Distributed Vending Machine

OOPT : OOI (The 3rd Cycle)

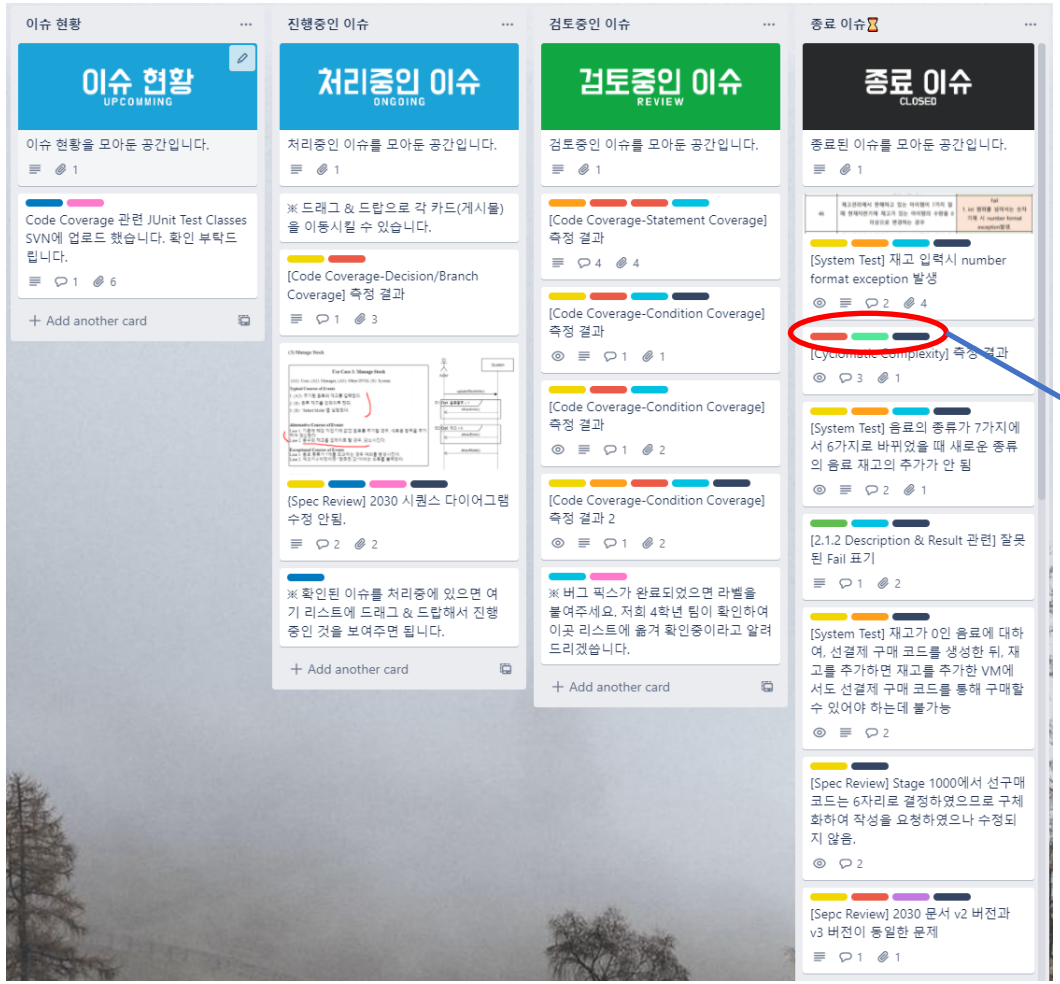
TEAM 1

정연수 김민환 이승헌 조벽정 황유란

INDEX

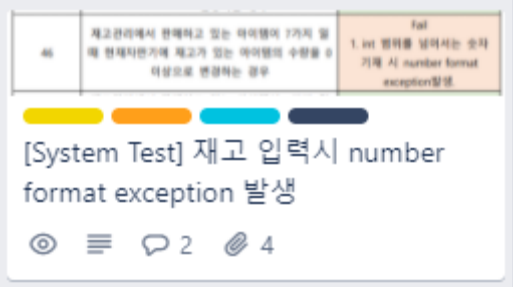
CTIP

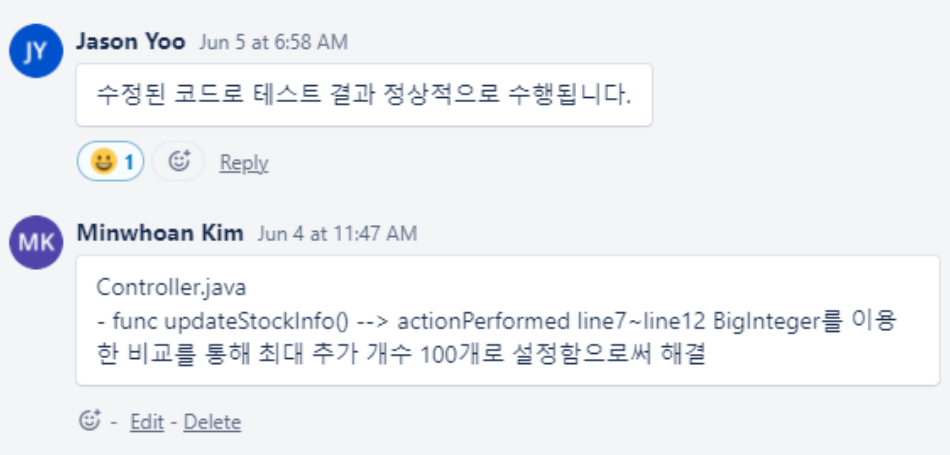
CTIP – Trello

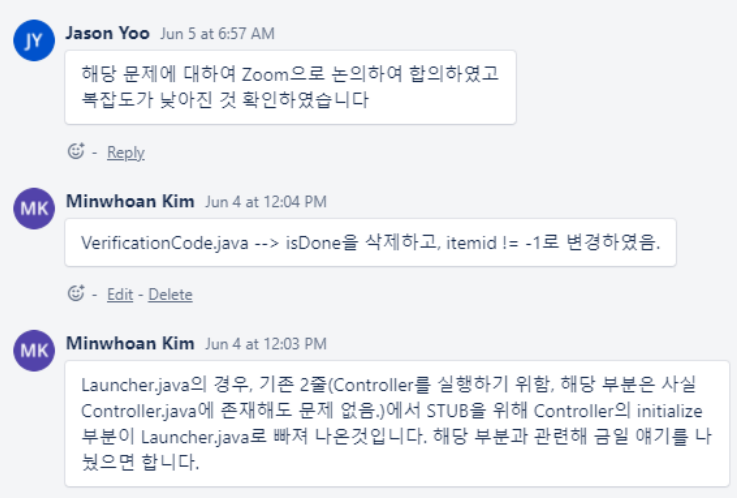

이슈 현황
이슈 현황
UPCOMMING
이슈 현황을 모아둔 공간입니다.
Code Coverage 관련 JUnit Test Classes SVN에 업로드 했습니다. 확인 부탁드립니다.
Add another card
진행중인 이슈
처리중인 이슈
ONGOING
처리중인 이슈를 모아둔 공간입니다.
※ 드래그 & 드랍으로 각 카드(게시물)을 이동시킬 수 있습니다.
[Code Coverage-Decision/Branch Coverage] 측정 결과
[Spec Review] 2030 시퀀스 다이어그램 수정 안됨.
※ 확인된 이슈를 처리중에 있으면 여기 리스트에 드래그 & 드랍해서 진행중인 것을 보여주면 됩니다.
Add another card
검토중인 이슈
검토중인 이슈
REVIEW
검토중인 이슈를 모아둔 공간입니다.
[Code Coverage-Statement Coverage] 측정 결과
[Code Coverage-Condition Coverage] 측정 결과
[Code Coverage-Condition Coverage] 측정 결과
[Code Coverage-Condition Coverage] 측정 결과 2
※ 버그 픽스가 완료되었으면 라벨을 붙여주세요. 저희 4학년 팀이 확인하여 이곳 리스트에 옮겨 확인중이라고 알려드리겠습니다.
Add another card
종료 이슈
종료 이슈
CLOSED
종료된 이슈를 모아둔 공간입니다.
[System Test] 재고 입력시 number format exception 발생
[System Test] 음료의 종류가 7가지에서 6가지로 바뀌었을 때 새로운 종류의 음료 재고의 추가가 안 됨
[2.1.2 Description & Result 관련] 잘못된 Fail 표기
[System Test] 재고가 0인 음료에 대하여, 선결제 구매 코드를 생성한 뒤, 재고를 추가하면 재고를 추가한 VM에서도 선결제 구매 코드를 통해 구매할 수 있어야 하는데 불가능
[Spec Review] Stage 1000에서 선구매 코드는 6자리로 결정하였으므로 구체화하여 작성을 요청하였으나 수정되지 않음.
[Sepc Review] 2030 문서 v2 버전과 v3 버전이 동일한 문제


