System Test & Static Analysis - V&V 5팀

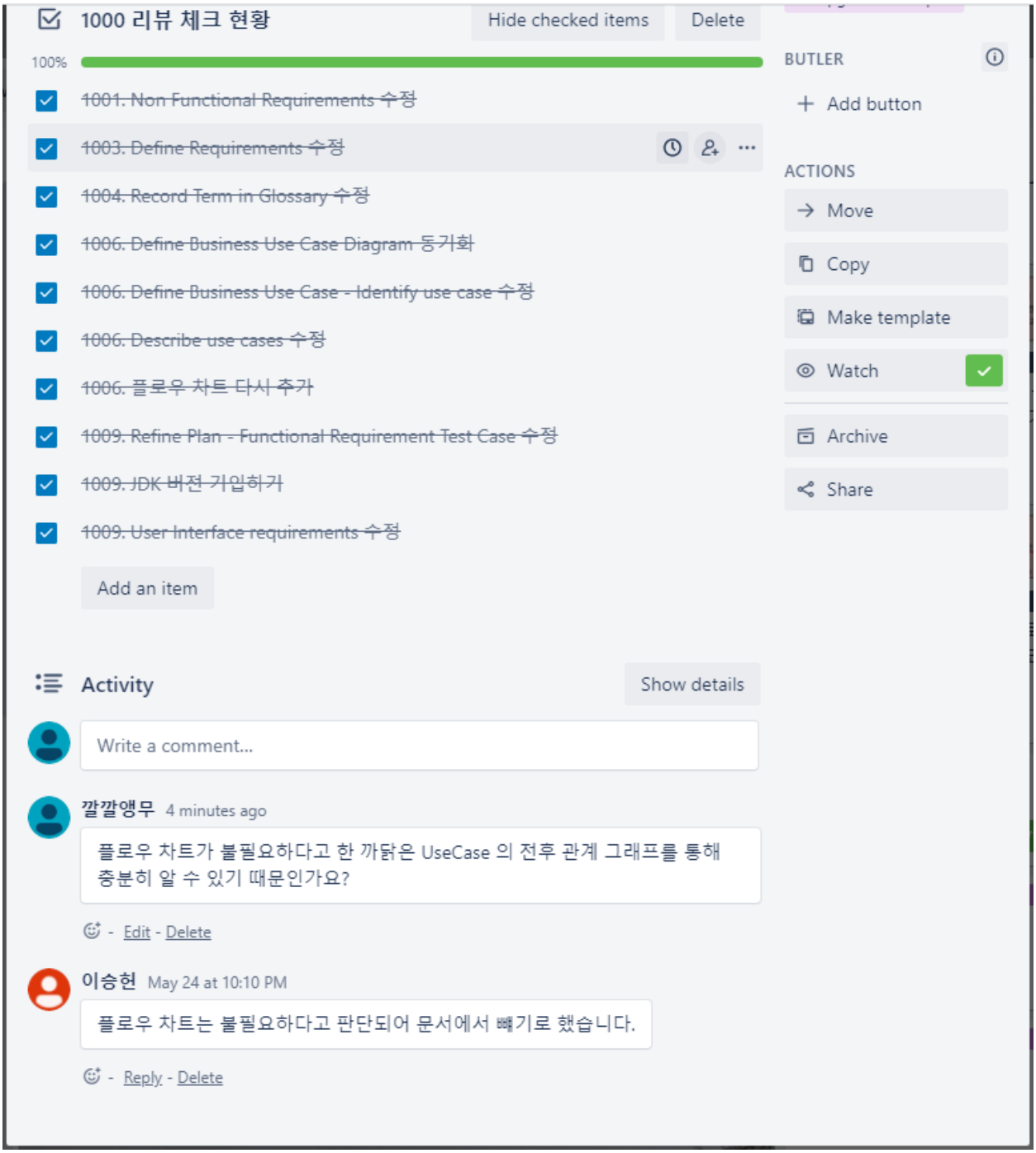
팀원 : 이지영(팀장), 강민호, 김정연, 유경원

목차

---

1. Spec Review 2nd

1차 검증 때 Trello에 올린 Spec Review 관련 일감에 대해 comment를 서로 남긴 모습입니다.



OOPT 1000: Planning 문서

- **1001. Define Draft Plan**
  - **Non Functional Requirements**

지적 사항

- 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.

Resource Estimation

👉 "직관적인 UI를 제공하고 사용하기 편해야 한다"라는 말은 너무 정성적인 평가입니다. 구체적으로 어떻게 직관적인 UI와 편한 UX를 제공했는지를 정량적으로 파악할 방법이 있는지 작성해야합니다.
⇒ **직관성은 곧 사용성**이므로 이미 존재하고 많이 이용되는 시스템을 벤치마킹한다고 말하면 이는 증명되는 것이므로 해당 서술로 수정하는 것도 좋습니다.

👉 "네트워크 통신 중 데이터 누락과 Latency가 없어야 한다."라는 항목은 물리적으로 딜레이가 아예 없을 수는 없어 실현하기 어렵습니다. 따라서 '**투명성**'과 '**무결성**'**이 제공된다**고 수정하면 좋을 것이라 생각됩니다.

💡 Java를 사용할 때 JDK 버전도 보고서에 작성하셔야합니다.

⇒ 수정 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.

Resource Estimation

- **1003. Define Requirements**
  - Performance Requirements

지적 사항

- 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**안에 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, 20p

> Q. 1초인가요? 0.1초인가요? SRS 문서에선 0.1초라고 되어있어 서로 말이 상충됩니다.하나로 통일 부탁드립니다.
> A. 1초로 통일하겠습니다.

⇒ 수정 완료

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

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

지적 사항

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

Stock	재고
Count	Item의 수량

> Stock과 Count이 문서상 같은 표현으로 사용되고 있습니다. 같은 의미를 혼용하고 있는데, **Stock을 '품목(Item의 종류)'으로 고치거나** 혼동을 줄이기 위해 **한 가지 표현으로 통일**하는 것이 좋습니다.
> 또한 Stock을 '재고의 수량'을 표현한 것이라면 "**재고" 대신에 "재고량"으로 수정**하셔야 합니다. '재고'는 물건을 나타내는 단어지 수량을 표현하는 단어가 아닙니다.

⇒ 수정 완료

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	

용어를 Count로 통일하고 용어의 정의를 item의 수량으로 명백하게 표현함.

- **1006. Define Business Use Case (p.9)**

  지적사항

- Identify use cases
  - Other Vending Machine : 동일한 테크워크에 관련한 관계

- Identify use cases
  - Use-cases by actor-based
    - Set Up, Select Mode, Select Item, Purchase, Select Verification Code
  - Use-case by evident-based
    - Show Item, Check Stock Count, Update stock, Che Create code, Show Code, Item Out, Check Verifica Message Request, Message Response
- Allocate system functions into related use cases

'Use-cases by actor-based 에 Use-cases by evident-based인 'Set UP'이 있습니다. Set up은 최초 프로그램 실행시 동작되는 UseCase 이므로 Hidden이 맞습니다.

Use-case by evident-based 에서 Use-case's' by evident-based로 오타 수정 부탁드립니다.

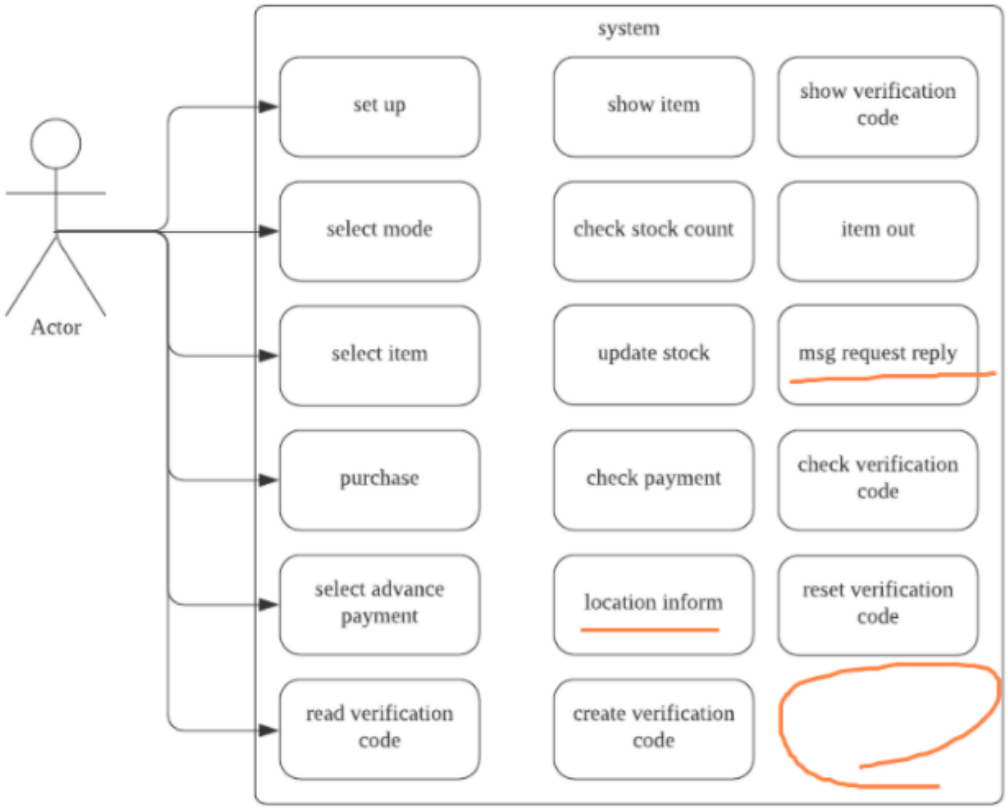
⇒ 수정 완료

Identify use cases

- Use-cases by actor-based
  - Select Mode, Select Item, Purchase, Select advance payment, Read Verification Code
- Use-cases by evident-based
  - Set Up, Show Item, Check Stock Count, Update stock, Check Payment, Inform Location, Create code, Show Code, Item Out, Check Verification Code, Reset verification code, Message Request, Message Response

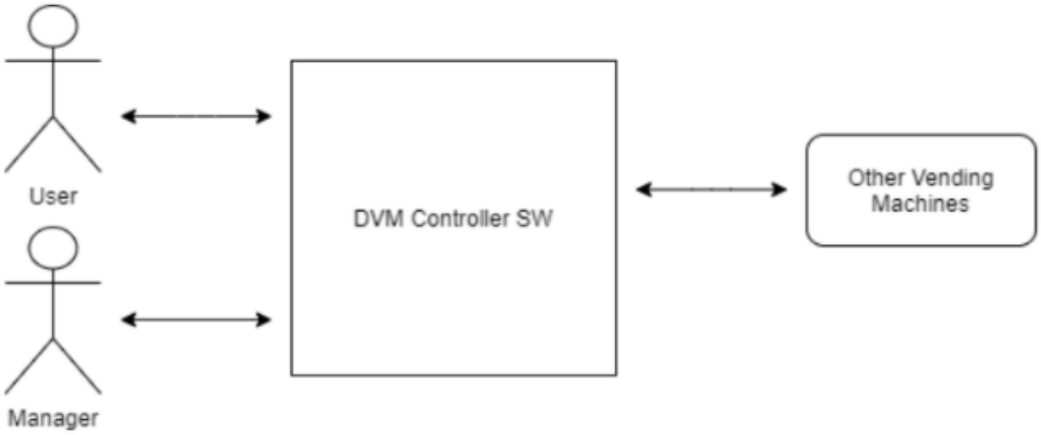
지적사항

- use case diagram



👉 'set up'은 hidden use case로 actor가 없어야 하는데 diagram 상에서는 있는 것으로 그려져 있습니다.

💡 UseCase의 명칭 오류
(없는 항목) → manage stock, advance purchase
msg request reply → message request, messge response
location inform → inform location

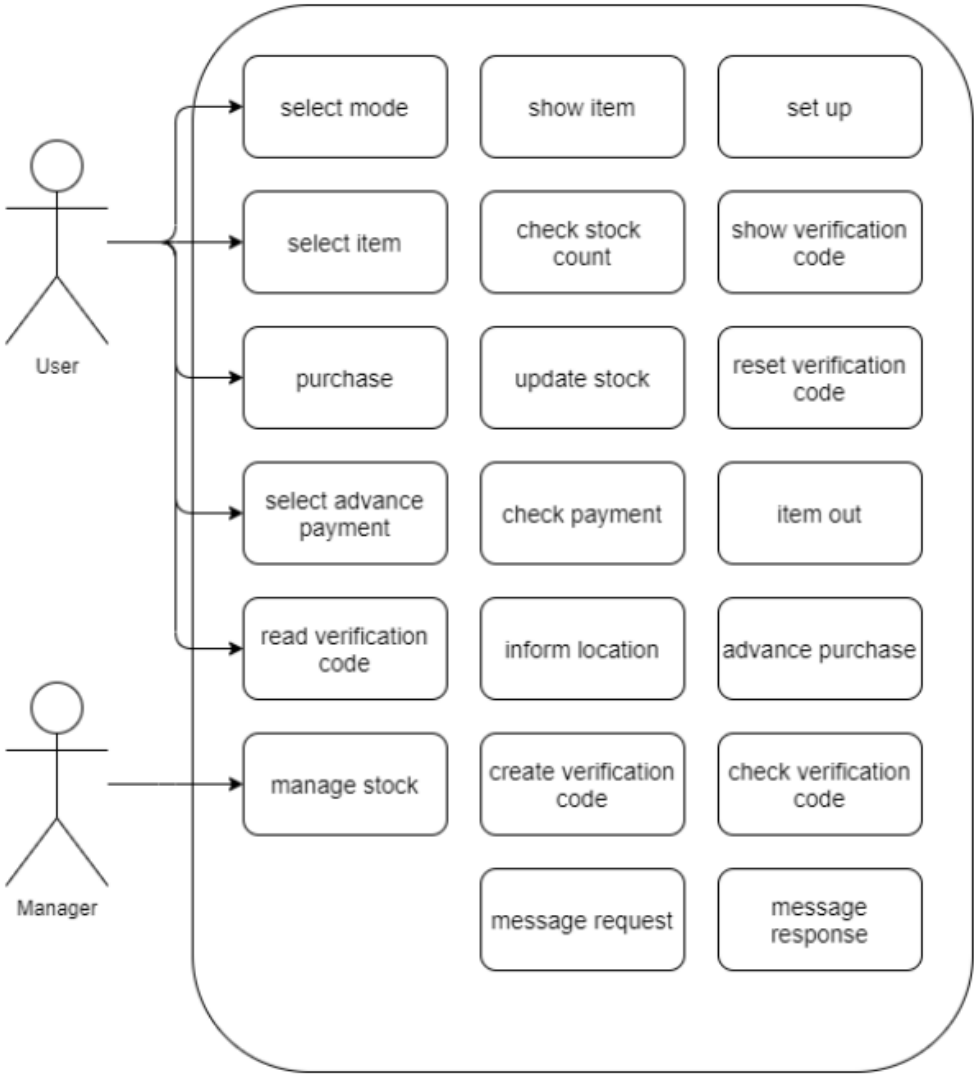


👉 위 그림(Define System boundary)에는 Manager Actor가 있는데 v2 버전에선 UseCase Diagram에 Actor만 표시되어있습니다. User와 Manager를 구분하여 각각 UseCase를 연결해 UseCase Diagram을 나타내는 것이 좋습니다.

