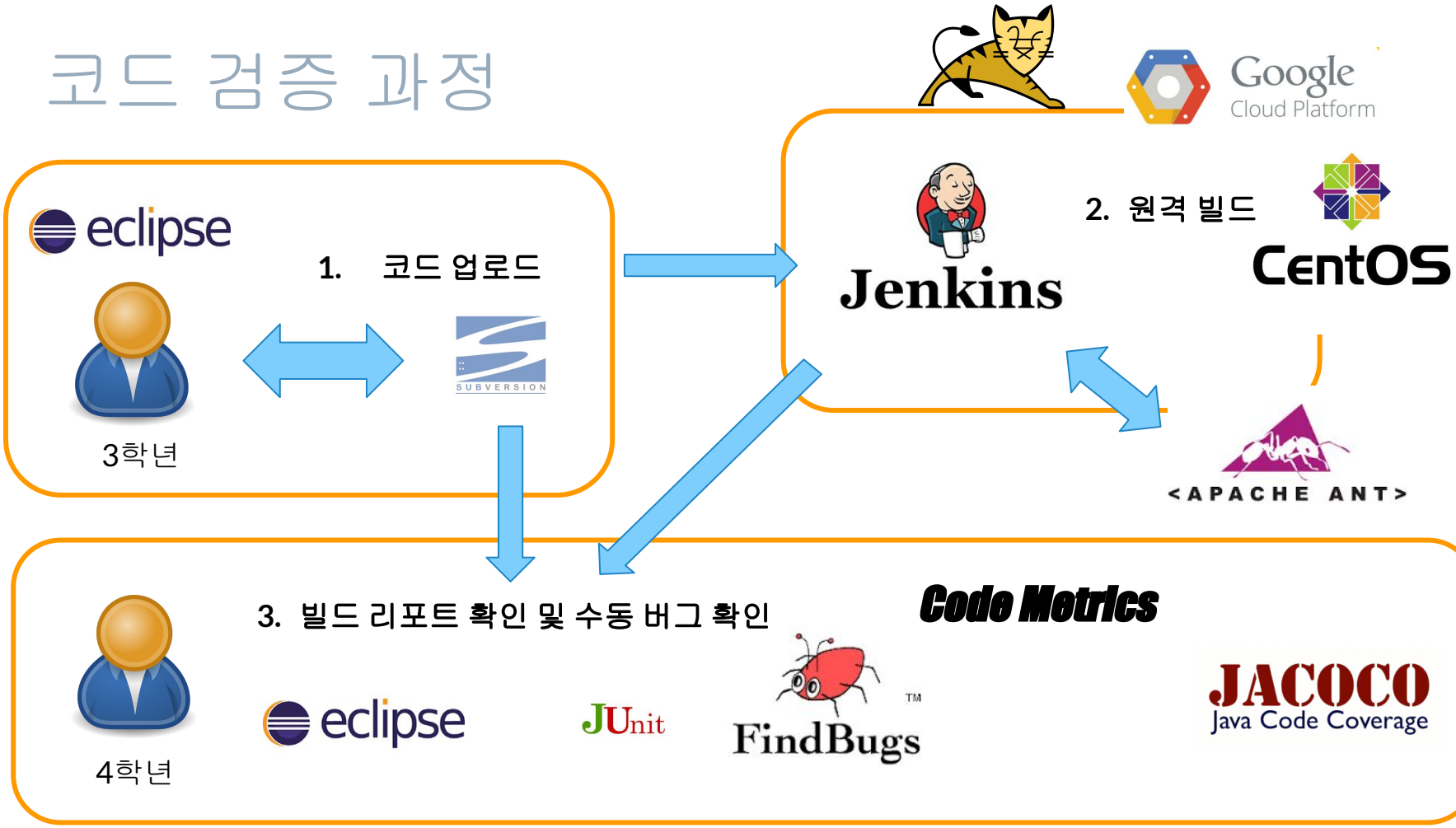
* 협업 태도에 대한 평가

  - 상호 간 열린 마음으로 요구사항 적극 수용, 빠른 의사 소통

06 Summary CTIP

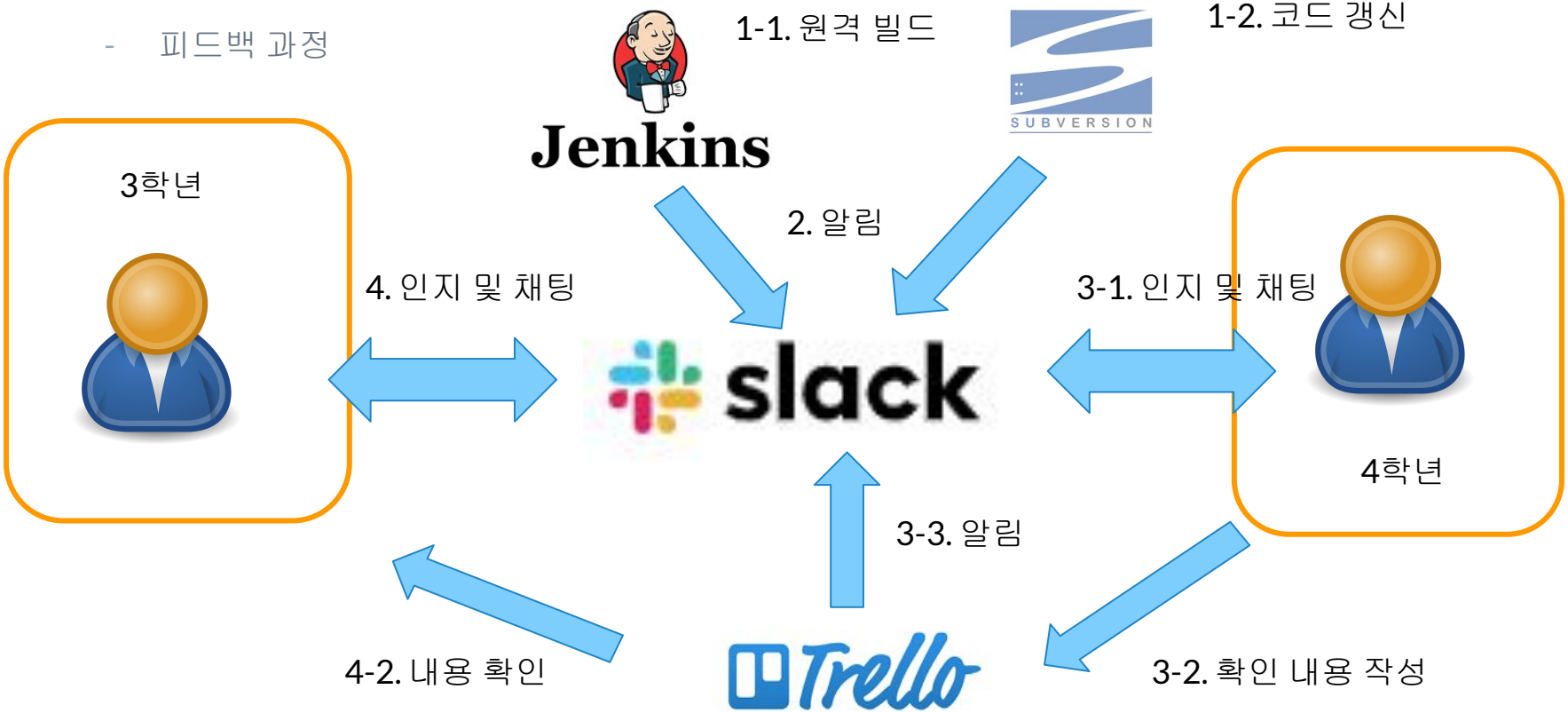


코드 검증 과정


eclipse
1. 코드 업로드
SUBVERSION
3학년
Jenkins
2. 원격 빌드
CentOS
<APACHE ANT>
3. 빌드 리포트 확인 및 수동 버그 확인
Code Metrics
eclipse
JUnit
FindBugs
JACOCO
Java Code Coverage
4학년