LABELS
심각도 낮음
심각도 보통
버그
심각도 높음
중요
의견 제안
수정 완료
검토중
참고자료
피드백을 요청합니다.
종료 이슈
기타사항

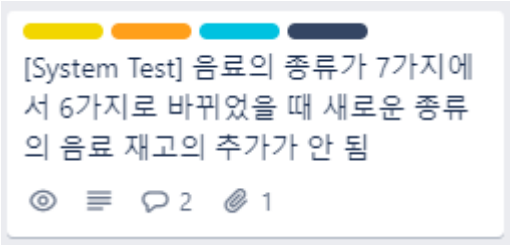
CTIP – Trello, Issue: CLOSED

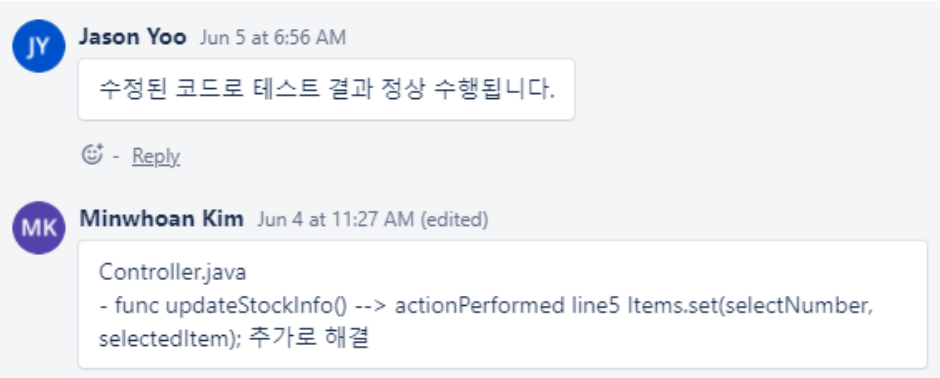


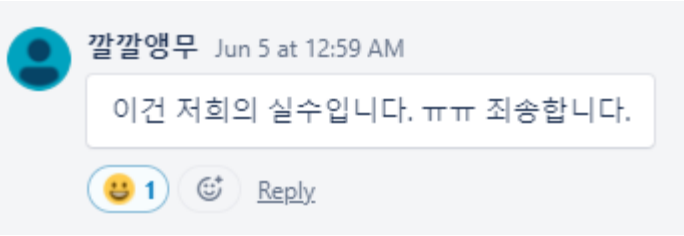




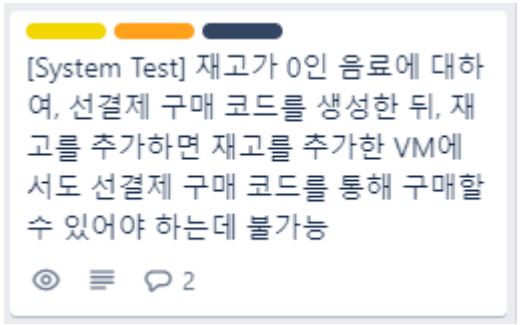
CTIP – Trello, Issue: CLOSED

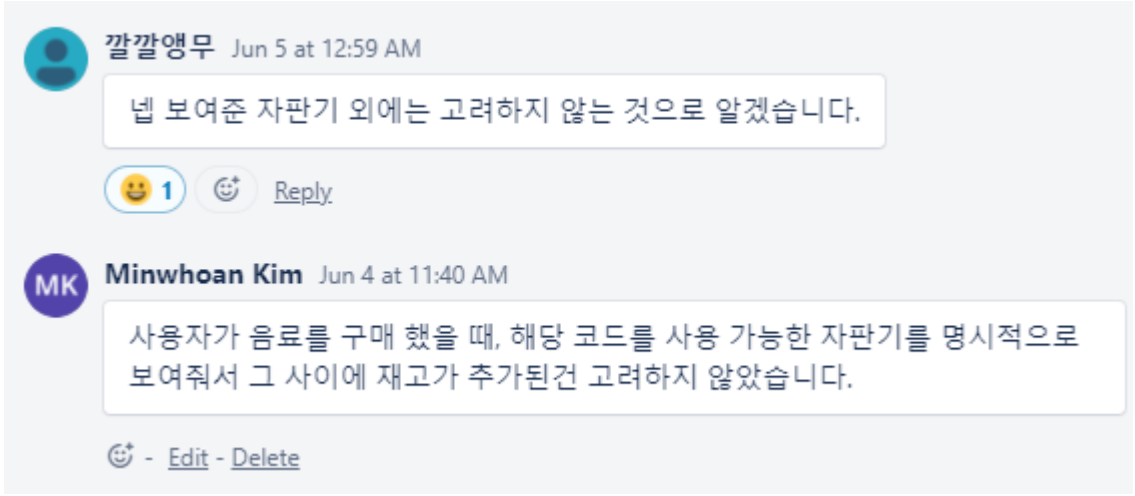


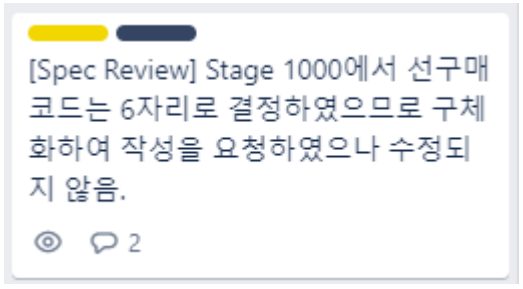


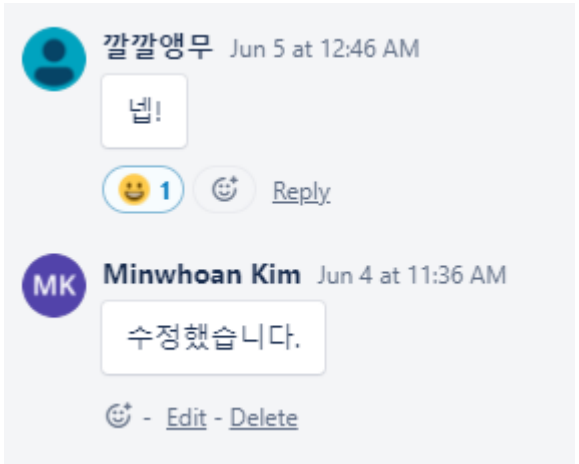


CTIP – Trello, Issue: CLOSED

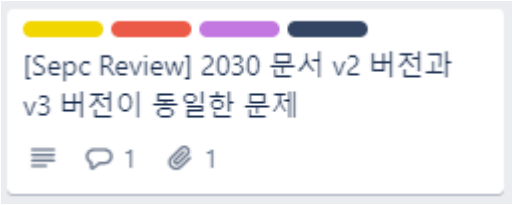




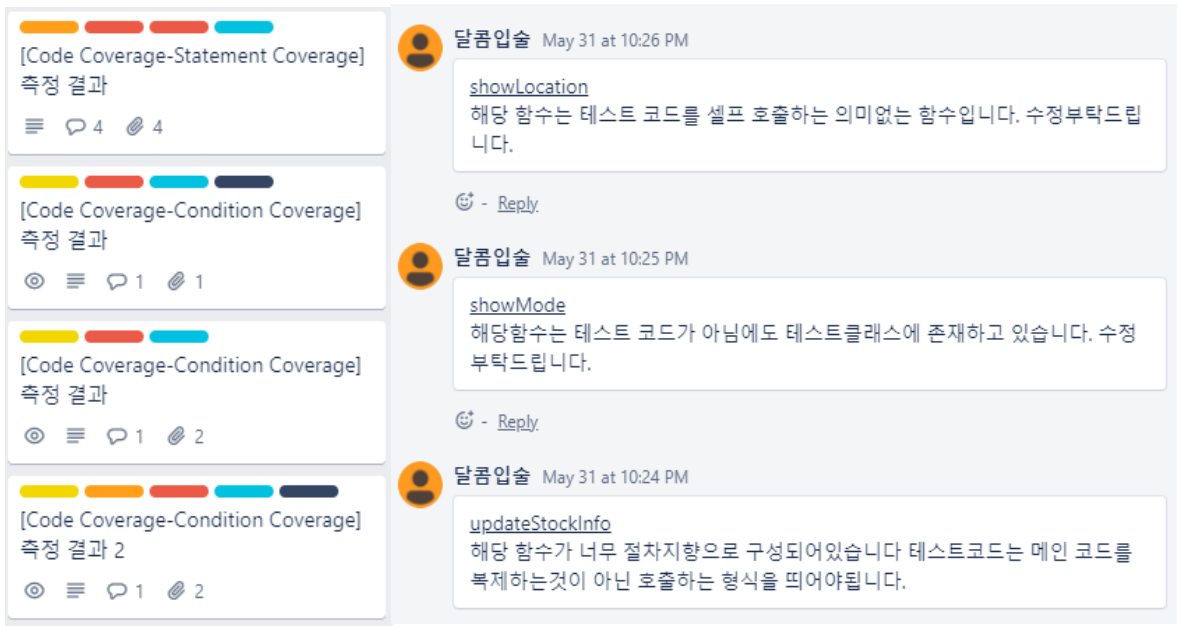


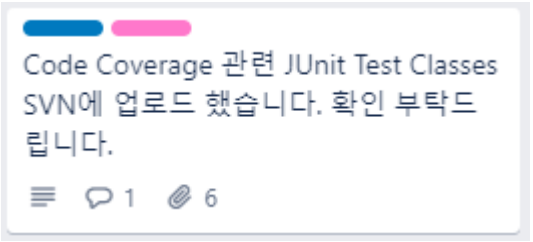


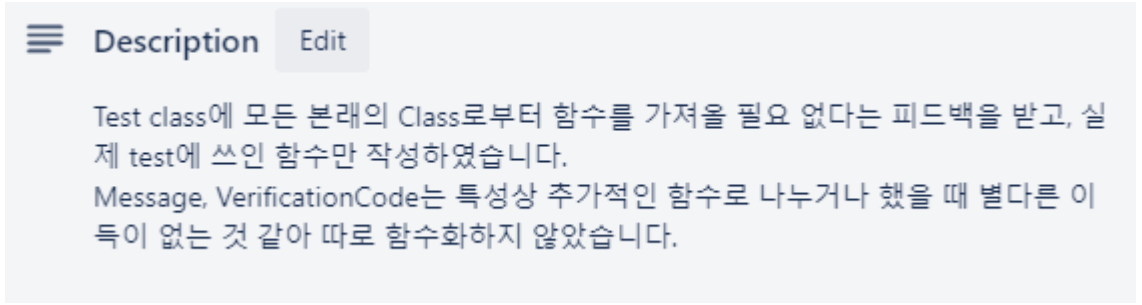
CTIP – Trello, Issue: CLOSED



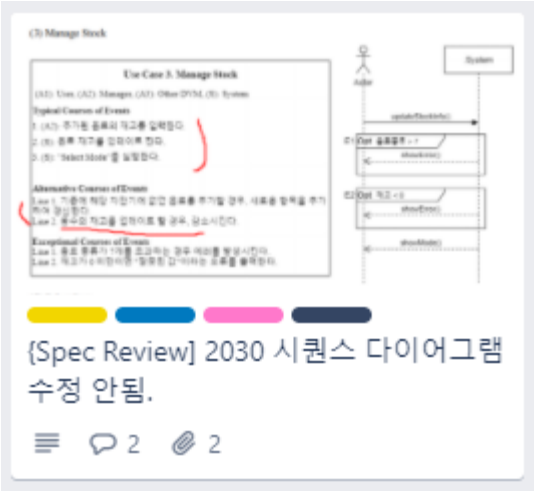
CTIP – Trello, Issue: REVIEW







CTIP – Trello, Issue: ONGOING



깔깔앵무 Jun 5 at 12:57 AM (edited)

예시로 첨부한 그 UseCase 3번의 각 Event의 내용이 시퀀스 노트와 일치가 안되어있는 것을 확인하실 수 있습니다.
같은 현상을 2, 7, 11, 18번에서도 발견하실 수 있습니다.