💡 위 내용들은 2030 문서부터 Refine 항목에 잘 고쳐져 있어 동기화를 부탁드립니다.

⇒ 수정 완료

- Draw a use case diagram



지적사항

- Describe use cases
  - 4. Show Item

Use Case	4. Show Item
Actor	None
Description	- 사용자가 현재 자판기의 재고 유무에 상관없이 구매 가능한 음료를 보여준다.

"사용자가 현재 자판기의 재고 유무에 상관없이 구매 가능한 **음료**를 보여준다."
→ "사용자가 현재 자판기의 재고 유무에 상관없이 구매 가능한 **모든 음료**를 보여준다." 로 구체화하여 수정 부탁드립니다.

  - 7. Update Stock

Q. '현장구매로 음료 구매시(=FR2.5=UC8)' 또는 '선구매코드로 음료 구매시(=FR2.8=UC11)' 또는 '관리자가 재고 변경시(=FR1.3=UC3)' 이 3가지 경우에 해당 VM의 재고가 변경되었음을 broadcast를 통해 곧 바로 알려서 모든 VM들의 재고 정보를 매 순간 동기화 시키는 것인지
vs
아니면 이 3가지 경우마다 broadcast를 하는 것이 아니라 다른 VM이 재고 정보를 요청할 때 응답만 하는 것인지 궁금합니다.
(OOPT 2030와 OOPT 2040 문서에서도 마찬가지)
A. local 재고 증감 후, remote에 재고 감소 요청을 진행합니다.

이 부분은 v2에서 Flow chart가 누락되면서 시퀀스를 알 수 없어 나온 질문입니다. 누락된 플로우 차트를 재작성 부탁드립니다.

⇒ 수정 완료

Use Case	4. Show Item
Actor	None
Description	- 사용자에게 모든 음료를 보여준다.

- 1008 단계를 구체화하여 설명하면서 플로우차트는 아예 삭제하기로 함.

지적사항

- 12. Inform Location

Use Case	12. Inform Location
Actor	None
Description	- 다른 자판기로부터 응답 msg가 도착한 후 실행되는 use case. - 사용자에게 선택한 음료를 판매하고 있는 자판기의 위치를 알려준다.

👉 "- 사용자에게 선택한 음료를 판매하고 있는 **자판기**의 위치를 알려준다" → "- 사용자에게, 선택한 음료를 판매하고 있는 **모든 자판기**의 위치를 알려준다" 로 수정 부탁드립니다.

- 15. Item Out

👉 "결제가 완료된 음료를 제공하는 use case" → "결제 완료된 음료를 제공하는 use case"
⇒ 사소한 표현 수정이므로 넘어가셔도 됩니다.

⇒ 수정 완료

Use Case	12. Inform Location
Actor	None
Description	- 다른 자판기로부터 응답 msg가 도착한 후 실행되는 use case. - 사용자에게 선택한 음료를 판매하고 있는 모든 자판기의 위치를 알려준다.

Use Case	15. Item Out
Actor	None
Description	- 결제 완료된 음료를 제공하는 use case.

1009. Refine Plan (v2 p.19)

지적 사항

- 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.

⇒ 수정 완료

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.

지적사항

- No.12 Location Inform
  - Positive Test Case

> 💡 "다른 자판기의 위치를 출력한다" → "결제할 상품이 판매되는 모든 자판기의 위치를 출력한다."로 더 자세하게 수정 부탁드립니다.

- No.18 Reset Verification Code
  - Positive Test Case
    - 사용된 코드를 폐기하고, 모든 자판기에 방송한다.

> 💡 '방송한다'를 "코드 폐기 내용을 알린다"로 수정하는게 좋습니다.

⇒ 수정 완료

11	Location Inform	1. 기본: 결제할 상품이 판매되는 모든 자판기의 위치를 출력한다.	1. 기본: 다시 메시지를 요청한다.

17	Reset Verification Code	1. 기본: 사용된 코드를 폐기하고, 모든 자판기에 코드 폐기 내용을 알린다.	1. 기본: 시스템이 재부팅 된다.

기존 No.11 Check Advanced Payment 유스케이스는 삭제되었다.

지적사항

- JAVA 언어를 사용할 때는 JDK 버전도 같이 써야합니다.

- Resources
  - Man Month : 5 Persons
    - A Team Leader
    - A Document Manager
    - 2-3 Engineers
  - Period : 6 Days(Around 20Hours)
  - Hardware : Intel PC
  - Software
    - OS : Windows 10
    - Programming Language : JAVA

20

- User Interface requirements

- User Interface requirements
  - UI가 직관적이어야 한다.
  - 처음 사용하는 사람도 큰 문제없이 사용할 수 있어야 한다.

> "UI가 직관적이어야 한다"를 구체적으로 나타낼 수 있는 방안과 함께 정량적으로 서술하거나, 직관성은 곧 사용성이므로 어떤 유명한 시스템을 벤치마킹한다는 내용이 추가되어야 합니다.

⇒ 수정 완료

- Software
  - OS : Windows 10
  - Programming Language : JAVA (JDK 1.8)
  - Case Tool : draw.io , miro.com
- Configuration Management
  - Version Control System 을 활용
- Quality Assurance Plan
  - Iteration 마다 Technical Review를 할 계획이다.
  - Quality 체크 리스트를 통해 점검할 것이다.

- User Interface requirements
  - fresh store 자판기의 UI를 벤치마크한다.
  - 처음 사용하는 사람도 큰 문제없이 사용할 수 있어야 한다.

OOPT 2030 : Analysis 문서

- **2131. Define Essential Use Cases**

지적 사항

- 1. Set Up (p.3)
  - Cross Reference

Use Case	1. Set Up
Actor	None
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Hidden
Cross Reference	System Functions: R1.1, R1.2, R4.1, R2.4, R4.1 Use Case: “Message Request”, “Update Stock”, “Select Mode”
Pre-Requisites	N/A

👉 Q. R4.2는 왜 없나요? (OOPT 2040 문서도 마찬가지)
A. 수정하면서 삭제되었습니다.

💡 "R4.1"이 중복되었으므로 삭제 부탁드립니다.

- Typical Course of Events

  2. (S): "다른 DVM의 재고를 받아온다" → "다른 DVM의 재고 정보를 받아온다."로 수정 부탁드립니다.

  3. (S): "자판기의 재고를 업데이트 한다." → "모든 자판기의 재고 정보를 업데이트 한다."로 수정 부탁드립니다.

- Alernative Courses of Events

Alternative Courses of Events	Line 1. (A3)의 응답이 3번 이상 오지 않을 경우, 해당 (A3)은 재고가 없다고 설정한다.
Exceptional Courses of Events	N/A

  적힌 내용은 다른 성공 시나리오의 사례가 아니라 예외처리에 대한 문장이므로 Exceptional Course of Events로 옮기는 것이 적절해 보입니다.

⇒ Alternative Courses of Events 제외 수정 완료

2040 문서와 동일하게 유지하기 위해 기존 이벤트 위치를 유지한 것으로 판단됨.

지적사항

- 2. Select Mode (p.4)
  - Alternative Courses of Events

Alternative Courses of Events	Line 1. 세 가지 모드를 동시에 선택할 경우 현장구매 모드로 간주한다.
Exceptional Courses of Events	N/A

  현재 구현된 프로그램으로는 마우스 버튼 입력으로 진행되기 때문에 세 가지 모드를 동시에 선택하는 경우가 생길 수 없습니다. 따라서 해당 내용은 삭제처리하는 것이 좋습니다.

⇒ 수정 완료

Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

지적사항

- 3. Manage Stock (p.4)

Cross Reference	System Functions: R1.3, R1.2, R2.4 Use Case: “Select Mode”, “Update Stock”
Pre-Requisites	N/A
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A2): 추가된 음료의 재고를 입력한다. 2. (S): 음료 재고를 업데이트 한다. 3. (S): “Select Mode”를 실행한다.
Alternative Courses of Events	Line 1. 기존에 해당 자판기에 없던 음료를 추가할 경우, 새로운 항목을 추가하여 업데이트 한다. Line 2. 음수의 재고를 업데이트 할 경우, 감소시킨다.
Exceptional Courses of Events	Line 1. 음료 종류가 7개를 초과하는 경우 에러를 발생시킨다. Line 2. 재고가 0 미만이면 “잘못 된 값” 메시지를 출력한다.

  - Cross Reference

    💡 Select Mode는 선행 조건이므로 삭제하고, Pre-Requisites에 추가하는 것이 좋습니다.

  - Pre-Requisites

    💡 Cross Reference의 수정에 따라 "재고 관리를 누른다"로 수정하는 것이 좋습니다.

- Typical Courses of Events

> 1. (A2) : 추가된 음료의 재고를 입력한다.
> → "추가된 음료 수량을 입력한다"로 수정 부탁드립니다.
> 2. (S): 음료 재고를 업데이트 한다.
> → 업데이트는 확실하지 않은 표현이므로 "입력한 값만큼 음료의 재고량을 증가시킨다"로 수정합니다. '재고'는 수량을 나타내는 단어가 아닙니다.

- Alternative Course of Events

> Line 2. "음수의 재고를 업데이트 할 경우, 감소시킨다." → "입력된 값이 음수일 경우, 재고량을 입력된 값만큼 감소 시킨다." 로 구체화하여 작성 부탁드립니다.

- Exceptional Course of Events

> Line 2. "재고가 0 미만이면 '잘못된 값' 메시지를 출력한다." → "입력 결과 예상 재고량이 0 미만이면 최신화를 하지 않고, '잘못된 값' 알림 메시지를 출력한다."로 수정 부탁드립니다.

⇒ Cross Reference 와 Pre-Requisites 제외 수정 완료

Use Case	3. Manage Stock
Actor	Manager
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Evident
Cross Reference	System Functions: R1.3, R1.2, R2.4 Use Case: “Select Mode”, “Update Stock”
Pre-Requisites	재고관리모드 상태
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A2): 추가된 음료의 수량을 입력한다. 2. (S): 입력한 값만큼 음료의 수량을 증가시킨다. 3. (S): “Select Mode”를 실행한다.
Alternative Courses of Events	Line 1. 기존에 해당 자판기에 없던 음료를 추가할 경우, 새로운 항목을 추가하여 업데이트 한다. Line 2. 입력된 값이 음수일 경우, 음료의 수량을 입력된 값만큼 감소시킨다.
Exceptional Courses of Events	Line 1. 음료 종류가 7개를 초과하는 경우 에러를 발생시킨다. Line 2. 재고가 0 미만이면 최신화를 하지 않고, “잘못 된 값” 메시지를 출력한다.

2040문서와 동일하게 가기 위해 두 항목은 그대로 유지하는 것으로 판단됨.

지적사항

- 4. Show Item (p.5)

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 음료 항목을 모두 보여준다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1. 취소 버튼을 누르면 “Select Mode”를 실행한다.

  - Typical Courses of Events

    1. (S) : "음료 항목을 모두 보여준다" → 20가지 음료 선택 버튼이 출력되는 상황이므로 "20가지 음료 선택 버튼을 모두 보여준다."로 수정합니다.

⇒ 수정 완료

Use Case	4. Show Item
Actor	None
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Hidden
Cross Reference	System Functions: R2.1, R1.2 Use Case: “Select Mode”
Pre-Requisites	현장구매모드 상태
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 20가지 음료 선택 버튼을 모두 보여준다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1. 취소 버튼을 누르면 “Select Mode”를 실행한다.

지적사항

- 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 현재자판기에 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.

⇒ 그대로 유지됨

지적사항

- 7. Update Stock (p.7)

Cross Reference	System Functions: R2.4, R1.1, R1.3 Use Case:"Set Up", "Manage Stock"
Pre-Requisites	최신화 해야되는 음료 정보와 최신화 값이 입력됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S) : 음료에 대해 재고를 최신화 한다.
	N/A

  - Cross Reference

    💡 다른 자판기에서 선결재 하여 음료 재고 정보가 변했을 때도 갱신이 일어나야 하므로 Use Case 20. Message Response(R4.2)도 추가되어야 합니다.

  - Pre-Requisites

    💡 '최신화'와 '업데이트' 두 가지로 용어가 혼용되고 있는데, 용어는 통일되는 편이 좋습니다.

  - Typical Course of Events

    💡 1. (S) : "음료에 대해 재고를 최신화 한다." → "수량 변동 내역이 있는 음료 재고 정보를 업데이트 한다." 로 수정하는 것이 좋습니다.

⇒ Pre-Requisites 제외 수정 완료

Type	Hidden
Cross Reference	System Functions: R2.4, R1.1, R1.3, R4.2 Use Case:“Set Up”, “Manage Stock”, “Message Response”
Pre-Requisites	최신화 해야되는 음료 정보와 최신화 값이 입력됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S) : 수량 변동 내역이 있는 음료의 수량을 최신화 한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Pre-Requisites는 최신화라는 용어로 그래도 사용됨. 의미가 같으므로 단어 자체를 통일 시킬 필요성이 없다고 판단된 것 같음.

지적사항

- 10. Check Payment (p.8)
  - Exceptional Courses of Events

Courses of Events	
Exceptional Courses of Events	Line 1. 정상적인 결제 카드가 아니라면, “잘못된 카드”라는 에러 메시지를 출력한다.

> 2050 문서를 보았을 때 해당 예외처리 사항은 존재하지 않았습니다. 따라서 해당 문구는 삭제하는 것이 좋습니다.

⇒ 수정 완료

Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

지적사항

- 11. Select Advance Payment (p.8)

Cross Reference	System Functions: R2.8, R2.3, R2.6, R4.1, R1.2 Use Case: “Check Stock Count”, “Advance Purchase”, “Message Request”, “Select Mode”
Pre-Requisites	음료가 선택됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A1): 선택한 음료에 대해 선구매를 선택한다. 2 (S): 선택된 음료에 대해 “Message Request”로 재고를 확인한다. 3 (S): 여전히 재고가 남아있다면 “Message Request”로 다른 자판기에 있는 음료의 재고를 깎는다. 4 (S): 선구매 진입을 반환한다.
Alternative Courses of Events	1. 확인과 취소를 둘다 누를 경우 사용자로 부터 재입력 받는다.
Exceptional Courses of Events	Line 1. 취소를 누를 경우 “Select Mode”로 돌아간다. Line 1. 위치보기를 누를 경우 “Inform Location”을 실행한다. Line 3. 재고가 없다면 품절 메시지를 띄우고, “Select Mode”로 돌아간다.