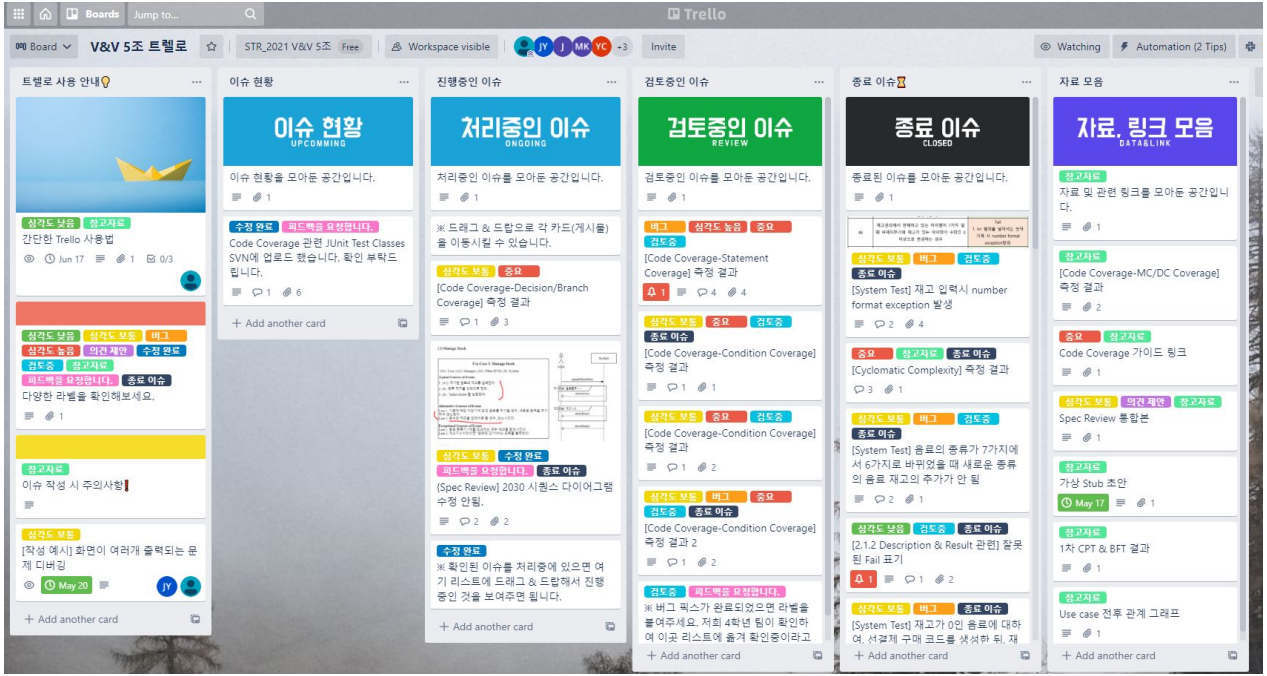
커뮤니케이션 과정

- 피드백 과정



06 Summary CTIP

일감 등록 및 이슈 관리는 트렐로로 관리함.



07 Thoughts (소감)

* 강민호 : CTIP 환경을 구축하는 법을 통해 소프트웨어 검증을 어떻게 진행해야 하는 지 알 수 있어 좋았습니다. 특히 문서화와 버그 픽스 등의 내용들을 검증하는 과정을 추후 프로젝트에서도 잘 적용할 수 있을 것 같습니다.

* 김정연 : 개발팀의 여러 문서를 비교하면서 리뷰를 진행하는과정에서 문서화를 하는것이 중요하다는것을 다시 한번 느낄 수 있었고, testing을 진행하는 부분에서 SQA의 경험과 역량에 따라 많은 부분이 달라진다는것을 배울 수 있었습니다.

07 Thoughts (소감)

* 유경원 : 개발팀의 산출물들을 리뷰하면서 문서화의 중요성을 느꼈고, 개발자 관점에서의 좋은 SW와 검증자 관점에서 좋은 SW가 다르기 때문에, 두 관점을 모두 합쳐서 생각해볼 수 있다는 걸 알게 됐습니다. 또한 테스트 매크로처럼 자동화된 툴이 생산성을 엄청나게 높여준다는 것을 느낄 수 있었습니다.

* 이지영 : 평소 게임잡, 잡코리아 등에서 V&V TDD역량을 가진 인재를 구한다는 글만 봤을때는 꼭 개발자가 문서 작업을 해야되는건가 라고 비관적이었으나, 실제로 V&V TDD를 수행을 해보면서 완성도가 높고 이후의 유지보수를 위한 프로그래밍을 위해서는 꼭 V&V , TDD단계가 필요한 것을 알게 되었습니다.

08 To OOAD Team

* 강민호 : 빠듯한 일정에도 저희들이 피드백한 내용들을 잘 반영해주셔서 감사합니다. V&V 수업도 좋은 경험이 될 수 있을 것이라고 생각해 내년에 듣는 것을 추천 드립니다. 다시 한 번 한학기 동안 정말 수고 많으셨습니다.

* 김정연 : 소통이 무엇보다 중요한 프로젝트에서 원활한 소통이 이루어질 수 있어서 다행이었고 빠듯한 일정으로 개발을 진행하시느라 한학기동안 수고 많으셨습니다.

08 To OOAD Team

* 유경원 : 객체 지향 설계를 나름 잘 하신 것 같은데, 더 깔끔하고 확장성 및 유연성 있는 설계를 하고 싶으시다면, DI(Dependency Injection)과 디자인 패턴을 공부하시는 걸 추천 드려요. 최범균 저 '개발자가 반드시 정복해야 할 객체 지향과 디자인 패턴'이라는 책 추천 드립니다.
* 이지영 : 개발기간이 짧았던 만큼 개발결과에 너무 연연하지 않으셨으면 좋겠습니다!!
부족한 V&V팀과 함께 해주셔서 감사합니다~
수업을 진행하는동안에 OOP와 TDD에 대한 역량이 많이 늘었길 바랄게요!

09 Conclusion

Q. V&V를 잘하기 위한 필요충분조건은?

A. 팀의 역량, SW 요구사항, 재정적 여유, 프로젝트의 데드라인 등을 고려하여, 이에 걸맞는 적절한 양과 질의 테스트 툴과 테스트 종류를 적절히 취사 선택해 테스팅을 진행함으로써, SW의 Quality를 보장하고, 이러한 과정에 대한 이론적 지식과 실무적 경험이 풍부해야 한다.