Reply

Yeonsoo Chung Jun 4 at 11:51 AM

시퀀스 다이어그램의 경우 2040 인터렉션 다이어그램에서 수정되었습니다.
2030 시스템 시퀀스 다이어그램 노트 수정이 필요하다고 한 부분에 관한 상세한 설명 요청드립니다.

Reply

Spec Review

Spec Review – Sequence Diagram

{Spec Review] 2030 시퀀스 다이어그램 수정 안됨.

in list 진행중인 이슈

LABELS

심각도 보통 | 수정 완료 | 피드백을 요청합니다. | 종료 이슈 | +

Description Edit

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

깔깔앵무 Jun 5 at 12:57 AM (edited)

예시로 첨부한 그 UseCase 3번의 각 Event의 내용이 시퀀스 노트와 일치가 안되어있는 것을 확인하실 수 있습니다.
같은 현상을 2, 7, 11, 18번에서도 발견하실 수 있습니다.

- Reply

Alternative courses of events 삭제

(2) Select Mode

Use Case 2. Select Mode

(A1): User, (A2): Manager, (A3): Other DVM, (S): System

Typical Courses of Events

1.1. (A1): 현장구매모드를 선택한다.
1.2. (S): "Show Item"을 실행한다.
2.1 (A1): 선결제코드 입력모드를 선택한다.
2.2 (S): "Read Verification Code"를 실행한다.
3.1 (A2): 재고 관리를 선택한다.
3.2 (S): "Manage Stock"을 실행한다.

Alternative Courses of Events

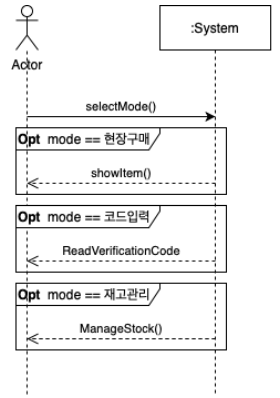
Line 1. 세 가지 모드를 동시에 선택할 경우 현장구매 모드를 실행한다.

Use Case 2. Select Mode

(A1): User, (A2): Manager, (A3): Other DVM, (S): System

Typical Courses of Events

1.1. (A1): 현장구매모드를 선택한다.
1.2. (S): "Show Item"을 실행한다.
2.1 (A1): 선결제코드 입력모드를 선택한다.
2.2 (S): "Read Verification Code"를 실행한다.
3.1 (A2): 재고 관리를 선택한다.
3.2 (S): "Manage Stock"을 실행한다.



Spec Review – Sequence Diagram

{Spec Review] 2030 시퀀스 다이어그램 수정 안됨.

in list 진행중인 이슈

LABELS

심각도 보통 | 수정 완료 | 피드백을 요청합니다. | 종료 이슈 | +

Description Edit

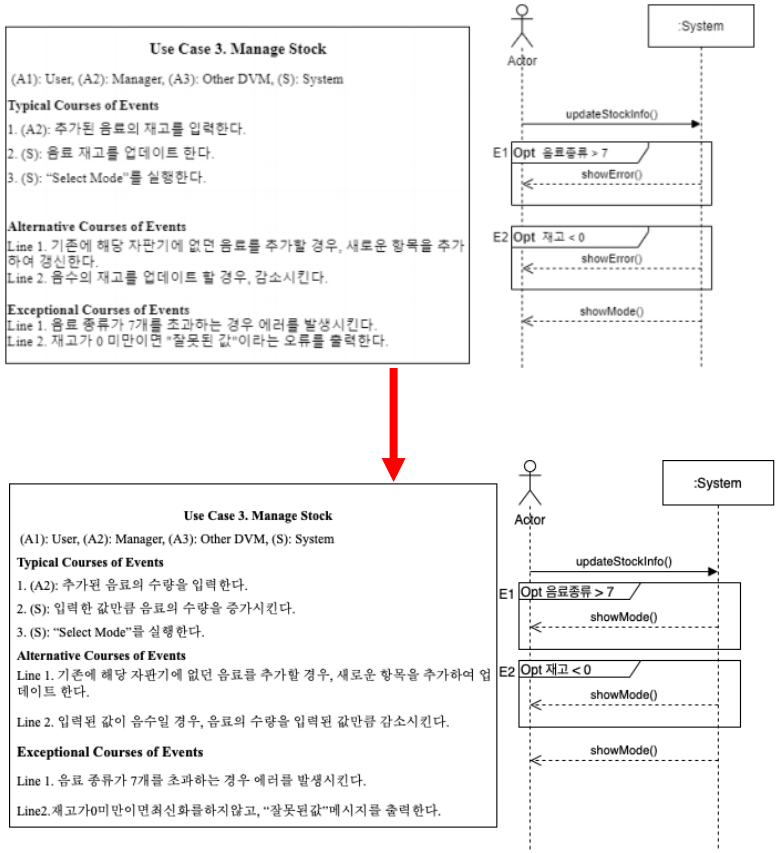
앞에서 수정한 내용에 맞게 2, 3, 7, 11, 18번 UseCase에 대한 시퀀스 다이어그램 노트 수정 필요

깔깔앵무 Jun 5 at 12:57 AM (edited)

예시로 첨부한 그 UseCase 3번의 각 Event의 내용이 시퀀스 노트와 일치가 안되어있는 것을 확인하실 수 있습니다.
같은 현상을 2, 7, 11, 18번에서도 발견하실 수 있습니다.

- Reply

Alternative courses of events 수정



Spec Review – Sequence Diagram

{Spec Review] 2030 시퀀스 다이어그램 수정 안됨.

in list 진행중인 이슈

LABELS

심각도 보통 | 수정 완료 | 피드백을 요청합니다. | 종료 이슈 | +

Description Edit

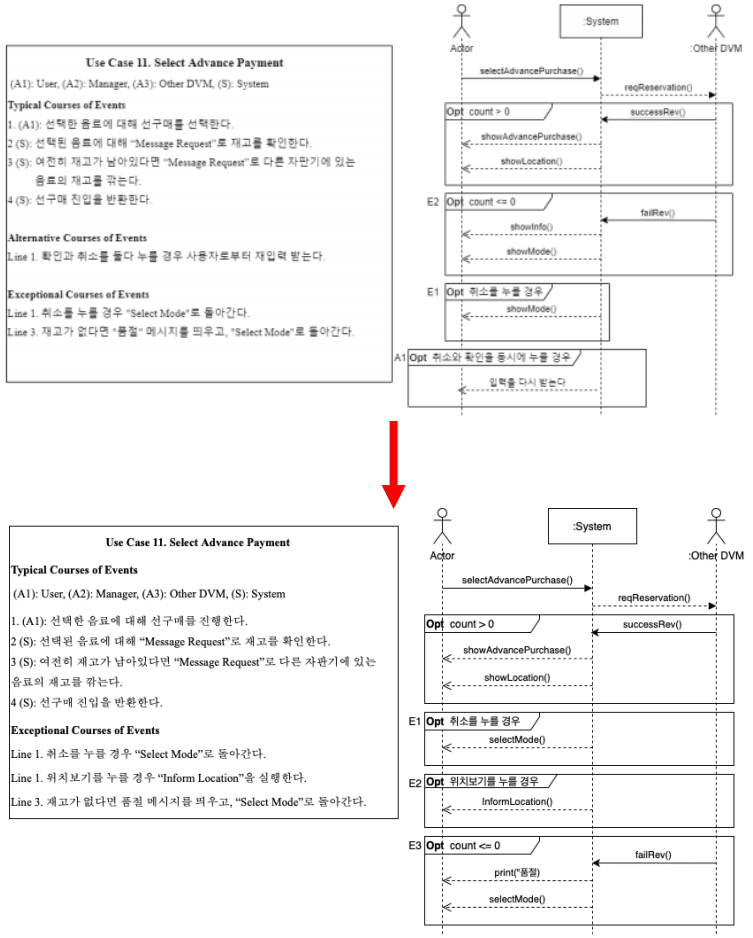
앞에서 수정한 내용에 맞게 2, 3, 7, 11, 18번 UseCase에 대한 시퀀스 다이어그램 노트 수정 필요

깔깔앵무 Jun 5 at 12:57 AM (edited)

예시로 첨부한 그 UseCase 3번의 각 Event의 내용이 시퀀스 노트와 일치가 안되어있는 것을 확인하실 수 있습니다.
같은 현상을 2, 7, 11, 18번에서도 발견하실 수 있습니다.