- Cross Reference

> 💡 Select Mode는 세 가지 모드를 선택하는 Use Case이므로 Cross Reference에 적절하지 않습니다. 따라서 R1.2(Select Mode)는 삭제하는 것이 좋습니다. Pre-Requisites에서 이미 'Select Item'(R2.2) Usecase가 진행된 상태이므로 R2.2는 굳이 추가하지 않아도 됩니다. 대신 Exceptional Courses of Event에서 "Infrom Location"이 실행되므로 R2.9를 추가하는 것이 맞습니다.

- Typical Courses of Events

  1. (A1) : "선택한 음료에 대해 선구매를 선택한다" → "선택한 음료에 대해 선구매를 진행한다." 로 수정 부탁드립니다.

- Alternative Courses of Events

  해당 내용은 예외 처리 사항이므로 Exceptional Courses of Events에 있어야 했습니다. 하지만 2050에서 구현된 프로그램은 **버튼 입력으로 진행되어 동시에 입력되는 경우가 없으므로 해당 문구를 삭제하는 것이 좋습니다.**

- Exceptional Courses of Events

  해당 내용들은 또 다른 성공 시나리오 이므로 Alternative Courses of Events로 옮기는 것이 좋습니다.

⇒ Exceptional Courses of Events 제외 수정 완료

Cross Reference	System Functions: R2.8, R2.3, R2.6, R4.1, R2.9 Use Case: "Check Stock Count", "Advance Purchase", "Message Request", "Inform Location"
Pre-Requisites	음료가 선택됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A1): 선택한 음료에 대해 선구매를 진행한다. 2 (S): 선택된 음료에 대해 "Message Request"로 재고를 확인한다. 3 (S): 여전히 재고가 남아있다면 "Message Request"로 다른 자판기에 있는 음료의 재고를 깎는다. 4 (S): 선구매 진입을 반환한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1. 취소를 누를 경우 "Select Mode"로 돌아간다. Line 1. 위치보기를 누를 경우 "Inform Location"을 실행한다. Line 3. 재고가 없다면 품절 메시지를 띄우고, "Select Mode"로 돌아간다.

Exceptional Courses of Events 내용을 그대로 유지한 까닭은 다른 자판기의 음료를 깎는 것이 해당 UseCase의 내용이고, 그 이외 사항은 모두 예외사항 이라고 해석이 가능하기 때문으로 보인다.

지적사항

- 12. Inform Location (p.9)

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

  - Pre-Requisites

    💡 위치보기 버튼을 누를 경우 실행되므로 "음료가 선택됨"이 아니라 "위치보기를 누름"으로 바뀌어야 합니다.

⇒ 수정 완료

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.
Alternatives	N/A

지적사항

- 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.

⇒ Typical Courses of Events 제외 수정 완료

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

지적사항

- 15. Item Out (p.11)
  - Typical Courses of Events

Cross Reference	System Functions: R2.12, R2.5, R3.1 Use Case: “Purchase”, “Read Verification Code”
Pre-Requisites	선택된 음료에 대한 결제가 완료됨
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 선택된 음료를 제공한다.
Alternative	N/A

💡 1. (S) : "선택된 음료를 제공한다." → "판매된 음료를 제공한다."로 수정 부탁드립니다.

⇒ 수정 완료

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 판매된 음료를 제공한다.

지적사항

- 16. Read Verification Code (p.12)

Cross Reference	System Functions: R3.1, R1.2, R3.2, R3.3, R2.12 Use Case: “Select Mode”, “Check Verification Code”, “Reset Verification Code”, ”Item Out”
Pre-Requisites	N/A
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A1): 선구매 코드를 입력한다.
Alternative Courses of Events	Line 1. 동시에 여러가지 숫자를 누를 경우 아무것도 입력받지 않는다.
Exceptional Courses of Events	Line 1. 취소를 누를경우 “Select Mode”로 돌아간다.

  - Cross Reference
    - 해당 UseCase는 이미 'Select Mode'가 선행된 상태이므로 R1.2는 삭제해도 됩니다.
  - Pre-Requisites
    - "Select Mode에서 선구매 버튼을 누름."이라고 수정합니다.
  - Typical Courses of Events
    - "선구매 코드 6자리를 입력한다"로 구체적으로 수정합니다.
  - Alternative Courses of Events
    - Exceptional Courses of Events로 내용을 옮깁니다.
- Exceptional Courses of Events
  - Alternative Course of Events로 내용을 옮깁니다.

⇒ 변경 없음

지적사항

- 17. Check Verification Code (p.12)

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 입력된 코드를 가지고 있는지 검사한다.

  - Typical Courses of Events

    💡 1. (S) : "입력된 코드를 가지고 있는지 검사한다." → "입력된 코드가 유효한지 검사한다"로 수정합니다.

⇒ 수정 완료

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 입력된 코드가 유효한지 검사한다.

지적사항

- 18. Reset Verification Code (p.13)

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 사용한 선구매 코드의 정보를 삭제한다. 2. (S): "Message Request"를 통해 모든 자판기에게 해당 코드가 사용되었음을 Broadcast 한다.

  - Typicla Courses of Events

    💡 1. (S) : "사용한 선구매 코드의 정보를 삭제한다" → "사용한 선구매 코드 정보를 삭제한다."
    ⇒ '~의'는 일본식 표현이라 재량껏 수정하시면 됩니다.

⇒ 수정 완료

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 사용한 선구매 코드 정보를 삭제한다. 2. (S): "Message Request"를 통해 모든 자판기에게 해당 코드가 사용되었음을 Broadcast 한다.

지적사항

- 19. Message Request (p.14)
  - Typical Courses of Events

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1.1. (S): Broadcast의 경우, 네트워크에 등록된 모든 (A3)에게 보내고자 하는 메시지를 전달한다. 1.2. (S): 돌려받은 응답을 반환한다.2.1. (S): 특정 (A3)에게 메시지를 보낼 경우, 해당 (A3)에게 메시지를 전달한다. 2.2 (S): 돌려받은 응답을 반환한다.

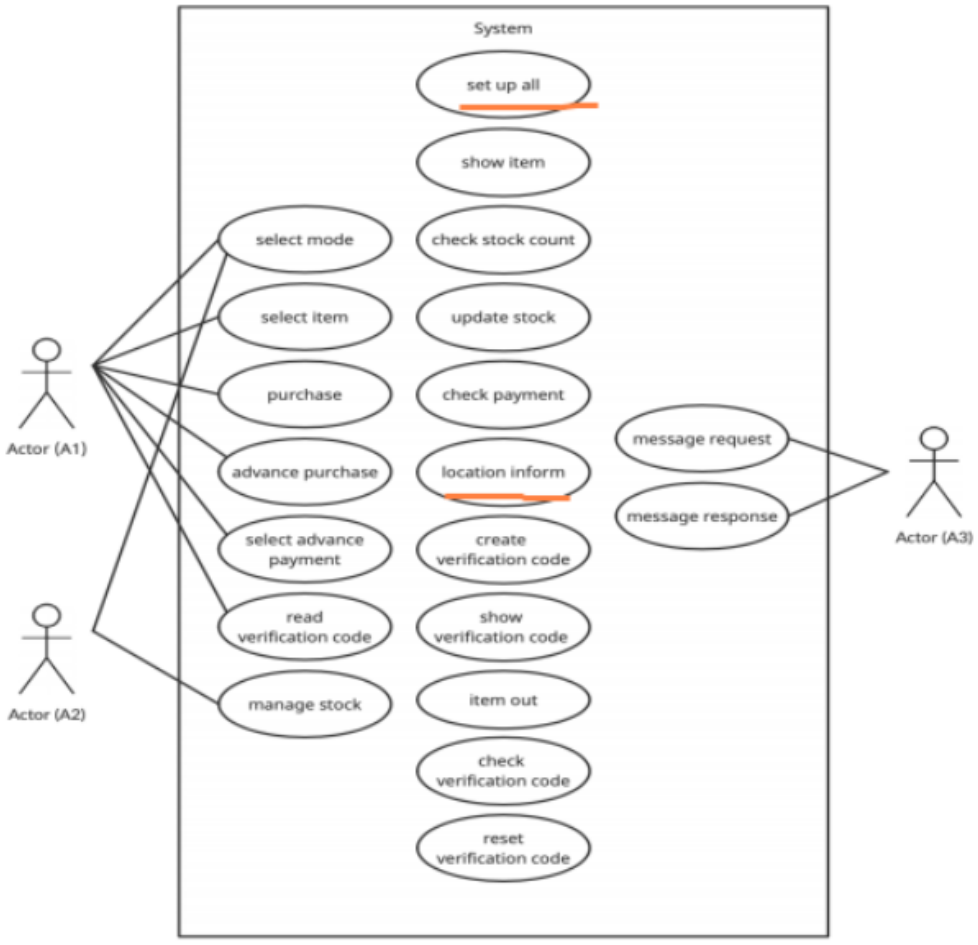
💡 개행이 안 되어 있습니다.

⇒ 수정 완료

- **2132. Refine Use Case Diagrams (p.15)**

지적사항

2132. Refine Use Case Diagrams



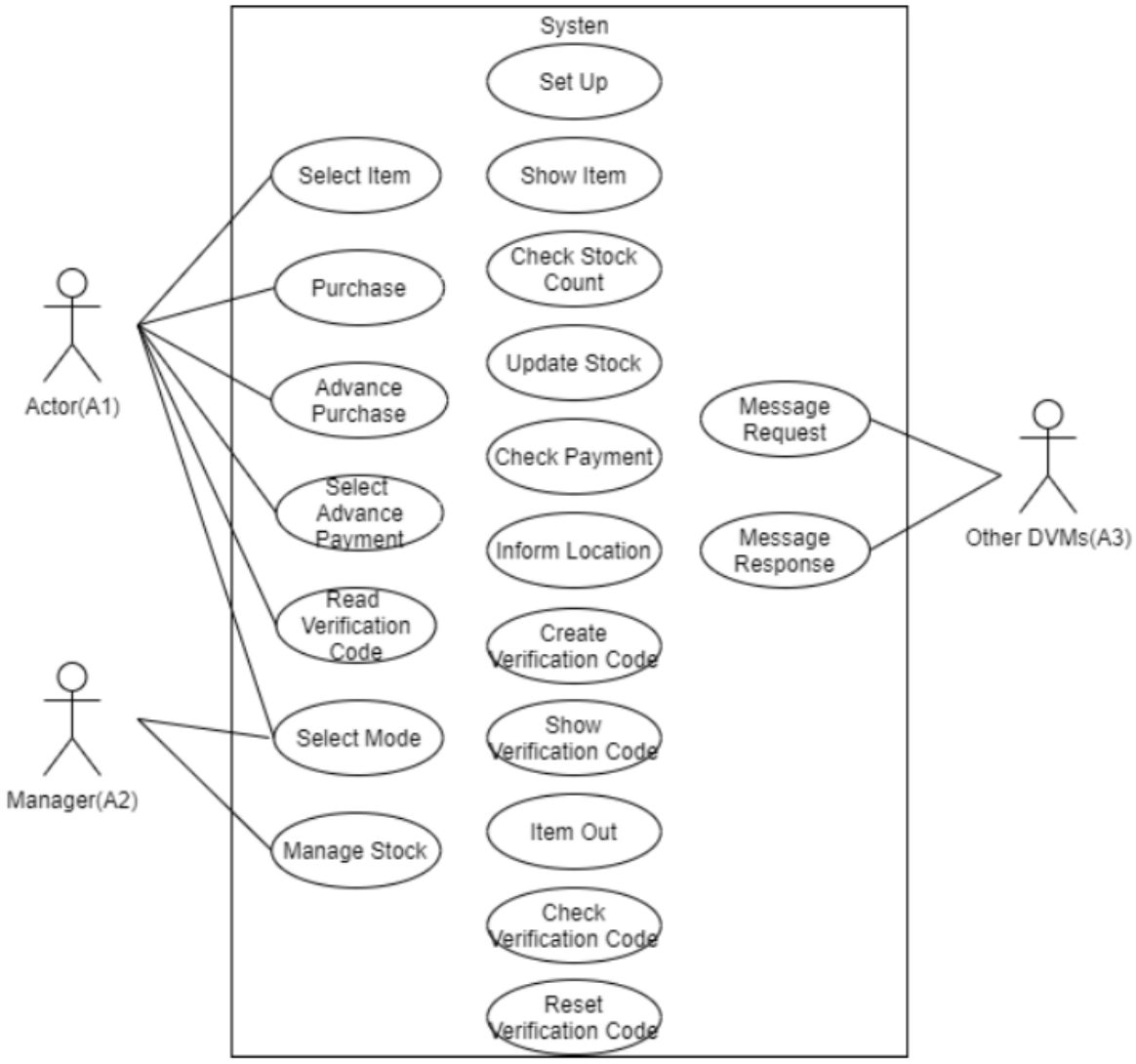
* (A1): User, (A2): Manager, (A3): Other DVMs

(1) 'set up all'은 function 명입니다. 해당 function에 대응되는 UseCase 명칭은 "set up"이므로 수정 부탁드립니다.

(2) 'location inform'은 function 명입니다. 해당 function에 대응되는 UseCase 명칭은 "inform location"이므로 수정 부탁드립니다.

⇒ 수정 완료

2132. Refine Use Case Diagrams



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

지적사항

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

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

⇒ 변동 없음

상기에 언급된 2, 3, 7, 11, 18번 UseCase에 대한 시퀀스 노트 내용이 수정되지 않음.

지적사항

- 2134. Refine Glossary (p.22)
  - Stock : Item의 재고
  - Count : Item의 수량

> 👉 Q. Stock은 DVM 전체에 걸쳐 있는 Item의 총 개수이고 Count는 특정 VM 내에 있는 Item의 개수인가요? (OOPT 2040 문서의 Class Diagram에 stock이라는 변수가 없는 걸 보고 그렇게 추측했습니다.)
>
> A. 둘 다 같은 의미로 사용되고 있습니다. Stock은 삭제하겠습니다.

⇒ 수정 완료

OOPT : Software Requirements Specification 문서

1. Introduction

- 1.3 Definitions, Acronyms and Abbreviations

지적사항

> 💡 앞선 문서에서는 Count와 Stock이 혼용되었고, Stock이란 단어를 삭제하기로 하였으나, 해당 문서에서는 Count가 삭제되고, Stock이 남아있습니다. 통일하는 것이 좋습니다.

> 💡 15~17번이 공란입니다. 행 삭제 부탁드립니다.

⇒ 수정 완료

- **2.4 Constraints**
  - 동시에 두 개 이상의 메시지를 받고 응답할 수 없다

💡 해당 조건은 2050 문서에서 구현된 버튼 입력 방식으로는 나올 수 없는 경우이므로 삭제하는 것이 좋습니다.

⇒ 수정 완료

- **3.2 Functional Requirements**
  - 3.2.9 Item Out (p.17)

3.2.9 Item Out

구분	내용
요구사항ID	Req-9
요구사항명	Show Verification Code
개요	음료를 제공한다.
사전조건	1. 음료가 선택되어 있다. 2. 선택된 음료에 대해 결제가 완료되었거나, 유요한 선구매 코드가 입력되었다.
시나리오	1. 선택된 음료를 제공한다.