- Reply

Exceptional courses of events 노트 수정
Alternative courses of events 삭제



Spec Review – Sequence Diagram

{Spec Review] 2030 시퀀스 다이어그램 수정 안됨.

in list 진행중인 이슈

LABELS

심각도 보통 | 수정 완료 | 피드백을 요청합니다. | 종료 이슈 | +

Description Edit

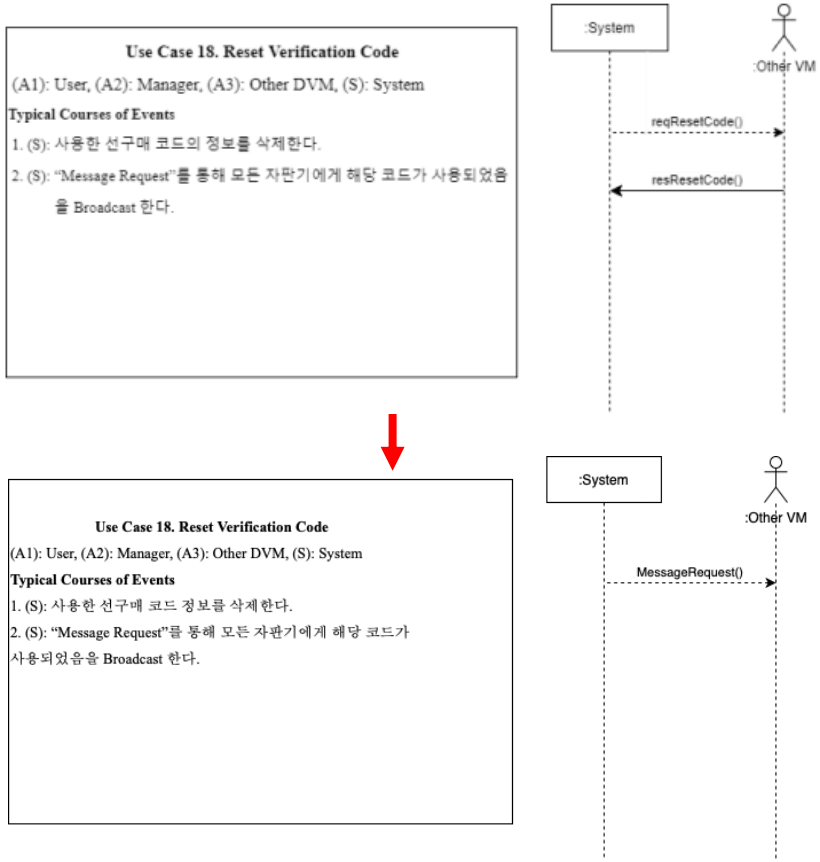
앞에서 수정한 내용에 맞게 2, 3, 7, 11, 18번 UseCase에 대한 시퀀스 다이어그램 노트 수정 필요

깔깔앵무 Jun 5 at 12:57 AM (edited)

예시로 첨부한 그 UseCase 3번의 각 Event의 내용이 시퀀스 노트와 일치가 안되어있는 것을 확인하실 수 있습니다.
같은 현상을 2, 7, 11, 18번에서도 발견하실 수 있습니다.

- Reply

System Sequence Diagram수정



Spec Review – Use Cases

- 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

Typical Courses of Events 수정

Use Case	13. Create Verification Code
Actor	None
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Hidden
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): 6자리 난수의 선구매 코드를 생성한다.
Alternative Courses of	N/A

Spec Review – Use Cases

- 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

내용 수정 완료

Use Case	16. Read Verification Code
Actor	User
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Evident
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	SelectMode에서 선구매 버튼을 누름
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (A1): 선구매 코드 6자리를 입력한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1. 취소를 누를경우 “Select Mode”로 돌아간다.

Category Partition Testing

Category Partition Testing – Result

Num	Description	Result	Remark
8	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가 0으로 떨어진 상태에서 유효한 선결제 코드를 입력후 확인선택	Failed	선택한 음료의 재고가 없음에도 선결제코드를 입력하면 음료 배출
12	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가 0으로 떨어진 상태에서 유효한 선결제 코드를 입력후 확인선택	Failed	선택한 음료의 재고가 없음에도 선결제코드를 입력하면 음료 배출
22	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Failed	Int 범위를 넘어서는 숫자 기재시 Number Format Exception 발생
28	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Failed	Int 범위를 넘어서는 숫자 기재시 Number Format Exception 발생
34	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Failed	Int 범위를 넘어서는 숫자 기재시 Number Format Exception 발생
40	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Failed	Int 범위를 넘어서는 숫자 기재시 Number Format Exception 발생
46	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Failed	Int 범위를 넘어서는 숫자 기재시 Number Format Exception 발생
52	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Failed	Int 범위를 넘어서는 숫자 기재시 Number Format Exception 발생

→ Total 56/64 (87.5%) Passed

Category Partition Test – 8, 12

해당 내용은 Spec상 정상적인 결과이고, 해당 내용을 4학년 팀에게 전달하여 PASS임을 확인함

Category Partition Test – 22, 28, 34, 40, 46, 52

```java
@Override // 수정 전
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    Item selectedItem = Items.get(JItemList.getSelectedIndex());
    if(selectedItem == null) {
        selectedItem = new Item(JItemList.getSelectedIndex(), 0, 0
    }
    int count = Integer.parseInt(countItem.getText());
    Integer oldCount = selectedItem.getCount(myId);
    System.out.println("count: " + count + " / " + oldCount);
    if(oldCount + count < 0)
        printe("잘못된 값입니다.");
    else {
        selectedItem.setCount(myId, oldCount + count);
        int cnt = 0;
        for(int i=0;i<Items.size();i++) {
            Integer cCount = 0;
            if(Items.get(i) != null)
                cCount = Items.get(i).getCount(myId);
            if(cCount != null && cCount > 0)
                cnt++;
        }
        if( cnt > 7 ) {
            selectedItem.setCount(myId, 0);
            JOptionPane.showMessageDialog(null,
                    ("7종류의 음료만 판매할 수 있습니다."), "Message",
                    JOptionPane.ERROR_MESSAGE);
        } else {
            manageStock.setVisible(false);
            showMode(UI);
        }
    }
  }
});
}
```

<수정 전>

```java
@Override // 수정 후
public void actionPerformed(ActionEvent e) {
    int selectNumber = JItemList.getSelectedIndex();
    Item selectedItem = Items.get(selectNumber);
    if(selectedItem == null) {
        selectedItem = new Item(selectNumber, 0, 0, myId);
        Items.set(selectNumber, selectedItem);
    }
    BigInteger bigI = new BigInteger(countItem.getText());
    BigInteger maxI = new BigInteger("101"); BigInteger minI = new
    if(bigI.compareTo(maxI) != -1 || bigI.compareTo(minI) == -1) {
        printe("한번에 등록할 수 있는 최대 수량은 +-100개입니다.");
        return;
    }
    int count = bigI.intValue();
    Integer oldCount = selectedItem.getCount(myId);
    oldCount = (oldCount == null ? 0 : oldCount);

    if(oldCount + count < 0)
        printe("잘못된 값입니다.");
    else {
        selectedItem.setCount(myId, oldCount + count);
        System.out.println("선택: " + selectedItem.getName() + " |
        int cnt = 0;
        for(int i=0;i<Items.size();i++) {
            Integer cCount = (Items.get(i) != null ? (Items.get(i)
            if(cCount != null && cCount > 0)
                cnt++;
        }

        if( cnt > 7 ) {
            selectedItem.setCount(myId, 0);
            JOptionPane.showMessageDialog(null,
                    ("7종류의 음료만 판매할 수 있습니다."), "Message",
                    JOptionPane.ERROR_MESSAGE);
        } else {
            manageStock.setVisible(false);
            showMode(UI);
        }
    }
  }
});
}
```

<수정 후>

분석

String으로 입력받은 값을 integer로 변환하기 때문에, int 범위를 넘어가는 숫자를 입력받는다면 Number Format Exception이 발생할 수 있음
→ BigInteger를 사용하여 한번에 관리할 수 있는 최대 / 최소 재고를 설정하여 문제를 해결함

Test 결과

Pass

Category Partition Test – 3rd cycle

Num	Description	Result	Remark
8	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가 0으로 떨어진 상태에서 유효한 선결제 코드를 입력후 확인선택	Pass	·
12	선구매 진행후 선구매코드가 broadcast되고 선택한 아이템의 전체 재고가 0으로 떨어진 상태에서 유효한 선결제 코드를 입력후 확인선택	Pass	·
22	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Pass	·
28	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Pass	·
34	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Pass	·
40	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Pass	·
46	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Pass	·
52	재고관리에서 판매하고 있는 아이템이 7가지인 경우 / 전체재고가 0인 아이템의 수량을 0 이상으로 변경하는 경우	Pass	·

→ Total 64/64 (100%) Passed

Brute Force Test

Brute Force Test – Result

Use case	Number	Test case	Result
Manage Stock	3-1	Manager가 갱신한 재고 정보가 적용되는지 확인	Failed
Read Verification Code	14-2	기록된 코드가 유효하다면 "Item Out"을 실행하는지 확인	Failed

→ Total 47/49 (96%) Passed

Brute Force Test – Manage Stock

```java
@Override // 수정 전
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    Item selectedItem = Items.get(JItemList.getSelectedIndex());
    if(selectedItem == null) {
        selectedItem = new Item(JItemList.getSelectedIndex(), 0, 0
    }
    int count = Integer.parseInt(countItem.getText());
    Integer oldCount = selectedItem.getCount(myId);
    System.out.println("count: " + count + " / " + oldCount);
    if(oldCount + count < 0)
        printe("잘못된 값입니다.");
    else {
        selectedItem.setCount(myId, oldCount + count);
        int cnt = 0;
        for(int i=0;i<Items.size();i++) {
            Integer cCount = 0;
            if(Items.get(i) != null)
                cCount = Items.get(i).getCount(myId);
            if(cCount != null && cCount > 0)
                cnt++;
        }
        if( cnt > 7 ) {
            selectedItem.setCount(myId, 0);
            JOptionPane.showMessageDialog(null,
                    ("7종류의 음료만 판매할 수 있습니다."), "Message",
                    JOptionPane.ERROR_MESSAGE);
        } else {
            manageStock.setVisible(false);
            showMode(UI);
        }
    }
  }
});
}
```

<수정 전>

```java
@Override // 수정 후
public void actionPerformed(ActionEvent e) {
    int selectNumber = JItemList.getSelectedIndex();
    Item selectedItem = Items.get(selectNumber);
    if(selectedItem == null) {
        selectedItem = new Item(selectNumber, 0, 0, myId);
        Items.set(selectNumber, selectedItem);
    }
    BigInteger bigI = new BigInteger(countItem.getText());
    BigInteger maxI = new BigInteger("101"); BigInteger minI = new
    if(bigI.compareTo(maxI) != -1 || bigI.compareTo(minI) == -1) {
        printe("한번에 등록할 수 있는 최대 수량은 +-100개입니다.");
        return;
    }
    int count = bigI.intValue();
    Integer oldCount = selectedItem.getCount(myId);
    oldCount = (oldCount == null ? 0 : oldCount);

    if(oldCount + count < 0)
        printe("잘못된 값입니다.");
    else {
        selectedItem.setCount(myId, oldCount + count);
        System.out.println("선택: " + selectedItem.getName() + " |
        int cnt = 0;
        for(int i=0;i<Items.size();i++) {
            Integer cCount = (Items.get(i) != null ? (Items.get(i)
            if(cCount != null && cCount > 0)
                cnt++;
        }

        if( cnt > 7 ) {
            selectedItem.setCount(myId, 0);
            JOptionPane.showMessageDialog(null,
                    ("7종류의 음료만 판매할 수 있습니다."), "Message",
                    JOptionPane.ERROR_MESSAGE);
        } else {
            manageStock.setVisible(false);
            showMode(UI);
        }
    }
  }
});
}
```

<수정 후>

분석

기존 Item이 DVM에 없을 경우 *selectedItem*이 null 이었고, 따라서 새로 생성한 Item에 대하여 DVM에 등록해주는 작업이 누락되어 정상적으로 수행되지 않았음

→ *Items.set(selectNumber, selectedItem)* 추가

Test 결과

Pass

Brute Force Test – Read Verification Code

테스트 결과, 정상적으로 동작하여 4학년 팀에 재테스트를 요청하였고, 그 결과 PASS 확인

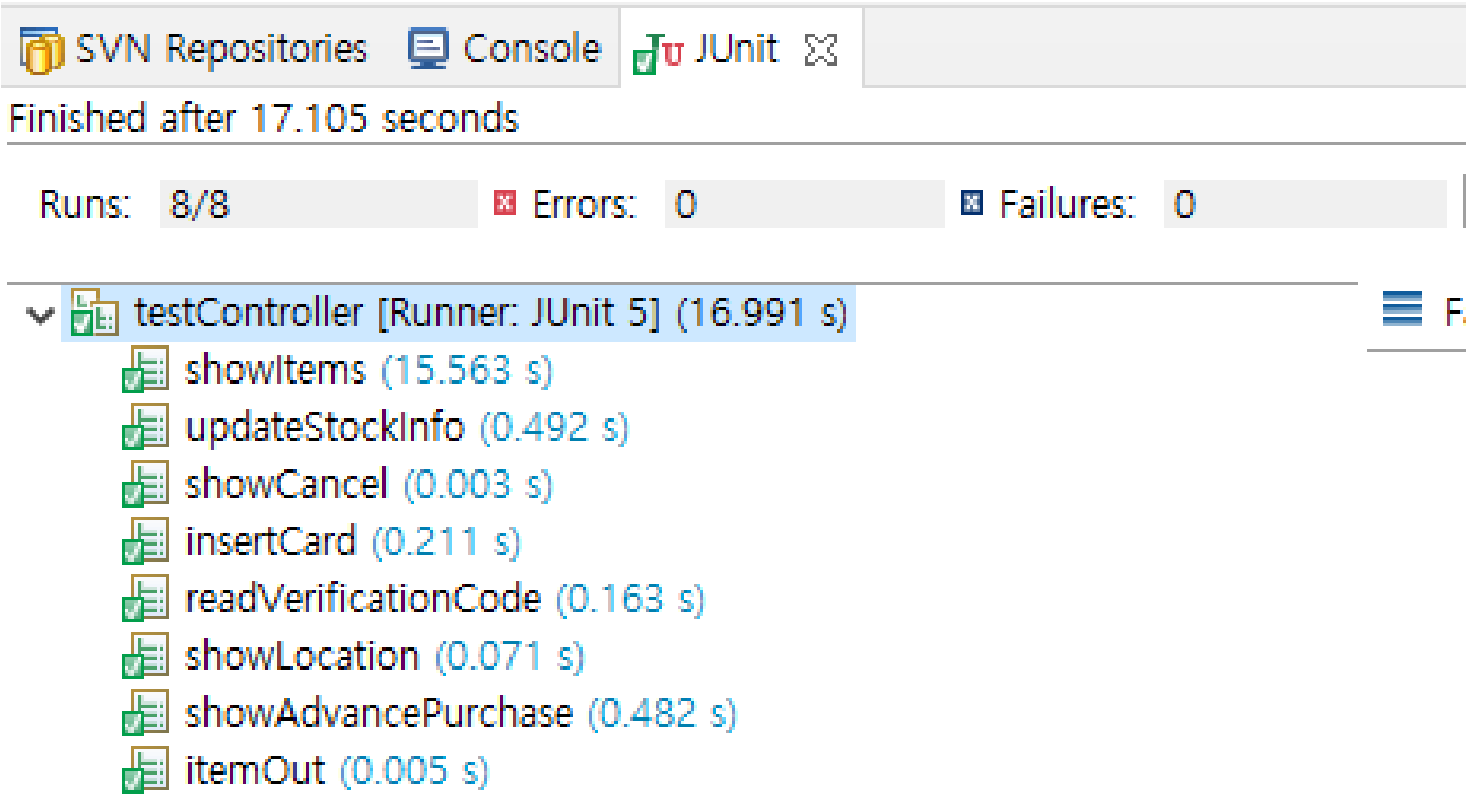
Brute Force Test – 3rd cycle

Use case	Number	Test case	Result
Manage Stock	3-1	Manager가 갱신한 재고 정보가 적용되는지 확인	Pass
Read Verification Code	14-2	기록된 코드가 유효하다면 "Item Out"을 실행하는지 확인	Pass

→ Total 49/49 (100%) Passed

Unit Test

JUnit Test – testController



JUnit Test – testController

```
        @Override
        public void actionPerformed(ActionEvent e) {
            List<Item> Items = new ArrayList<>();
            for(int i = 0; i < 20; i++) {
                Items.add(new Item(i, new Random().nextInt(3), Item.prices[i], myId));
            }
            Integer bbI = new Random().nextInt(110);
            Integer selectNumber = new Random().nextInt(20);
            Item selectedItem = Items.get(selectNumber);
            assertEquals(selectedItem.getName(), Item.names[selectNumber]);

            BigInteger bigI = new BigInteger(String.valueOf(bbI));
            BigInteger maxI = new BigInteger("100");
            if(bigI.compareTo(maxI) != -1) {
                System.err.println("한번에 등록할 수 있는 최대 수량은 100개입니다.");
                return;
            }
            int count = bigI.intValue();
            Integer oldCount = selectedItem.getCount(myId);
            oldCount = (oldCount == null) ? 0 : oldCount;
            assertNotNull(oldCount);

            if(oldCount + count < 0)
                System.err.println("잘못된 값입니다.");
            else {
                Integer targetCount = oldCount + count;
                selectedItem.setCount(myId, targetCount);
                assertThat(selectedItem.getCount(myId), is(targetCount));
                int cnt = 0;
                for(int i=0;i<Items.size();i++) {
                    Integer cCount = (Items.get(i) != null ? (Items.get(i).getCount(myId) != null ? (Items.get(i).getCount(myId)) : 0) : 0);
                    if(cCount != null && cCount > 0)
                        cnt++;
                }

                if( cnt > 7 ) {
                    selectedItem.setCount(myId, 0);
                    assertThat(selectedItem.getCount(myId), is(0));
                    System.err.println("/송류의 봄묘만 판매할 수 있습니다.");
                } else {
                    manageStock.setVisible(false);
                }
            }
        }
    });
```

JUnit Test – testController

```
254         String[] alpha = {"0", "1", "2", "3", "4"};
255         assertThat(dvms, is(alpha));
256
257         if(dvms != null) {
258             String[] locations = {"신공학관 1층", "새천년관1층", "학생회관1층", "법학관1층", "도서관1층"};
259             for(int i = 0; i < dvms.length; i++) {
260                 Integer dvmId = Integer.parseInt(dvms[i]);
261                 dvmLocations += locations[dvmId];
262
263                 if(i < dvms.length-1)
264                     dvmLocations += ",";
265             }
266         } else {
267             dvmLocations = "없음";
268         }
```

JUnit Test – testController

```
302     public void showAdvancePurchase(/*JFrame UI*/) { // (Item item)
303         JFrame UI = new JFrame();
304         JPanel showAdvancePurchase1 = new JPanel(new GridLayout(3,1));
305         JPanel itemPanel = new JPanel(new GridLayout(0,1));
306         JPanel itemLabel = new JPanel(new FlowLayout());
307         JLabel item = new JLabel("음료 :");
308         Item currentItem = new Item(0, 1, Item.prices[0], myId);
309         JLabel itemName = new JLabel(currentItem.getName());
310         assertThat(currentItem.getName(), is(Item.names[0].toString()));
311         assertThat(currentItem.getPrice(), is(Item.prices[0]));
```