👉 요구사항명이 잘못되어 있습니다. 수정 부탁드립니다.

⇒ 수정 완료

- **3.3 Performance Requirements**

3.3 Performance Requirements

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

💡 앞선 문서에서 전송 시간은 1초 이내에 이루어지는 것으로 되어있으나 해당 문서에선 0.1초로 되어있습니다. 1초로 수정 부탁드립니다.

⇒ 수정 완료

3.3 Performance Requirements

- Network Message들의 전송속도는 1초안에 이루어 지고 전송간 오류가 없어야 한다.
- 사용자가 screen에 입력 후 해당 기능이 실행되는데 걸리는 시간은 0.1초 이내여야 한다.

OOPT 2040 : Design 문서

- **2141. Define Essential Use Cases**

지적사항

- 4. Show Item(pg 4)

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 모든 *Item*을 모두 보여준다.

👉 Q. 모든 아이템을 모두 보여준다는게 수량과 종류를 모두 보여준다는 의미인가요?
A. 음료의 종류만을 보여준다는 의미입니다.

더 명확한 의미로 수정 요청

⇒ 수정 완료

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 모든 *Item*의 선택 버튼을 보여준다.
Alternative Courses of Events	Line 1. 취소 버튼을 누르면 "*showMode*"를 실행한다.
Exceptional Courses of Events	N/A

지적사항

- 5. Select Item(pg 5)

Use Case	5. Select Item
Actor	User
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Evident
Cross Reference	System Functions: R2.2, R1.2, R2.1, R2.3 Use Case: “Select Mode”, “Check Stock Count”, “Show Item”
Pre-Requisites	“*showItem*”이 실행됨

  - Pre-Requisites

    👉 Q. select item은 현장구매모드에서 사용자가 구매하려는 음료를 선택하는 것이기 때문에 "mode값이 1이다.(현장구매모드이다)"라는 내용을 추가하면 더 명확해져서 좋을 것 같아요

⇒ 수정 완료

	Use Case: “Select Mode”, “Check Stock Count”, “Show Item”
Pre-Requisites	“*showItem*”이 실행됨(mode 값이 1이며 현장구매모드이다.)
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System

지적사항

- 6. Check Stock Count(pg 6)

Use Case	6. Check Stock Count
Actor	None
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Hidden
Cross Reference	System Functions R2.3, R2.2, R4.1 Use Case: "Select Item", "Message Request"
Pre-Requisites	*Item*이 선택되었다.

  - Pre-Requisites

    Q. 음료를 선택하고 재고를 확인하는 부분이니 선택한 자판기의 "mode값이 1이다.(현장구매모드이다)"라는 내용을 추가하면 더 명확해져서 좋을 것 같아요

⇒ 수정 완료

	Use Case: "Select Item", "Message Request"
Pre-Requisites	*Item*이 선택되었다.(mode 값이 1이며 현장구매모드이다.)
Typical Courses	(A1): User, (A2): Manager, (A3): Other DVM, (S): System

수정사항

- 8. Purchase와 9. Advance Purchase(pg 7)

  : isAdvance에 대한 설명이 없음.

⇒ 수정 완료

Pre-Requisites	1. *Item*이 선택되었고 *mode* 값이 1 이다. 2. *count* > 0 이다. 3. isAdvance 는 false이다.

Pre-Requisites	1. *Item*이 선택되었고 *isAdvance*가 *true* 이다. 2. *count* > 0 이다.

추가 수정 사항

Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A1): 선택한 *Item*에 대해 선구매를 선택한다. 2 (S): 선택된 *Item*에 대해 "*sendMessage*"로 재고를 확인한다. 3 (S): 여전히 재고가 남아있다면 "*sendMessage*"로 다른 자판기에 있는 *Item*의 *count*를 깎는다. 4 (S): "*showAdvancePurchase*"를 실행한다.

4번째 줄의 메서드 명칭 변경. AdvancePurchase → showAdvancePurchase

지적사항

Use Case	7. Update Stock
Actor	None
Courses of Events	
Exceptional Courses of Events	N/A

Caption Original

에러 메시지를 생성하는 경우가 발생한다면 2141. Define Essential Use Cases 7.Update Stock 의 Exceptional Courses of Events부분에 에러메시지가 발생함을 명시해 주셨으면 합니다.

⇒ 수정 완료

Exceptional Courses of Events	Line 1. 음료의 종류의 가짓수가 7개를 초과할 경우 [음료 종료 초과] 라는 printe를 실행한다.

지적사항

- Alternative Courses of Events

Alternative Courses of Events	1. 확인과 취소를 둘다 누를 경우 사용자로 부터 재입력 받는다.
Exceptional	Line 1 취소를 누를 경우 "showMode"로 돌아간다

1. 확인과 취소를 둘 다 누를 경우 사용자로 부터 재입력 받는다

> Q. 확인과 취소 둘다 버튼으로 입력을 받기때문에 동시에 입력받는 경우는 없을 것 같습니다.

⇒ 수정 완료

Alternative Courses of Events	
Exceptional	Line 1 취소를 누를 경우 "showMode"로 돌아간다

삭제함.

2142. Define Reports, UI, and Storyboards

지적사항

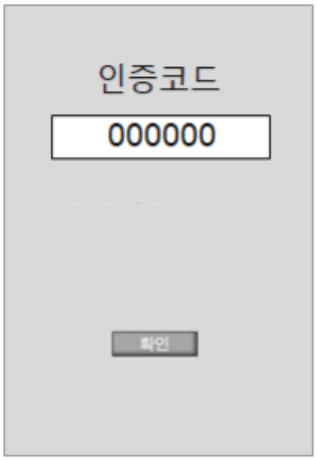
- 7) Show Verification Code(pg 18)

7) Show Verification Code

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

⇒ 수정 완료

7) Show Verification Code



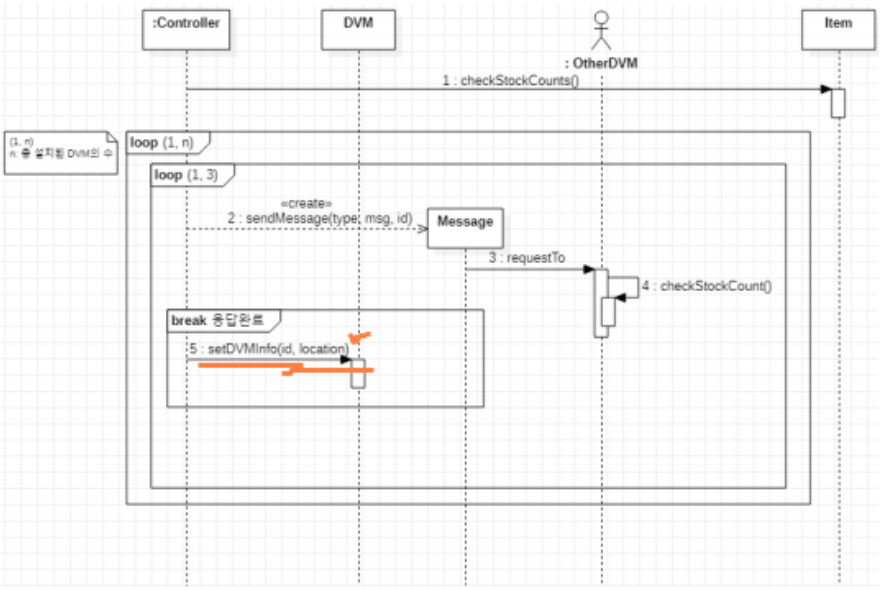
2143. Define Interaction Diagrams

지적사항

- (1) Set up(pg 19)

2143. Define Interaction Diagrams

(1) Set Up

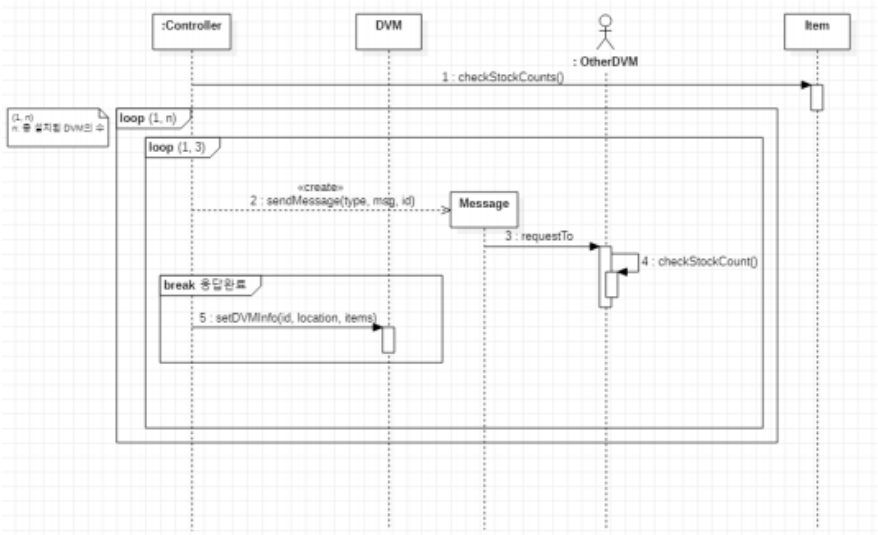


"setDVMInfo" 오퍼레이션은 DVM의 id, location 그리고 item의 count 항목을 업데이트하는 것이므로 매개변수로 item도 같이 넣어야 합니다.
⇒ "setDVMInfo(id, location)" → setDVMInfo(id, location, item)

⇒ 수정 완료

2143. Define Interaction Diagrams

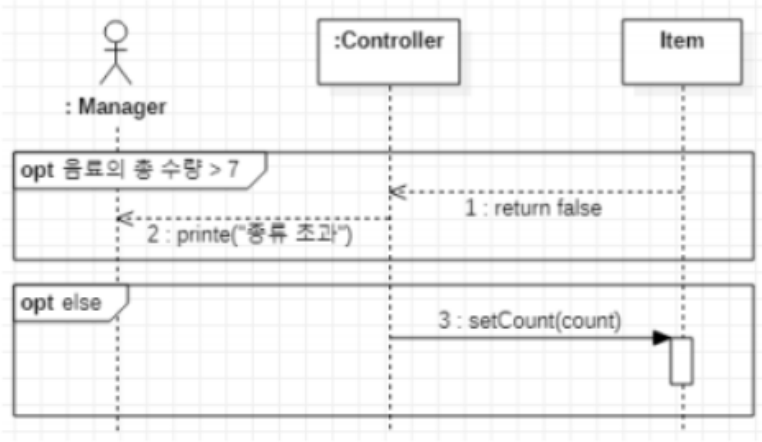
(1) Set Up



지적사항

- (7) Update Stock (pg 22)

(7) Update Stock

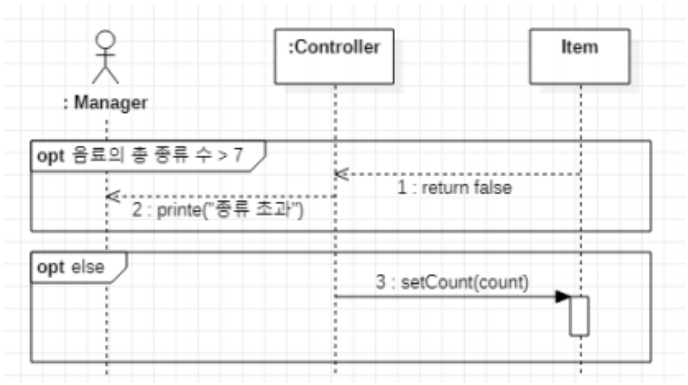


👉 Q. 음료의 총 수량이 7초과일경우 종류 초과 라는 에러 메시지를 생성한다고 되어있는데 "음료의 총 수량"이라는 의미는 음료의 종류가 7가지를 넘어간 경우인가요/음료의 재고의 수가 7개를 넘어간 경우인가요?
A. 음료의 종류의 가짓수가 7개를 초과한 경우입니다.

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

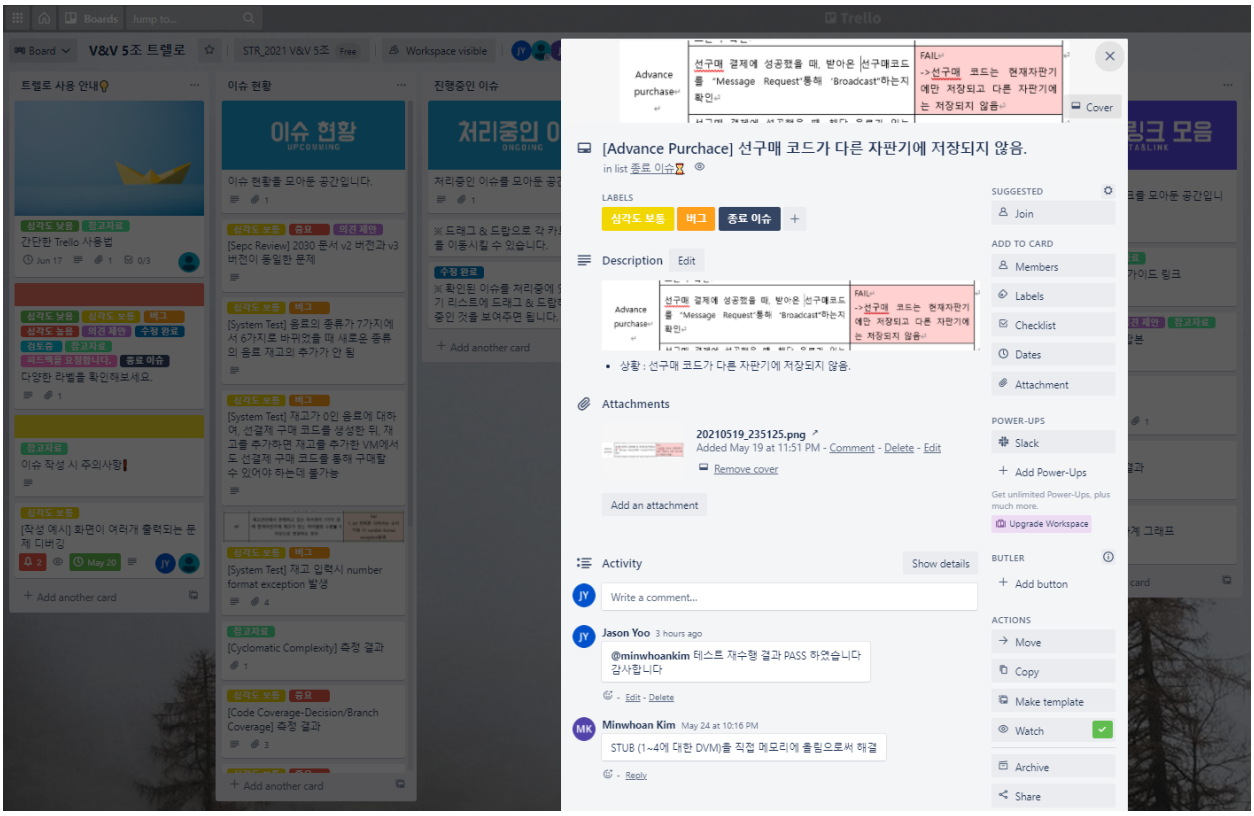
⇒ 수정 완료

(7) Update Stock



2. System Testing 2nd

1차 검증 때 Trello에 올린 System Testing 관련 일감에 대해 comment를 서로 남긴 모습입니다.