JUnit Test – testController

```
353             public void actionPerformed(ActionEvent e) {
354                 showAdvancePurchase1.setVisible(false);
355                 //current Item을 갖고 있는 dvm id 구하기
356                 Integer myId = new Random().nextInt(5);
357                 HashMap<Integer, Integer> otherStocks = new HashMap<>();
358 
359                 assertNotNull(otherStocks);
360                 String dvms = null;
361 
362                 for(int td = 0; td < 5; td++) {
363                     if(otherStocks.get(td) != null) {
364                         if(otherStocks.get(td) > 0) {
365                             assertThat(otherStocks.get(td), is(Items.get(0).getCount(td)));
366                             if(dvms == null)
367                                 dvms = "";
368                             dvms += td + "^";
369                         }
370                     }
371                 }
```

JUnit Test – testController

```
444                 currentItem = Items.get(itemid_int);
445                 assertThat(currentItem.getName(), is(Item.names[itemid_int]));
446                 int stockOK = -1;
447                 if(currentItem == null) {
448
449                 } else if(currentItem.getCount(myId) == null || currentItem.getCount(myId) <= 0) {
450                     HashMap<Integer, Integer> otherStocks = currentItem.checkStockCount(myId);
451                     Set<Integer> keys = otherStocks.keySet();
452                     Iterator<Integer> iter = keys.iterator();
453
454                     while(iter.hasNext()) {
455                         Integer key = iter.next();
456                         if(otherStocks.get(key) != 0) {
457                             assertThat(Items.get(itemid_int).getCount(key), is(otherStocks.get(key)));
458                             stockOK = 1;
459                             break;
460                         }
461                     }
```

JUnit Test – testController

```
498     public void insertCard(/*JFrame UI*/) { //
499         JFrame UI = new JFrame();
500         SimpleDateFormat format2 = new SimpleDateFormat ( "yyyy년 MM월dd일 HH시mm분ss초");
501         Date time = new Date();
502         String time2 = format2.format(time);
503 
504         if(currentItem == null) {
505             System.out.println("음료가 선택되지 않았습니다.");
506         }
507         currentItem = new Item(1, 7, 1000, 1);
508         int itemPrice = currentItem.getPrice();
509         assertThat(itemPrice, is(currentItem.getPrice())); //Test Code
510         JPanel insertCard = new JPanel();
511         insertCard.setLayout(new GridLayout(4,0));
```

JUnit Test – testController

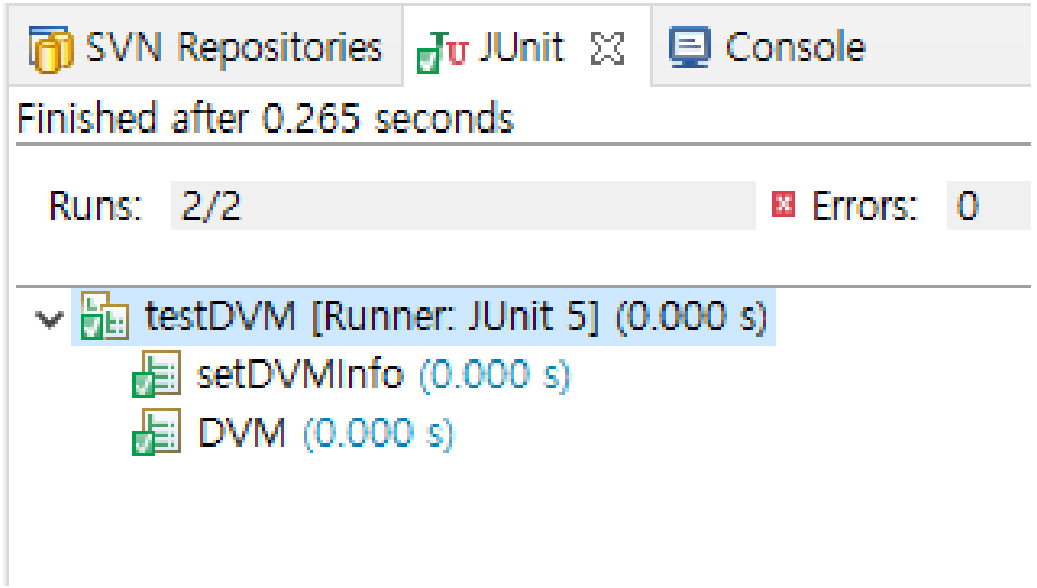
```
if(purchase.getPaymentResult()) {
    insertCard.setVisible(false);
    if(purchase.IsAdvanced()) {
        //current Item을 갖고 있는 dvm id 구하기
        assertTrue(myCount <= 0);
        HashMap<Integer, Integer> otherStocks = new HashMap<>();
        String dvms = null;

        for(int td = 0; td < 5; td++) {
            if(otherStocks.get(td) != null) {
                if(otherStocks.get(td) > 0) {
                    if(dvms == null)
                        dvms = "";
                    dvms += td + "^";
                }
            }
        }
        int newCode = VerificationCode.getInstance(myId).createCode();
        assertTrue(newCode > -1);
    }
    else {
        int targetId = currentItem.getId();
        Item item = Items.get(targetId);
        int targetCount = item.getCount(myId)-1;
        item.setCount(myId, targetCount);
        assertThat(item.getCount(myId), is(targetCount));
        itemOut();
    }
} else {
```

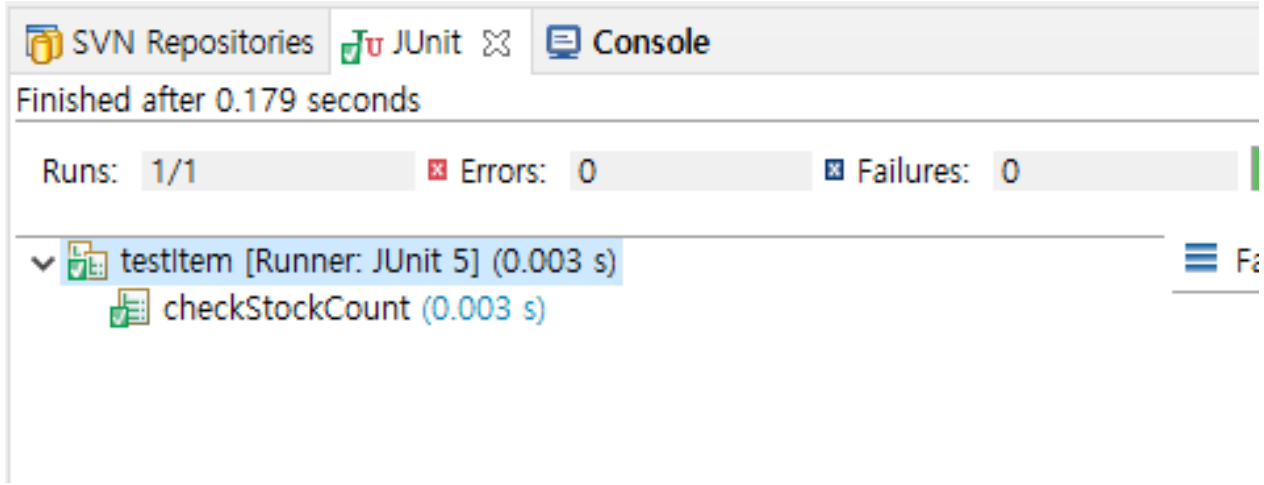
JUnit Test – testController

```
696                 if(codeItem > -1) {
697                     readVcode.setVisible(false);
698                     currentItem = Items.get(codeItem);
699                     assertNotNull(currentItem);
700                     VerificationCode.getInstance(myId).resetCode(inputCode, true);
701                     assertThat(VerificationCode.getInstance(myId).checkCode( inputCode ), is(-1));
702                     itemOut();
703                 } else {
704                     System.err.println("잘못된 코드입니다.");
```

JUnit Test – testDVM

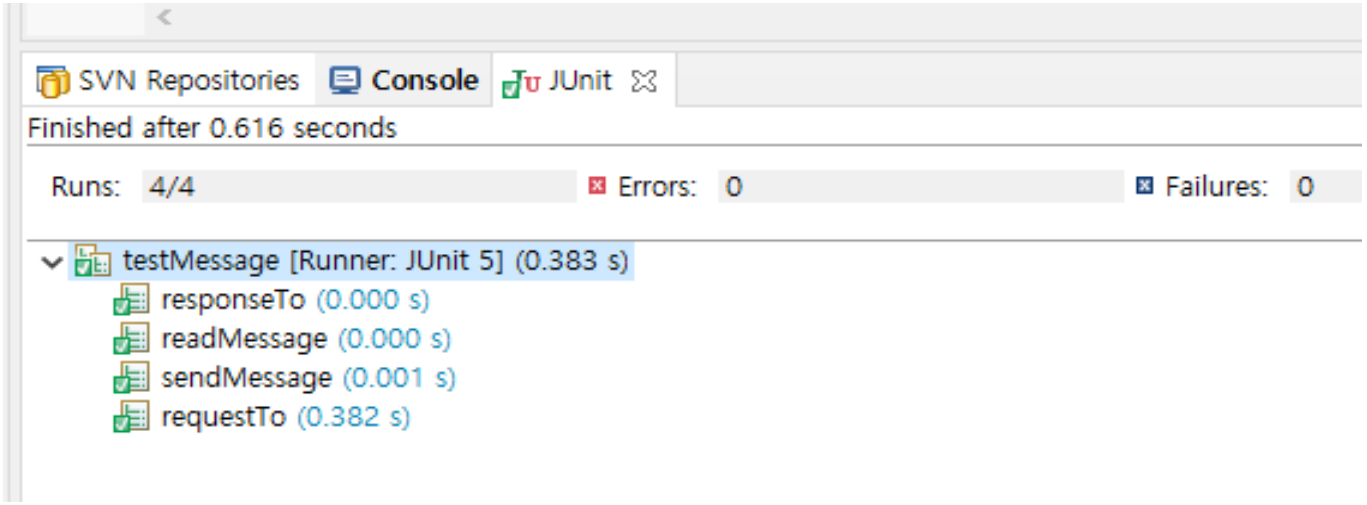


JUnit Test – testItem



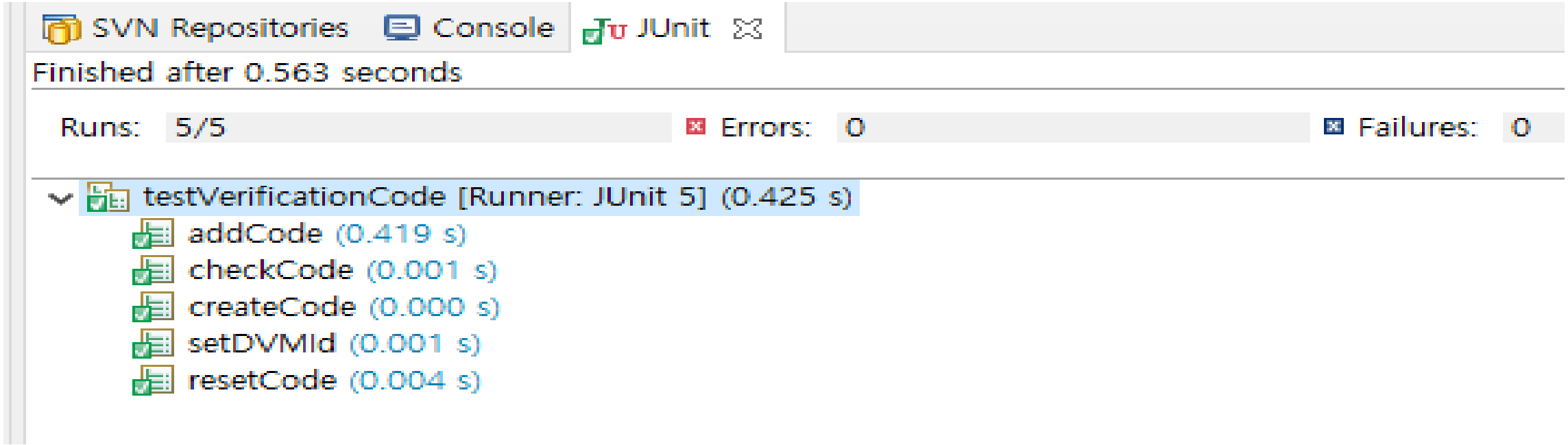

```
109         int dvm_Id = dvm_Set[0];
110         int dst_Id = dvm_Set[1];
111
112         assertNotSame(dvm_Id, dst_Id);
113
114         if(getCount(dvm_Id) == null || getCount(dvm_Id) <= 0) {
115
116
117             Message msg = new Message();
118             //broadcast
119             tmap = msg.sendMessage(1, "" + this.id, dvm_Id, dst_Id);
```

JUnit Test – testMessage




```
112             assertSame(src_Id, dst_Id);
113             //
114             response = new HashMap<Integer, String>();
115             for(int i = 0; i < 5; i++) {
116                 if(i != src_id) {
117                     targetDVM = Controller.getInstance(i);
118
119                     assertNotNull(targetDVM);
120                     Map<Integer, String> retFromDVM = new HashMap<>();
121                     if(retFromDVM.get(2) != null) {
122                         int tCount = Integer.parseInt(retFromDVM.get(2)
123                         assertNotNull(tCount);
```

JUnit Test – testVerificationCode



JUnit Test – testVerificationCode

```
public void addCode() {
    // TODO implement here
    Random random=new Random();
    Integer itemid=random.nextInt(20);
    Integer code=random.nextInt(999999);
    boolean flag=true;
    try {
        ArrayList<Integer> oldList = (codeList.get(itemid) == null ? new ArrayList<Integer>() : codeList.get(itemid));
        Item item = Controller.getInstance(dvmId).getItems(dvmId).get(itemid);
        if(item != null) {
            Integer count = item.getCount(dvmId);
            if(count != null && count > 0) {
                assertTrue(count > 0);
                oldList.add(code);
                codeList.put(itemid, oldList);
                item.setCount(dvmId, count-1);
            }
        }
    } catch (Exception e) {
        System.err.println("Error occured while adding code");
        flag=false;
    }
}
```

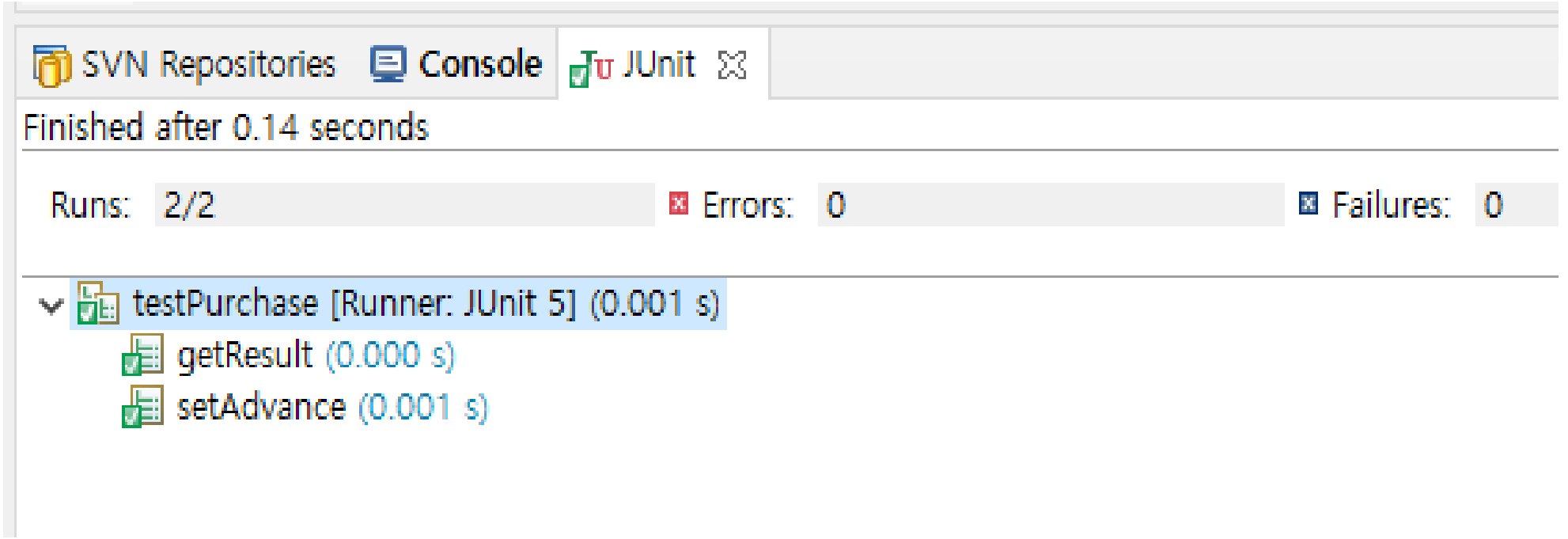
JUnit Test – testVerificationCode

```
public void resetCode() {
    Integer code=123456;
    Boolean broadcast=true;
    int itemid = -1;
    boolean isDone = false;
    try {
        Set<Integer> itemIds = codeList.keySet();
        Iterator<Integer> iter = itemIds.iterator();

        while(iter.hasNext()) {
            Integer key = iter.next();
            ArrayList<Integer> itemCodes = codeList.get(key);
            itemCodes.add(123456);
            for(int i = 0; i < itemCodes.size(); i++) {
                assertEquals(itemCodes.get(i),code);
                if(itemCodes.get(i).equals(code)) {
                    itemid = key;
                    itemCodes.remove(i);
                    codeList.put(itemid, itemCodes);
                    isDone = true;
                    break;
                }
            }
            if(isDone)
                break;
        }
    }
    catch(NullPointerException e) {
        System.err.println("There is no code: " + code);
        //return false;
    }
        /* broadcast */

    if(isDone) {
        if(broadcast) {
                Message net = new Message();
                //net.sendMessage(4, (itemid +"^"+ code), dvmId, dvmId);
            } else {
                Item item =Controller.getInstance(dvmId).getItems(dvmId).get(itemid);
                if(item != null) {
                    assertNotNull(item);
                    Integer count = item.getCount(dvmId);
                    count = (count == null ? 0 : count);
                    assertTrue(count >= 0);
                    item.setCount(dvmId, --count);
                }
            }
        }
    //return isDone;
}
```

JUnit Test – testPurchase



Static Code Analysis

Static Code Analysis

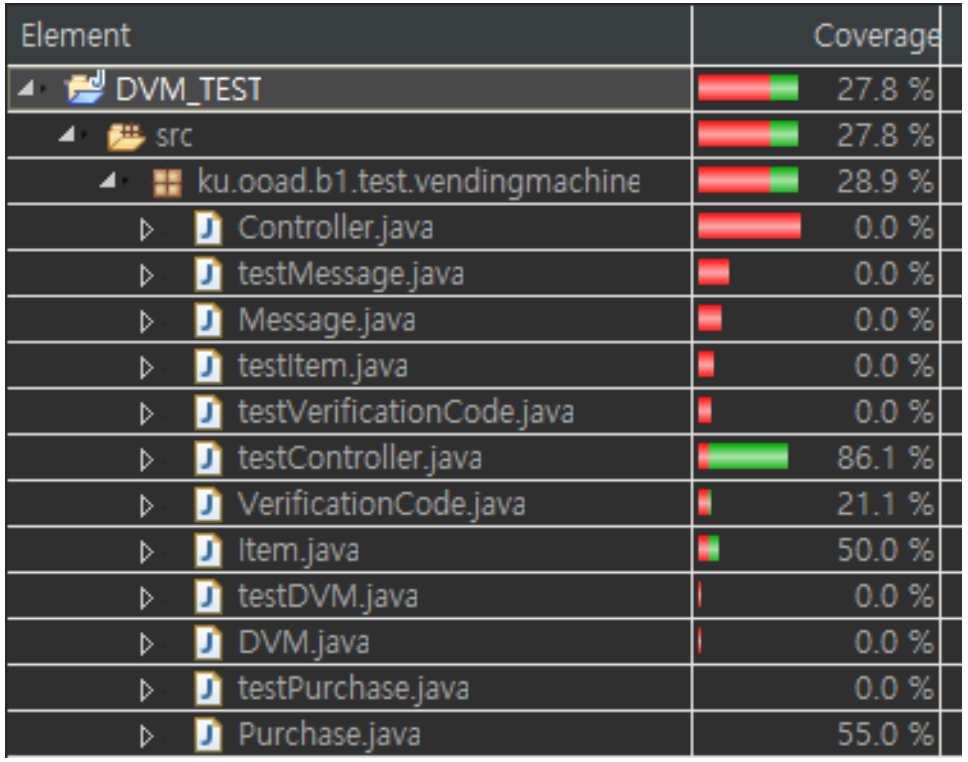
4학년 팀의 요청

코드커버리지가 40퍼를 넘지 못하기때문에 최소한 50퍼는 넘게 수정부탁드립니다.