2.1 Categorize Partition Testing (CPT)

2.1.1 Categorize & Property & Constraints

Group	Category	Value	Property	Constraints
View	Current UI	초기화면	ui1	
		음료선택화면	ui2	
		선결제코드 입력화면	ui3	
		재고관리화면	ui4	
		결제화면	ui5	
		선결제안내화면	ui6	
		위치안내화면	ui7	
		선결제코드 안내 화면	ui8	
Mode	Prepurchase	선구매 진행		[if ui3 || ui5 || ui7 || ui8]
		선구매 하지않음		[if ui5]
Stock	item stock	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]
	Item types	7개		[if ui4]
		7개 미만		[if ui4]
Msg	Msg broadcast	Success		[if isAdvance]
		Fail		[error]
Input	Button	cancle		[if ui6]
		현장결제 버튼		[if ui1]
		선결제코드 입력 버튼		[if ui1]
		재고 관리 버튼		[if ui1]
		아이템 선택		[if ui2 || ui4]
		결제(1%실패)		[if ui5]
		실패 테스트		[if ui5]
		위치보기		[if ui6]
		선구매		[if ui6]
	Page move	확인		[if ui3 || ui4 && item || ui7 || ui8]
		취소		[if ui2 && !item || ui3 || ui4 && item]
	Modify item	아이템 수량 0이하로 변경		[if ui4]
		아이템 수량 0으로 변경		[if ui4]
		아이템 수량 0이상으로 변경		[if ui4]
	Verification code	유효한 선결제 코드		[if ui3 && isAdvance]
		유효하지 않은 코드 입력		[if ui3 && isAdvance]

<TSL Generator Input File>

```
#
# 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]


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]

<TSL Generator Output File>

Test Case 1 <error>
선구매코드 broadcast : Fail

Test Case 2 (Key = 1.0.0.0.0.1.0.0.0.)
Current UI : initial page
선구매 여부 : <n/a>
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 현장구매
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 3 (Key = 1.0.0.0.0.2.0.0.0.)
Current UI : initial page
선구매 여부 : <n/a>
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 선결제코드 입력
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 4 (Key = 1.0.0.0.0.3.0.0.0.)
Current UI : initial page
선구매 여부 : <n/a>
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 재고 관리
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 5 (Key = 2.0.1.0.0.4.0.0.0.)

Current UI : item select
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 6 (Key = 2.0.2.0.0.4.0.0.0.)
Current UI : item select
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 7 (Key = 2.0.3.0.0.4.0.0.0.)
Current UI : item select
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 8 (Key = 3.1.1.0.1.0.1.0.1.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인

Modify item : <n/a>
Verification Code : 유효한 선결제 코드 입력

Test Case 9 (Key = 3.1.1.0.1.0.1.0.2.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : 유효하지 않은 코드 입력

Test Case 10 (Key = 3.1.1.0.1.0.2.0.1.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 취소
Modify item : <n/a>
Verification Code : 유효한 선결제 코드 입력

Test Case 11 (Key = 3.1.1.0.1.0.2.0.2.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 취소
Modify item : <n/a>
Verification Code : 유효하지 않은 코드 입력

Test Case 12 (Key = 3.1.2.0.1.0.1.0.1.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행

선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : 유효한 선결제 코드 입력

Test Case 13 (Key = 3.1.2.0.1.0.1.0.2.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : 유효하지 않은 코드 입력

Test Case 14 (Key = 3.1.2.0.1.0.2.0.1.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 취소
Modify item : <n/a>
Verification Code : 유효한 선결제 코드 입력

Test Case 15 (Key = 3.1.2.0.1.0.2.0.2.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 취소
Modify item : <n/a>
Verification Code : 유효하지 않은 코드 입력

Test Case 16 (Key = 3.1.3.0.1.0.1.0.1.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : 유효한 선결제 코드 입력

Test Case 17 (Key = 3.1.3.0.1.0.1.0.2.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : 유효하지 않은 코드 입력

Test Case 18 (Key = 3.1.3.0.1.0.2.0.1.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 취소
Modify item : <n/a>
Verification Code : 유효한 선결제 코드 입력

Test Case 19 (Key = 3.1.3.0.1.0.2.0.2.)
Current UI : input verificaion code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : <n/a>

선구매코드 broadcast : Success
Button : <n/a>
Page Move : 취소
Modify item : <n/a>
Verification Code : 유효하지 않은 코드 입력

Test Case 20 (Key = 4.0.1.1.0.4.1.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 21 (Key = 4.0.1.1.0.4.1.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 22 (Key = 4.0.1.1.0.4.1.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 23 (Key = 4.0.1.1.0.4.2.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 24 (Key = 4.0.1.1.0.4.2.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 25 (Key = 4.0.1.1.0.4.2.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 26 (Key = 4.0.1.2.0.4.1.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택

Page Move : 확인
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 27 (Key = 4.0.1.2.0.4.1.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 28 (Key = 4.0.1.2.0.4.1.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 29 (Key = 4.0.1.2.0.4.2.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 30 (Key = 4.0.1.2.0.4.2.2.0.)
Current UI : modify sotck

선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 31 (Key = 4.0.1.2.0.4.2.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 32 (Key = 4.0.2.1.0.4.1.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 33 (Key = 4.0.2.1.0.4.1.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0으로 변경

Verification Code : <n/a>

Test Case 34 (Key = 4.0.2.1.0.4.1.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 35 (Key = 4.0.2.1.0.4.2.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 36 (Key = 4.0.2.1.0.4.2.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 37 (Key = 4.0.2.1.0.4.2.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0

판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 38 (Key = 4.0.2.2.0.4.1.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 39 (Key = 4.0.2.2.0.4.1.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 40 (Key = 4.0.2.2.0.4.1.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 41 (Key = 4.0.2.2.0.4.2.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 42 (Key = 4.0.2.2.0.4.2.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 43 (Key = 4.0.2.2.0.4.2.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock 0 & remote stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 44 (Key = 4.0.3.1.0.4.1.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>

Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 45 (Key = 4.0.3.1.0.4.1.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 46 (Key = 4.0.3.1.0.4.1.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 47 (Key = 4.0.3.1.0.4.2.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 48 (Key = 4.0.3.1.0.4.2.2.0.)

Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 49 (Key = 4.0.3.1.0.4.2.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 50 (Key = 4.0.3.2.0.4.1.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 51 (Key = 4.0.3.2.0.4.1.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인

Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 52 (Key = 4.0.3.2.0.4.1.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 확인
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 53 (Key = 4.0.3.2.0.4.2.1.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0미만으로 변경
Verification Code : <n/a>

Test Case 54 (Key = 4.0.3.2.0.4.2.2.0.)
Current UI : modify sotck
선구매 여부 : <n/a>
선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0으로 변경
Verification Code : <n/a>

Test Case 55 (Key = 4.0.3.2.0.4.2.3.0.)
Current UI : modify sotck
선구매 여부 : <n/a>

선택한 아이템의 재고 : local stock > 0
판매하는 아이템 가짓수 : 7개 미만
선구매코드 broadcast : <n/a>
Button : 아이템 선택
Page Move : 취소
Modify item : 아이템 수량 0이상으로 변경
Verification Code : <n/a>

Test Case 56 (Key = 5.1.0.0.1.7.0.0.0.)
Current UI : payment
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : 결제(1%실패)
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 57 (Key = 5.1.0.0.1.8.0.0.0.)
Current UI : payment
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : 실패 테스트
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 58 (Key = 5.2.0.0.0.7.0.0.0.)
Current UI : payment
선구매 여부 : 선구매 하지않음
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 결제(1%실패)
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 59 (Key = 5.2.0.0.0.8.0.0.0.)
Current UI : payment
선구매 여부 : 선구매 하지않음
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 실패 테스트
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 60 (Key = 6.0.0.0.0.5.0.0.0.)
Current UI : inform prepurchase
선구매 여부 : <n/a>
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 위치보기
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 61 (Key = 6.0.0.0.0.6.0.0.0.)
Current UI : inform prepurchase
선구매 여부 : <n/a>
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : <n/a>
Button : 선구매
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 62 (Key = 6.0.0.0.0.9.0.0.0.)
Current UI : inform prepurchase
선구매 여부 : <n/a>
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>

선구매코드 broadcast : <n/a>
Button : 취소
Page Move : <n/a>
Modify item : <n/a>
Verification Code : <n/a>

Test Case 63 (Key = 7.1.0.0.1.0.1.0.0.)
Current UI : inform location
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : <n/a>

Test Case 64 (Key = 8.1.0.0.1.0.1.0.0.)
Current UI : inform verification code
선구매 여부 : 선구매 진행
선택한 아이템의 재고 : <n/a>
판매하는 아이템 가짓수 : <n/a>
선구매코드 broadcast : Success
Button : <n/a>
Page Move : 확인
Modify item : <n/a>
Verification Code : <n/a>

2.1.2 Description & 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 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0미만으로 변경하는 경우	pass
33	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0으로 변경하는 경우	pass

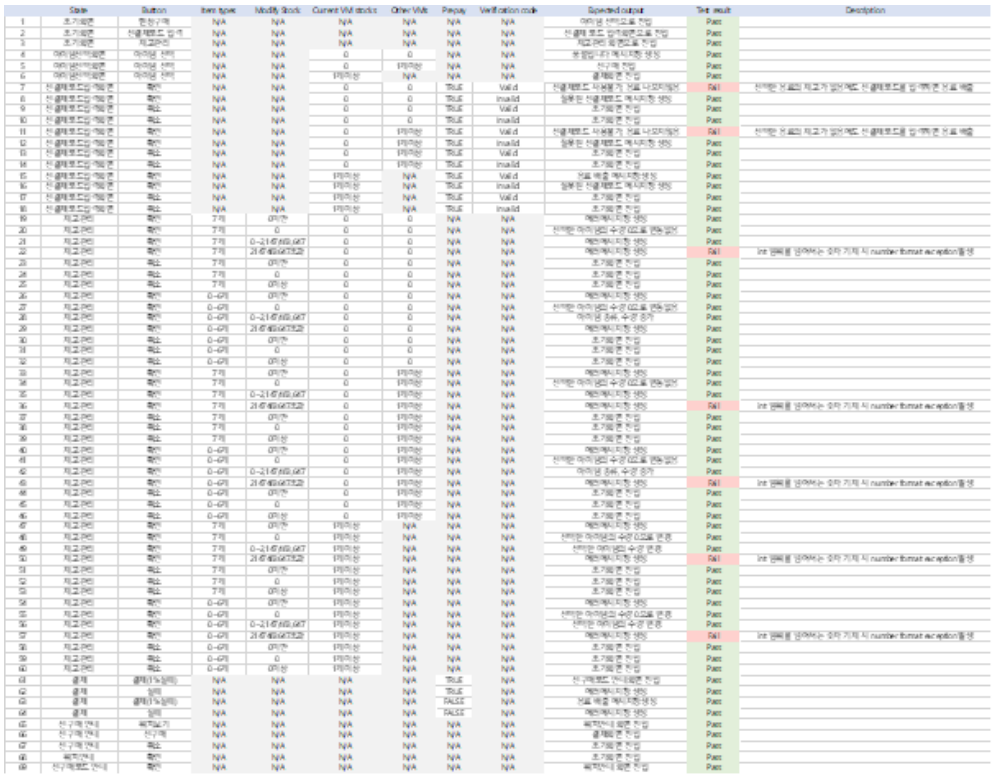
34	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0이상으로 변경하는 경우	Fail 1. int 범위를 넘어서는 숫자 기재 시 number format exception발생.
35	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 0미만으로 설정하고 취소 선택	pass
36	재고관리에서 판매하고 있는 아이템이 7가지 인 경우 현재자판기에는 재고가 없고 다른자판기에 재고가 있는 아이템의 수량을 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

2.1.3 Data Specification



- 원본 파일 링크 : https://docs.google.com/spreadsheets/d/1CORkEdazr1w5t6sUXgz1lqjMUp-OdyqOJdf3gWSpu6I/edit?usp=sharing

2.2 Brute Force Testing

2.2.1 BFT Specification & Result

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 시작되었을대 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	PASS	PASS

		돌아가는지 확인		
	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	

47개 / 49개 = 96%

2.2.2 FAIL된 항목 2개의 Input & Environment

▼ No. 3-1

- FAIL 사유
  - 음료의 종류가 7가지에서 6가지로 바뀌었을 때 새로운 종류의 음료 재고의 추가가 안 됨
- 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

---

3. Static Analysis

3.1 Cyclomatic Complexity

- Cyclomatic Complexity를 줄이는 방법
  - 메서드를 분리하고 if문을 줄인다
    - https://stackoverflow.com/questions/16418038/how-to-reduce-cyclomatic-complexity
  - 복잡한 if문에 대하여, 이를 for문과 함께 사용하여 if문을 단순화 시킨다
    - https://stackoverflow.com/questions/42224595/how-to-reduce-cyclomatic-complexity-in-a-if-condition
  - 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

- 복잡도 분석 및 솔루션 가이드라인
  - Launcher.java
    - 메서드 : public static void main(String[] args)
      - 복잡도가 높은 원인 : 코드가 너무 김 (약 60줄)
      - 솔루션 : 메서드를 분리할 것
        - 예시
          - before : complexity 28

```
public class Launcher {
  public static void main(String[] args) {

    ArrayList<Controller> DVMs = new ArrayList<>();
    for(int i = 0; i < 5; i++) {
      DVMs.add(null);
    }
    String[] locations = {"신공학관 1층", "새천년관1층", "학생회관1층", "법학관1층", "도서관1층"};
    List<Item> items = new ArrayList<>();
    List<DVM> dvms = new ArrayList<>();
    for(int i = 0; i < 20; i++) {
      items.add(null);
    }
    for(int i = 0; i < 5; i++) {
      dvms.add(null);
    }

    /* 재고 설정 */
    for(int i = 0; i < 5; i++) {
      int totalCount = 0; // 현재 DVM의 음료의 종류의 개수
      Random rnd = new Random();
      while(totalCount < 7) {
        int cur = rnd.nextInt(20); // 음료 번호 선택
        if(items.get(cur) != null) {
          if(items.get(cur).getCount(i) != null) {
            continue;
          }
          else {
            //다른자판기에 있는거임
            int count = rnd.nextInt(9)+1;
            Item item = items.get(cur);
            item.setCount(i, count);
            totalCount++;
            System.out.println(i + "에 " + Item.names[cur] + " 음료를 " + count + "만큼 설정");
          }
        } else {
          int count = rnd.nextInt(9)+1;
          Item item = new Item(cur, count, Item.prices[cur], i);
          /* FOR STUB */
          items.set(cur, item);
          System.out.println(i + "에 " + Item.names[cur] + " 음료를 " + count + "만큼 설정");
          /* FOR STUB */
          totalCount++;
        }
      }
    }
    /* 재고 설정 */
    for(int i = 0; i < dvms.size(); i++) {
      dvms.set(i, new DVM(i, locations[i], items));
```

```
    }
    for(int i = 0; i < 5; i++) {
      DVMs.set(i,Controller.getInstance(i));
      System.out.println(i);
      DVMs.get(i).setDVMId(i);
      DVMs.get(i).initialize(dvms, items);
    }
    for(int i = 0; i < 5; i++) {
      VerificationCode.getInstance(i).setDVMId(i);
    }

    /* 재고 확인 */
    for(int i = 0; i < 20; i++) {
      if(items.get(i) != null)
        items.get(i).printCount();
    }
    /* 재고 확인 */
  }
}
```

- after : complexity 13

```
public class Launcher {
  static ArrayList<Controller> DVMs = new ArrayList<>();
  static List<Item> items = new ArrayList<>();
  static List<DVM> dvms = new ArrayList<>();
  static String[] locations = {"신공학관 1층", "새천년관1층", "학생회관1층", "법학관1층", "도서관1
층"};

  public static void main(String[] args) {
    methodA();
    methodB();
    methodC();
    methodD();
  }

  static void methodA(){
    for(int i = 0; i < 5; i++) {
      DVMs.add(null);
    }
    for(int i = 0; i < 20; i++) {
      items.add(null);
    }
    for(int i = 0; i < 5; i++) {
      dvms.add(null);
    }
  }

  static void methodB(){
    /* 재고 설정 */
    for(int i = 0; i < 5; i++) {
      int totalCount = 0; // 현재 DVM의 음료의 종류의 개수
      Random rnd = new Random();
      while(totalCount < 7) {
        int cur = rnd.nextInt(20); // 음료 번호 선택
        if(items.get(cur) != null) {
          if(items.get(cur).getCount(i) != null) {
            continue;
          }
          else {
            //다른자판기에 있는거임
            int count = rnd.nextInt(9)+1;
            Item item = items.get(cur);
```

```
            item.setCount(i, count);
            totalCount++;
            System.out.println(i + "에 " + Item.names[cur] + " 음료를 " + count + "만큼 설
정");
          }
        } else {
          int count = rnd.nextInt(9)+1;
          Item item = new Item(cur, count, Item.prices[cur], i);
          /* FOR STUB */
          items.set(cur, item);
          System.out.println(i + "에 " + Item.names[cur] + " 음료를 " + count + "만큼 설
정");
          /* FOR STUB */
          totalCount++;
        }
      }
    }
  }

  static void methodC(){
    /* 재고 설정 */
    for(int i = 0; i < dvms.size(); i++) {
      dvms.set(i, new DVM(i, locations[i], items));
    }
    for(int i = 0; i < 5; i++) {
      DVMs.set(i,Controller.getInstance(i));
      System.out.println(i);
      DVMs.get(i).setDVMId(i);
      DVMs.get(i).initialize(dvms, items);
    }
    for(int i = 0; i < 5; i++) {
      VerificationCode.getInstance(i).setDVMId(i);
    }
  }

  static void methodD(){
    /* 재고 확인 */
    for(int i = 0; i < 20; i++) {
      if(items.get(i) != null)
        items.get(i).printCount();
    }
  }
}
```

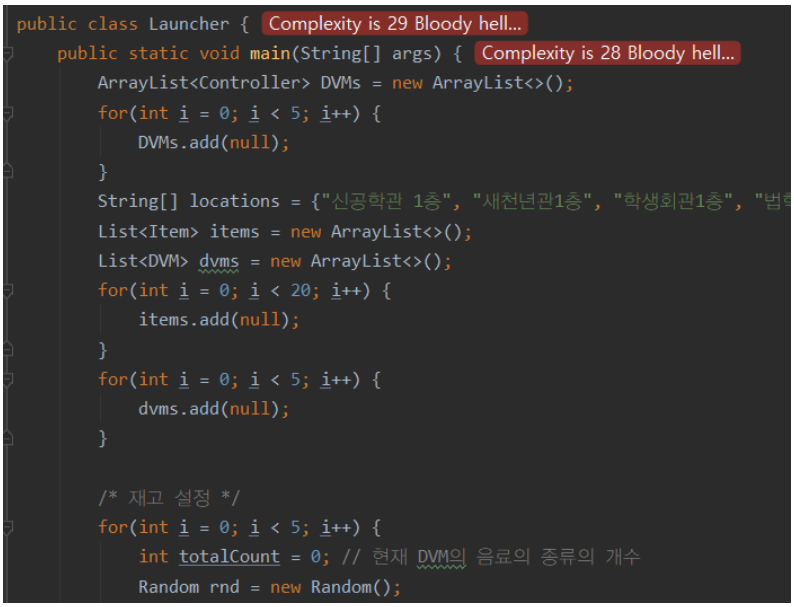
- Controller.java
  - 복잡도 문제 없음
- DVM.java
  - 복잡도 문제 없음
- Item.java
  - 복잡도 문제 없음
- Message.java
  - 메서드 : private Map<Integer, String> requestTo()
    - 복잡도 : 53
    - 복잡도가 높은 원인 : 메시지 타입에 따른 action이 분리되어 있지 않음

        - 솔루션 : 메시지 타입에 따른 action들을 별도의 메서드로 분리
- Purchase.java
    - 복잡도 문제 없음
- VerificationCode.java
    - 메서드 : public Boolean resetCode(Integer code, Boolean broadcast)
        - 복잡도가 높은 원인 : 불필요한 if문
        - 솔루션 : if문 제거
            - 다음 코드 제거

```
if(isDone)
  break;
```

        - 비고 : 성능과 복잡도 간에 trade-off가 있어서, 굳이 없애지 않아도 될 것 같긴 함

- Intellij Plugin (CodeMetrics) 화면
    - before



    - after

```java
public class Launcher {  Complexity is 13 You must be kidding
    static ArrayList<Controller> DVMs = new ArrayList<>();
    static List<Item> items = new ArrayList<>();
    static List<DVM> dvms = new ArrayList<>();
    static String[] locations = {"신공학관 1층", "새천년관1층",

    public static void main(String[] args) {
        methodA();
        methodB();
        methodC();
        methodD();
    }
```

```java
static void methodA(){  Complexity is 7 It's time to do something...
    for(int i = 0; i < 5; i++) {
        DVMs.add(null);
    }
    for(int i = 0; i < 20; i++) {
        items.add(null);
    }
    for(int i = 0; i < 5; i++) {
        dvms.add(null);
    }
}
```

```java
static void methodB(){  Complexity is 12 You must be kidding
    /* 재고 설정 */
    for(int i = 0; i < 5; i++) {
        int totalCount = 0; // 현재 DVM의 음료의 종류의 개수
        Random rnd = new Random();
        while(totalCount < 7) {
            int cur = rnd.nextInt( bound: 20); // 음료 번호
            if(items.get(cur) != null) {
                if(items.get(cur).getCount(i) != null) {
                    continue;
                }
                else {
                    //다른자판기에 있는거임
                    int count = rnd.nextInt( bound: 9)+1;
                    Item item = items.get(cur);
```

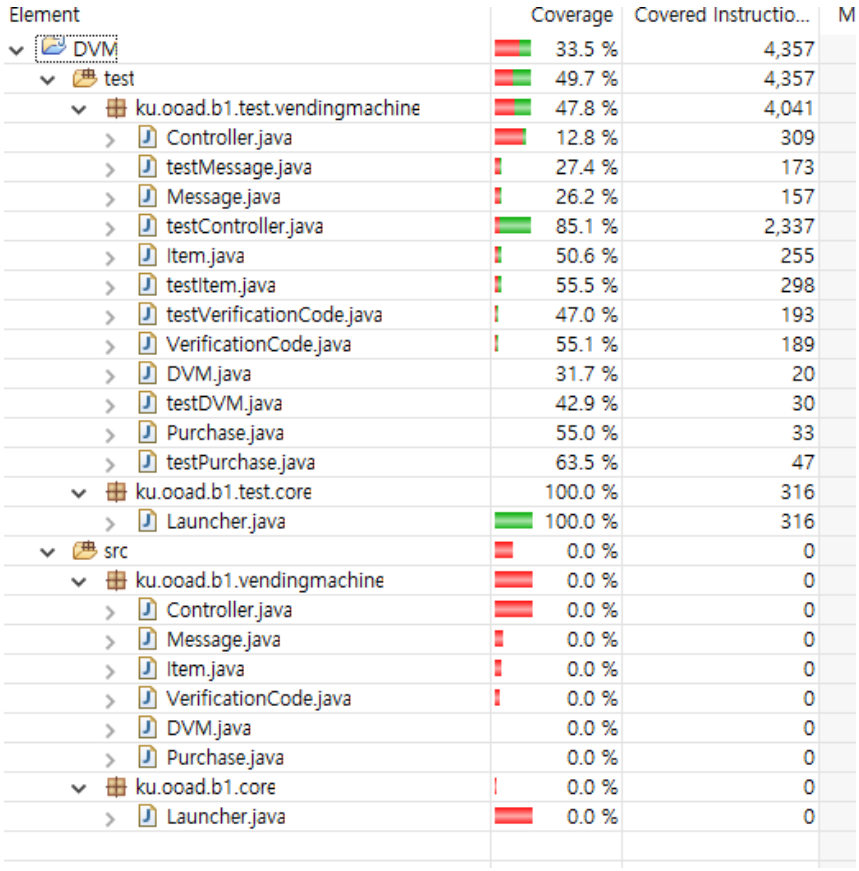
```java
static void methodC(){ Complexity is 7 It's time to do something...
    /* 재고 설정 */
    for(int i = 0; i < dvms.size(); i++) {
        dvms.set(i, new DVM(i, locations[i], items));
    }
    for(int i = 0; i < 5; i++) {
        DVMs.set(i,Controller.getInstance(i));
        System.out.println(i);
        DVMs.get(i).setDVMId(i);
        DVMs.get(i).initialize(dvms, items);
    }
    for(int i = 0; i < 5; i++) {
        VerificationCode.getInstance(i).setDVMId(i);
    }
}
```

```java
static void methodD(){ Complexity is 5 Everything is cool!
    /* 재고 확인 */
    for(int i = 0; i < 20; i++) {
        if(items.get(i) != null)
            items.get(i).printCount();
    }
}
```

3.2 Code Coverage

코드 커버리지는 기본적으로 Jacoco 툴을 중심으로 사용하였다

3.2.1 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	

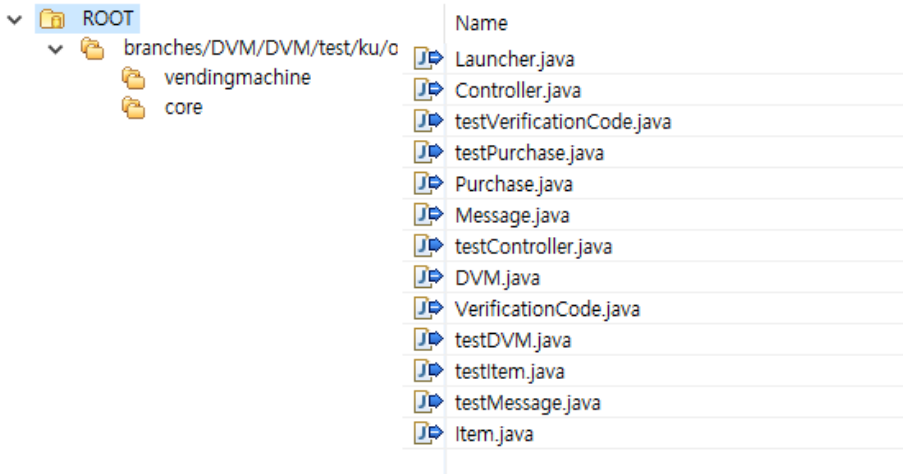
테스트를 위한 클래스와 메인 프로그램 실행에 대한 구분이 안되있어 실질적인 커버리지는 0% 였으며, 유닛테스트 폴더내의 커버리지도 49.7%로 상당히 미달됨을 확인할 수 있었습니다.

실질적인 커버리지를 올리기위하여 메인클래스들과 겹치게 생성되있는 테스트폴더내의 클래스를 모두 삭제하고

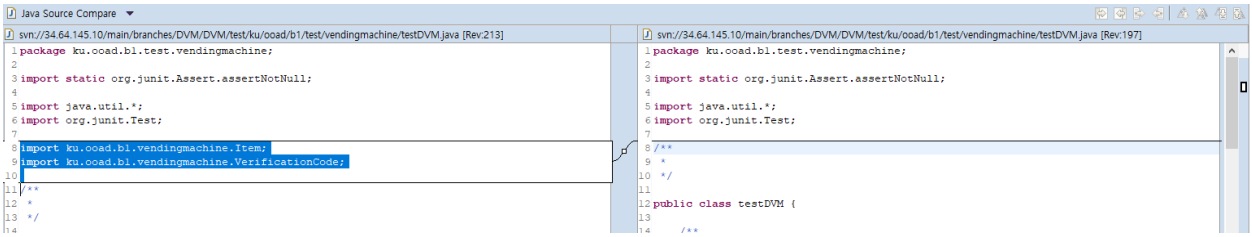
DVM

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...

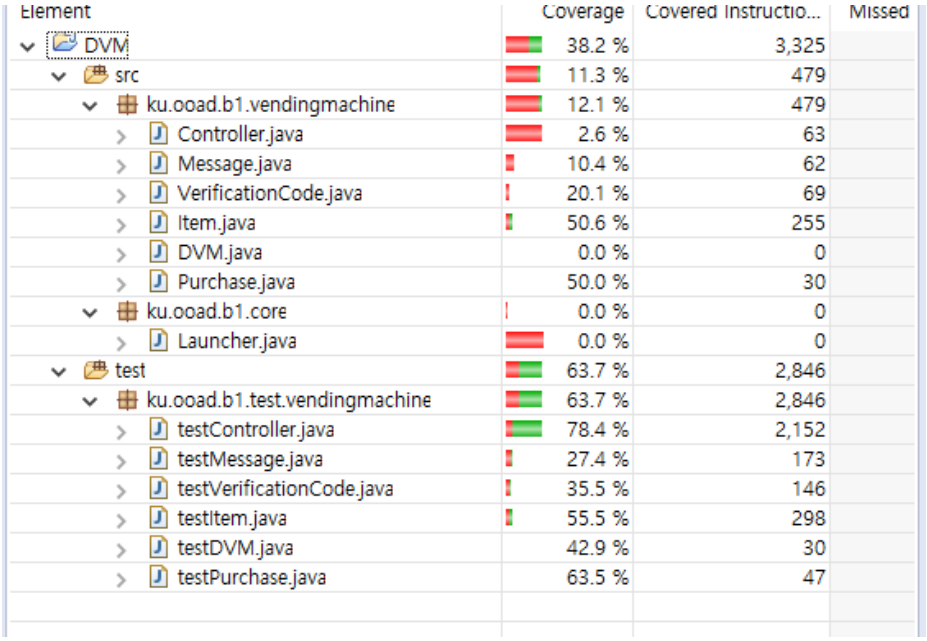
코드 커버리지 계산을 위한 수정



동시에 메인과 유닛테스트클래스를 연결해주는 임포트를 추가하였습니다.



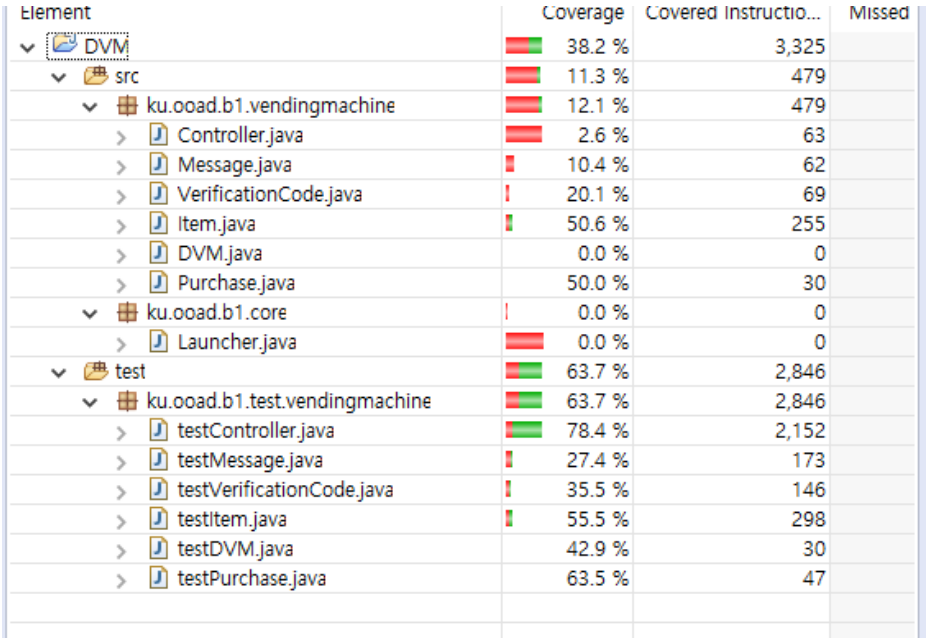
앞선 솔루션으로 달라진 커버리지는



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	

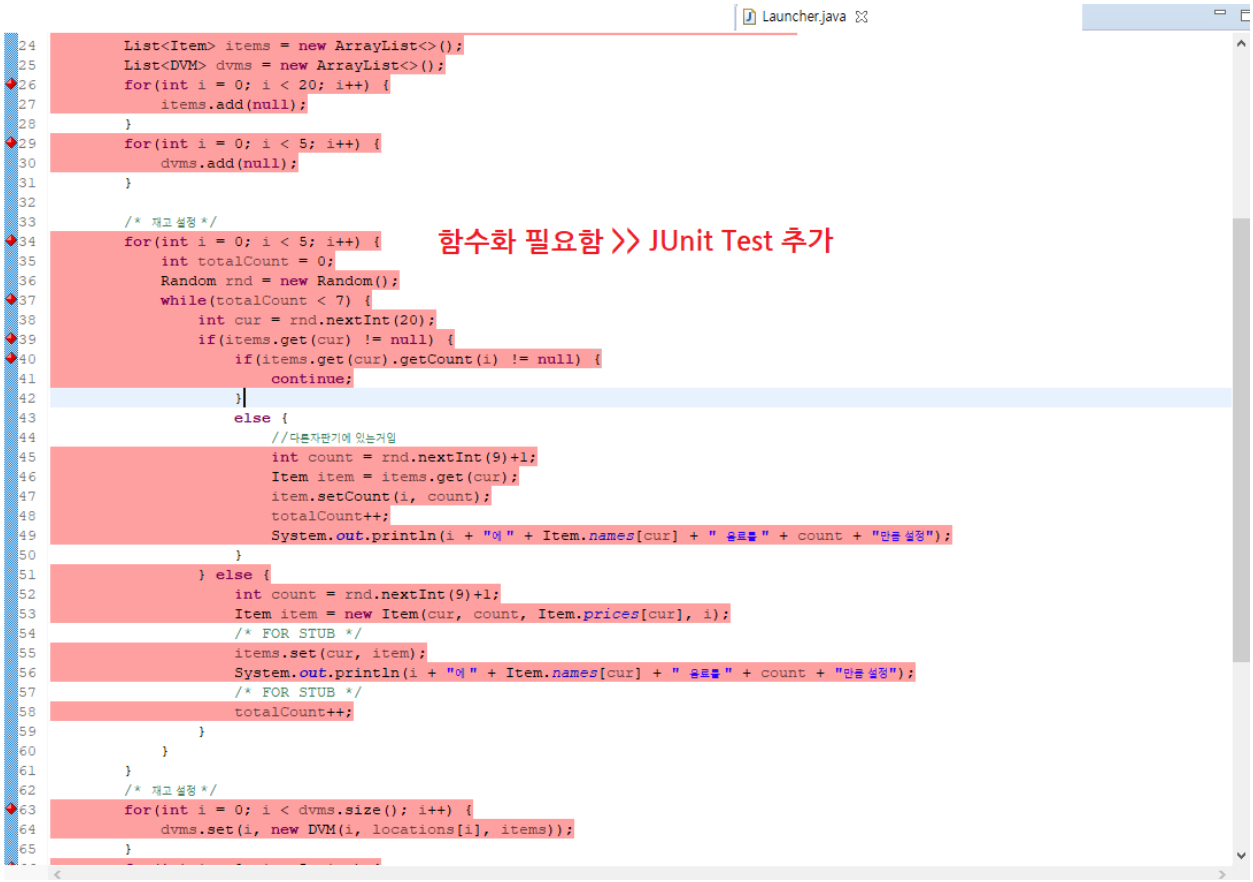
다음과같이 전체 커버리지가 38.2%, 메인클래스들의 커버리지가 11.3%로 증가한것을 확인 할 수 있습니다.

3.2.2 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,




```
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++) {
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] + " 음료를 " + 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] + " 음료를 " + 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          }
```

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

이를 3학년팀에 트렐로로 이슈를 등록하여 수정하게끔 요청한 상태입니다.

[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

3.2.3 Condition Coverage 측정 결과

1) 특정 배열을 비웠을때의 Coverage

```
public static String names[] = {"클라","사이다", "환타", "스프라이트", "켄시",
        "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[] = {};
```

```
388
389         if(dvms != null) {
390             String[] locations = {};
391             for(int i = 0; i < dvms.length; i++) {
```

위와 같이 가격들을 비우고 장소들을 비웠을 때에 대한 Coverage를 재보았는데

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 같은 커버리지를 확인할수있었고, 이에 대한 코드를 분석해보았습니다.

```
386         String[] alpha = {"0", "1", "2", "3", "4"};
387         assertThat(dvms, is(alpha));
388
389         if(dvms != null) {
390             String[] locations = {"신공학관 1층", "새천년관1층", "학생회관1층", "법학관1층", "도서관1층"};
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
```

```
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);
```

배열이 비어있음에 따라 적절하게 커버리지가 빠지는것을 볼 수 있었습니다.

```
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
```

```
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       */
```

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

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("음료 : ");
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);
```

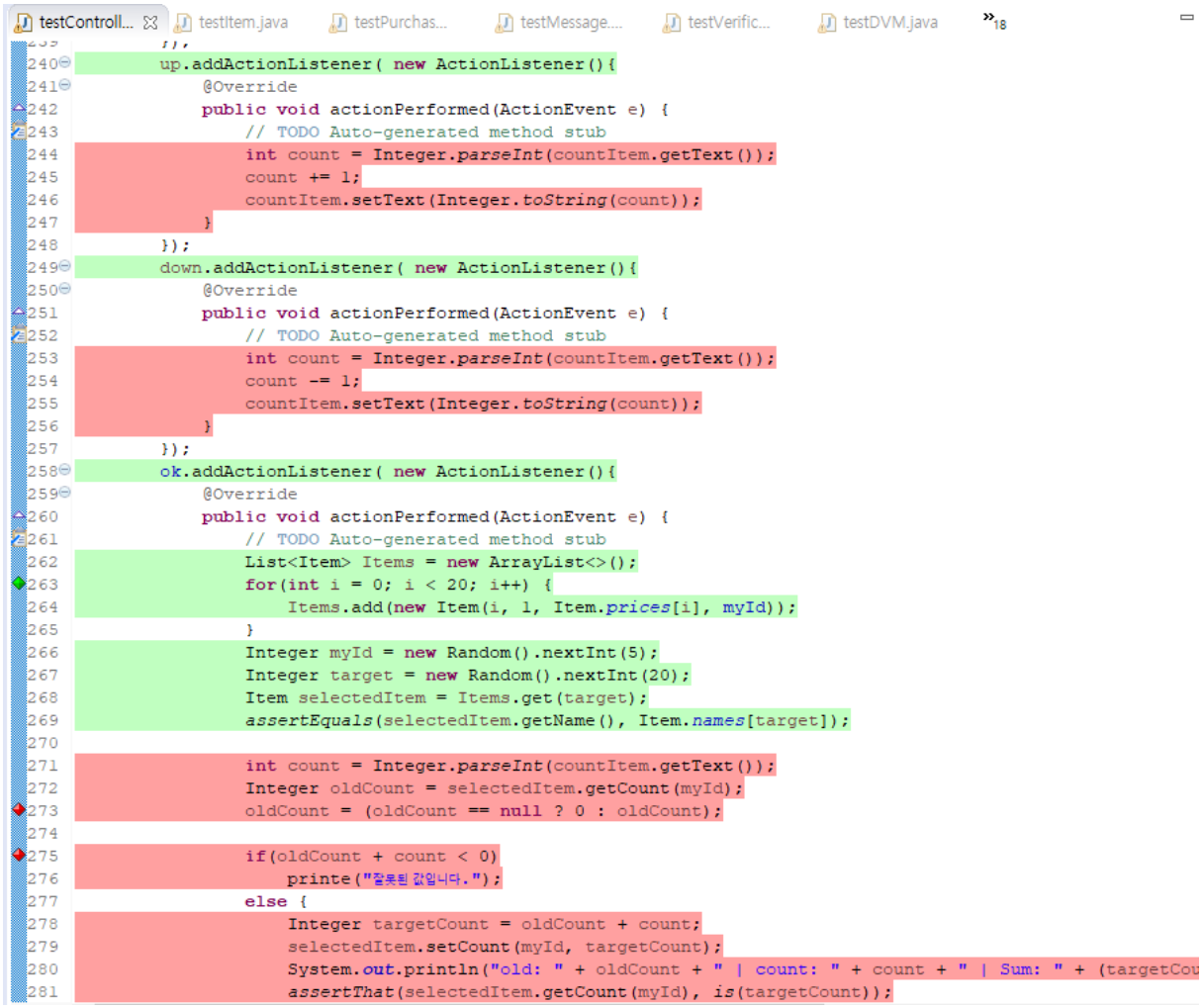
기존코드는 1이었을때의 코드 커버리지는 코드를 봤을때

```
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));
```

매우 커버리지가 잘되는편이었으나, 아래와 같이 값을 10000000000000000000000000로 주었을때는 코드 커버리지 수치부터 하락하며, 코드자체 또한 커버가 제대로 되지않음을 확인할 수 있었습니다.

```java
    @Test
    public void updateStockInfo() {
        JFrame UI = new JFrame();
        JPanel manageStock = new JPanel();
        manageStock.setLayout(new GridLayout(3,1));

        JPanel manageItems = new JPanel();
        JPanel manageEtc = new JPanel();
        JPanel manageButton = new JPanel();
        JPanel updown = new JPanel(new GridLayout(1,0));

        JLabel itemName = new JLabel();
        String itemList[] = Item.names;
        JComboBox JItemList = new JComboBox<String>(itemList);
        itemName.setText("음료 : ");

        JLabel itemCount = new JLabel();
        JTextField countItem = new JTextField(6);
        itemCount.setText("수량 : ");
        countItem.setText("10000000000000000000000000");

        JButton up = new JButton("+");
        JButton down = new JButton("-");

```

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



이는 물론 특정 Condition에 따라 커버리지가 안되는걸로 보일 수 있지만 Integer의 범위에 대한 예외처리가 안된상태에서 유닛테스트를 진행하여 일어난 이슈이기때문에 해당 이슈 또한 트렐로에 기재하였습니다.

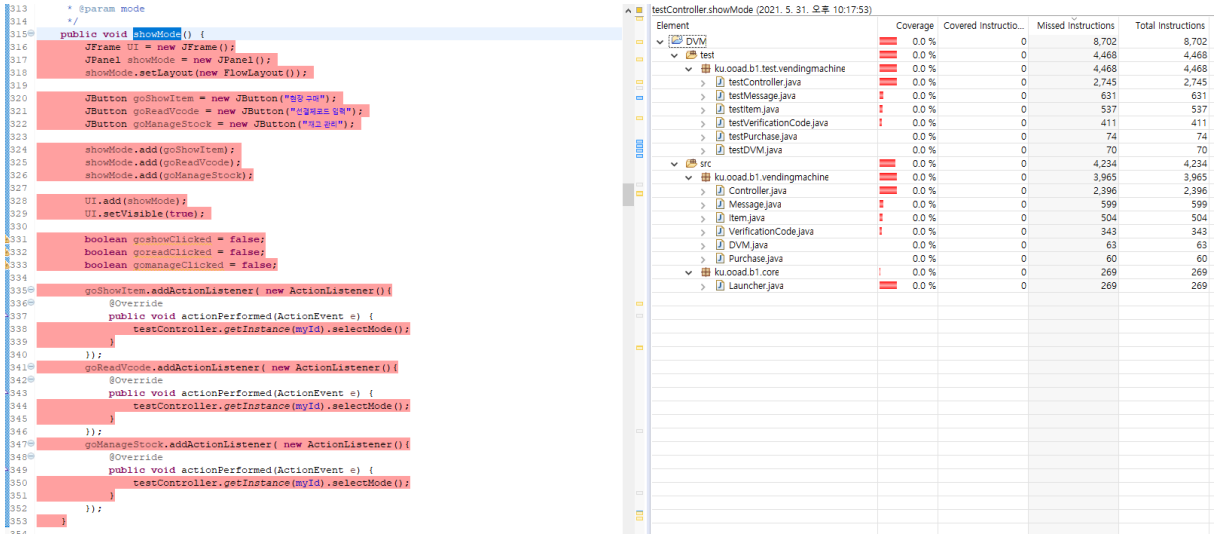
3.2.4 Statement Coverage 측정 결과

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 어떤 컨트롤러나 클래스를 제외하고 하여야 될지 의문이었습니다.

2) showMode



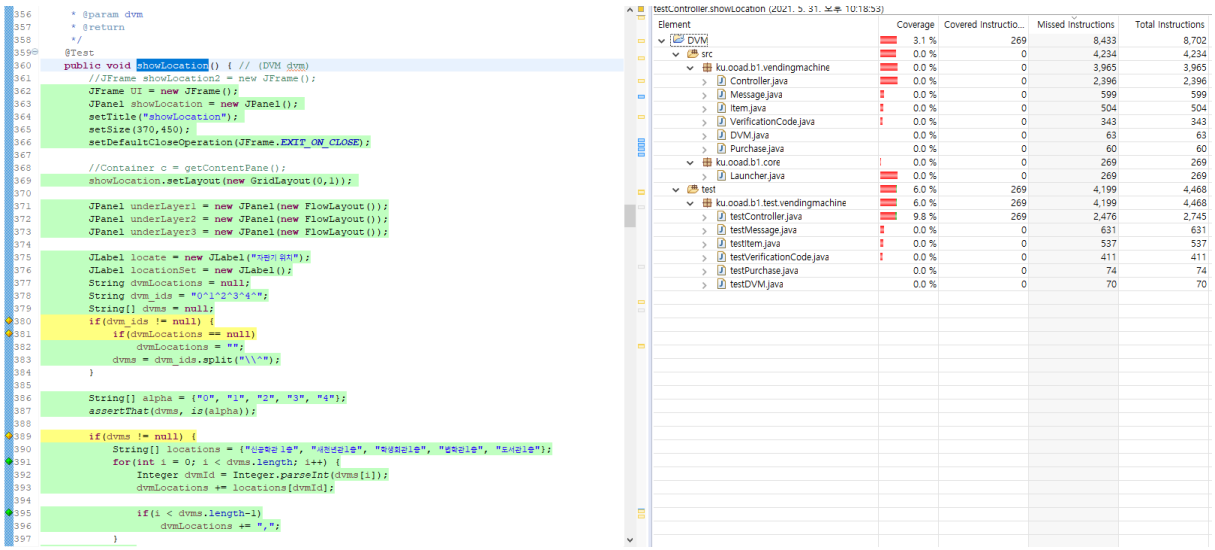
```
313     * @param mode
314     */
315    public void showMode() {
316        JFrame UI = new JFrame();
317        JPanel showMode = new JPanel();
318        showMode.setLayout(new FlowLayout());
319
320        JButton goShowItem = new JButton("현장 구매");
321        JButton goReadVcode = new JButton("선결제코드 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.

3) showLocation

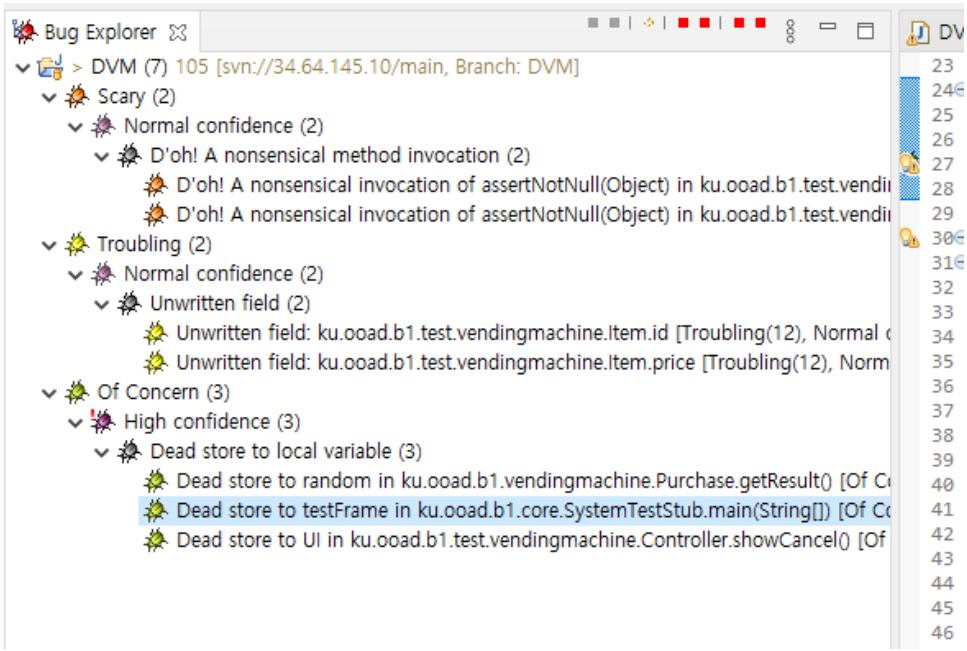


testController.showLocation (2021. 5. 31. 오후 10:18:53)

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 수정요청을 드렸습니다.

3.3 FindBugs 결과



⇒ Findbugs로 버그를 탐색해 본 결과 Fatal한 버그는 없는 것으로 확인 되었다.

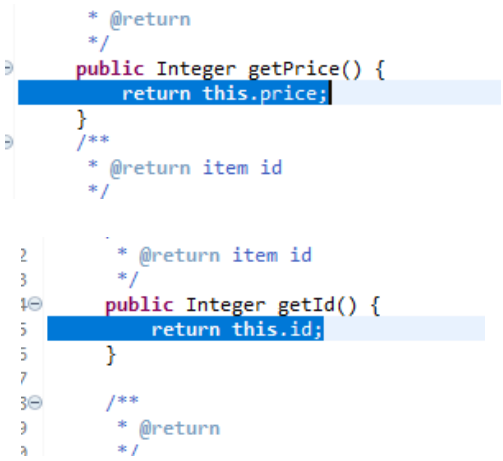
대체로 사용되지 않은 필드, 변수에 의해 문제가 될 수 있을 수도 있는 code smell이 발견 되었다.

Scary

- assertNotNull에 null 값이 아닌 String이 전달되어서 검출된 것으로 보임. 심각한 버그는 아니고 그냥 주의 정도. Java에서는 String이 Object로 읽혀지기 때문에 큰 문제는 없음.

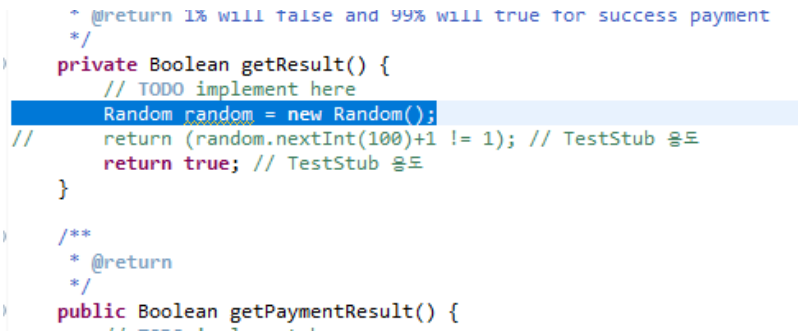
Troubling

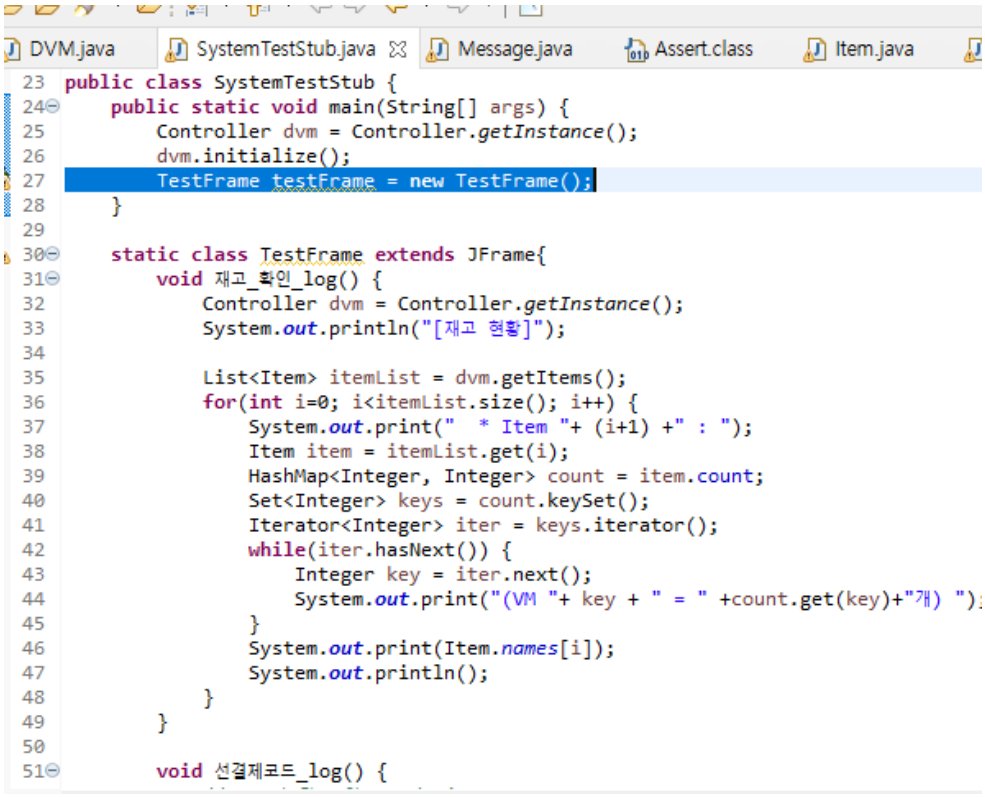
- Test에서 읽혀지지 않은 필드가 존재함.
- id와 Price가 사용되지 않음.

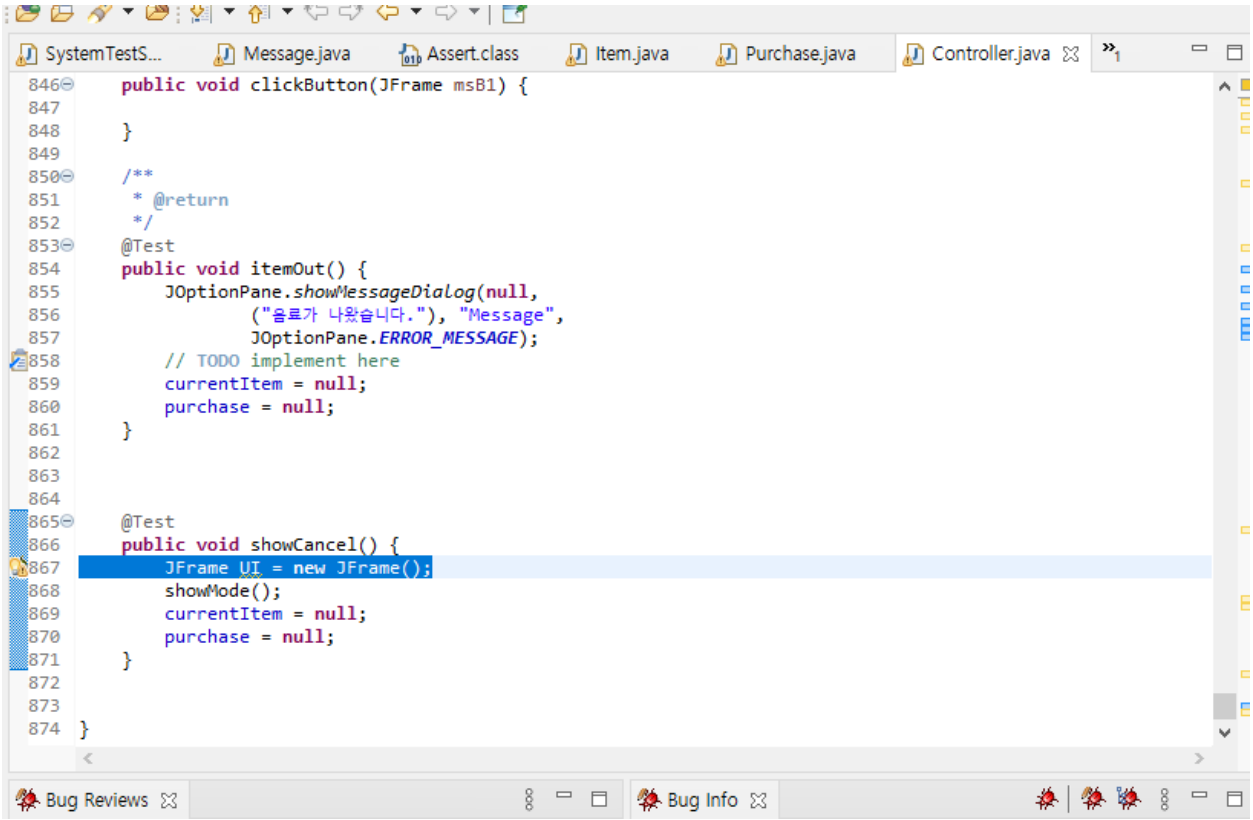


Of Concern

- Dead Store가 존재 : 만들어진 변수가 있으나 전혀 사용되지 않음. →




DVM.java
SystemTestStub.java
Message.java
Assert.class
Item.java
23 public class SystemTestStub {
24 public static void main(String[] args) {
25 Controller dvm = Controller.getInstance();
26 dvm.initialize();
27 TestFrame testFrame = new TestFrame();
28 }
29
30 static class TestFrame extends JFrame{
31 void 재고_확인_log() {
32 Controller dvm = Controller.getInstance();
33 System.out.println("[재고 현황]");
34
35 List<Item> itemList = dvm.getItems();
36 for(int i=0; i<itemList.size(); i++) {
37 System.out.print(" * Item "+ (i+1) +" : ");
38 Item item = itemList.get(i);
39 HashMap<Integer, Integer> count = item.count;
40 Set<Integer> keys = count.keySet();
41 Iterator<Integer> iter = keys.iterator();
42 while(iter.hasNext()) {
43 Integer key = iter.next();
44 System.out.print("(VM "+ key + " = " +count.get(key)+"개) ");
45 }
46 System.out.print(Item.names[i]);
47 System.out.println();
48 }
49 }
50
51 void 선결제코드_log() {


SystemTestS...
Message.java
Assert.class
Item.java
Purchase.java
Controller.java
846 public void clickButton(JFrame msB1) {
847
848 }
849
850 /**
851 * @return
852 */
853 @Test
854 public void itemOut() {
855 JOptionPane.showMessageDialog(null,
856 ("음료가 나왔습니다."), "Message",
857 JOptionPane.ERROR_MESSAGE);
858 // TODO implement here
859 currentItem = null;
860 purchase = null;
861 }
862
863
864
865 @Test
866 public void showCancel() {
867 JFrame UI = new JFrame();
868 showMode();
869 currentItem = null;
870 purchase = null;
871 }
872
873
874 }
Bug Reviews
Bug Info

4. General Review

- CPT 기준 87.5% / 100% (가중치 x 40)
- BFT 기준 96% / 100% (가중치 x 40)
- Code Coverage 6.14점 / 100점 (가중치 x 16)
- Cyclomatic Complexity 80점 / 100점 (가중치 x 4)

→ (87.5 x 40 + 96 x 40 + 6.14 x 16 + 80 x 4) / 100 = 77.5824

→ 종합적으로 77.6점 의 SW Quality로 평가