→ **테스트에 필요 없는 코드 삭제**

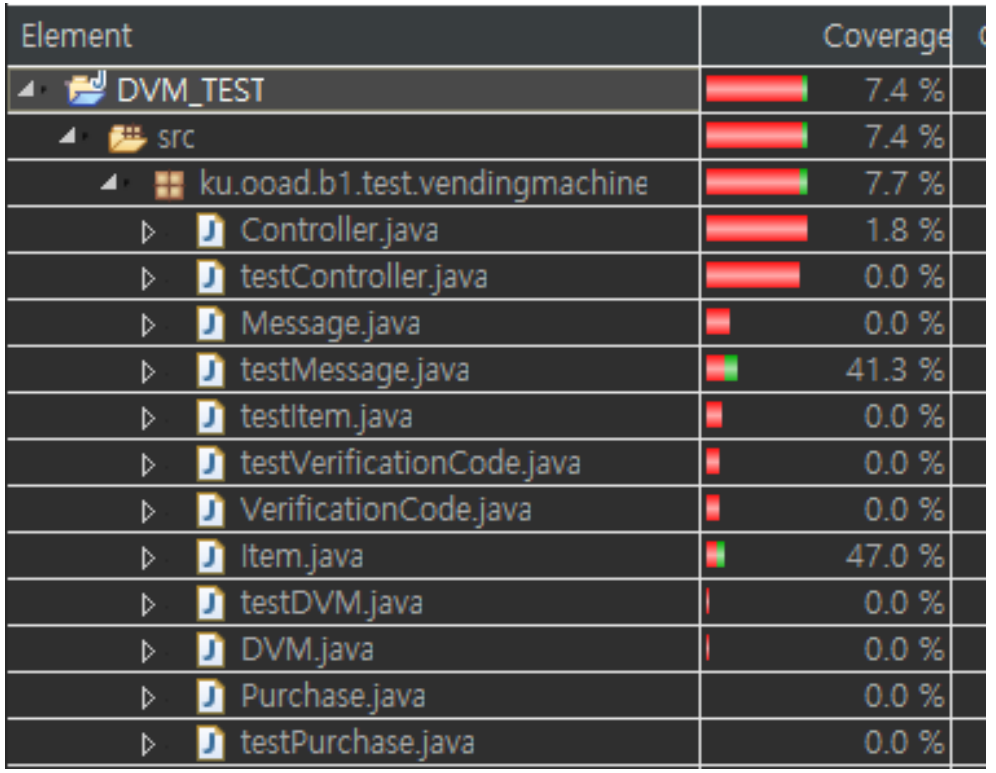
Static Code Analysis

testController



Element	Coverage
DVM_TEST	27.8 %
src	27.8 %
ku.ooad.b1.test.vendingmachine	28.9 %
Controller.java	0.0 %
testMessage.java	0.0 %
Message.java	0.0 %
testItem.java	0.0 %
testVerificationCode.java	0.0 %
testController.java	86.1 %
VerificationCode.java	21.1 %
Item.java	50.0 %
testDVM.java	0.0 %
DVM.java	0.0 %
testPurchase.java	0.0 %
Purchase.java	55.0 %

testMessage

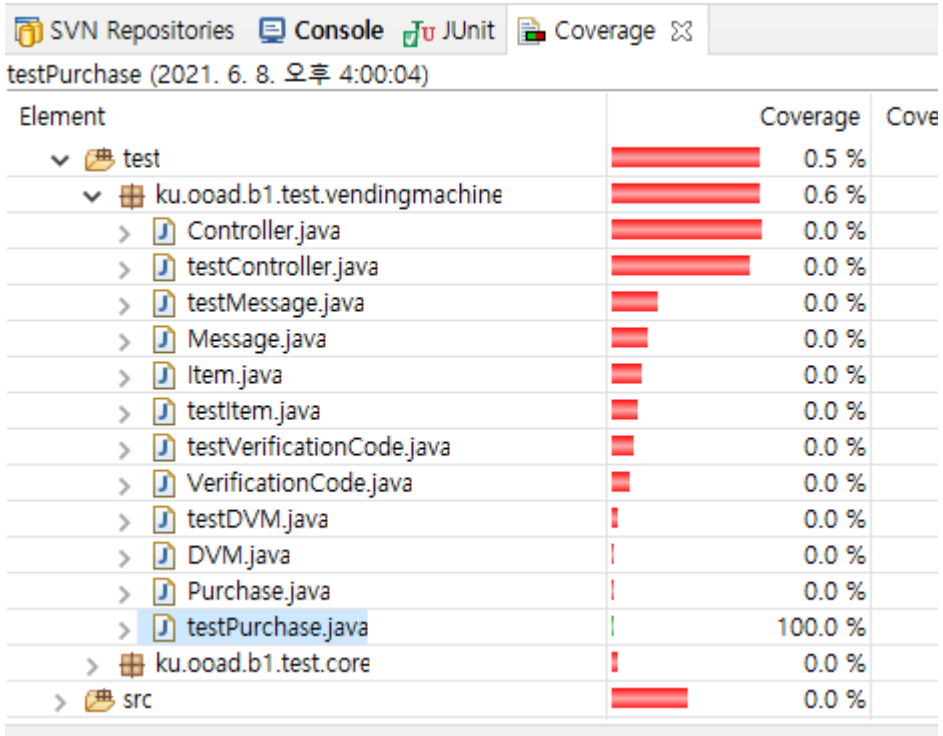


Element	Coverage
DVM_TEST	7.4 %
src	7.4 %
ku.ooad.b1.test.vendingmachine	7.7 %
Controller.java	1.8 %
testController.java	0.0 %
Message.java	0.0 %
testMessage.java	41.3 %
testItem.java	0.0 %
testVerificationCode.java	0.0 %
VerificationCode.java	0.0 %
Item.java	47.0 %
testDVM.java	0.0 %
DVM.java	0.0 %
Purchase.java	0.0 %
testPurchase.java	0.0 %

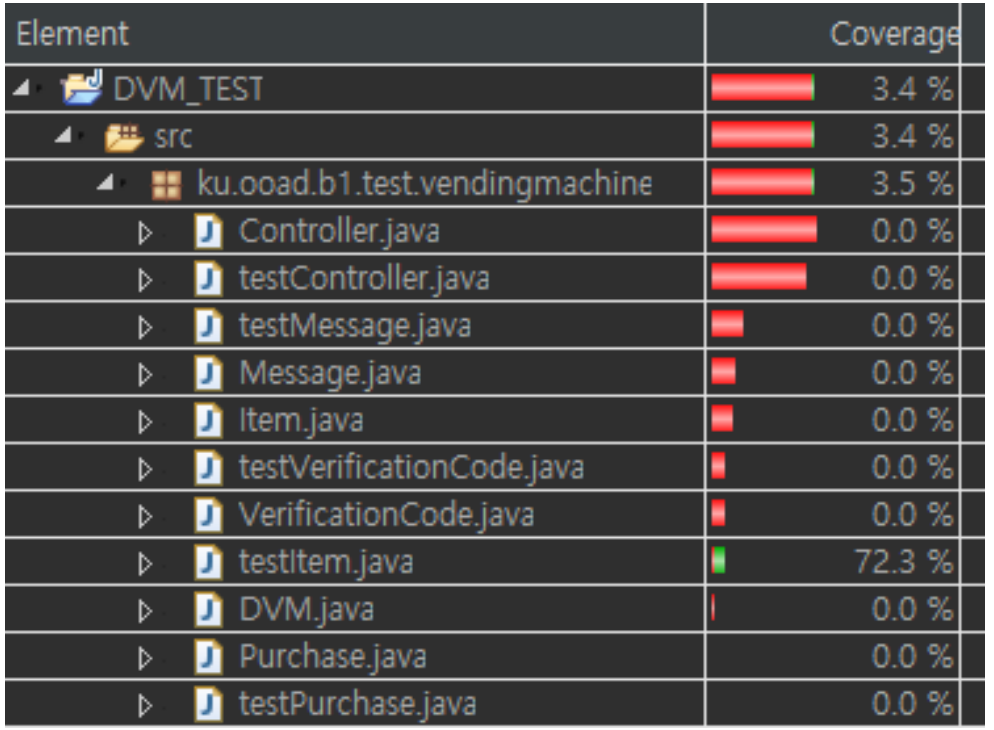
→ TestMessage의 경우, 함수화를 하더라도
해당 함수의 호출은 결국 조건문에 의해 분리되고,
함수화를 함으로써 얻는 coverage 이득이 낮다고 판단하여
수정하지 않음.

Static Code Analysis

testPurchase

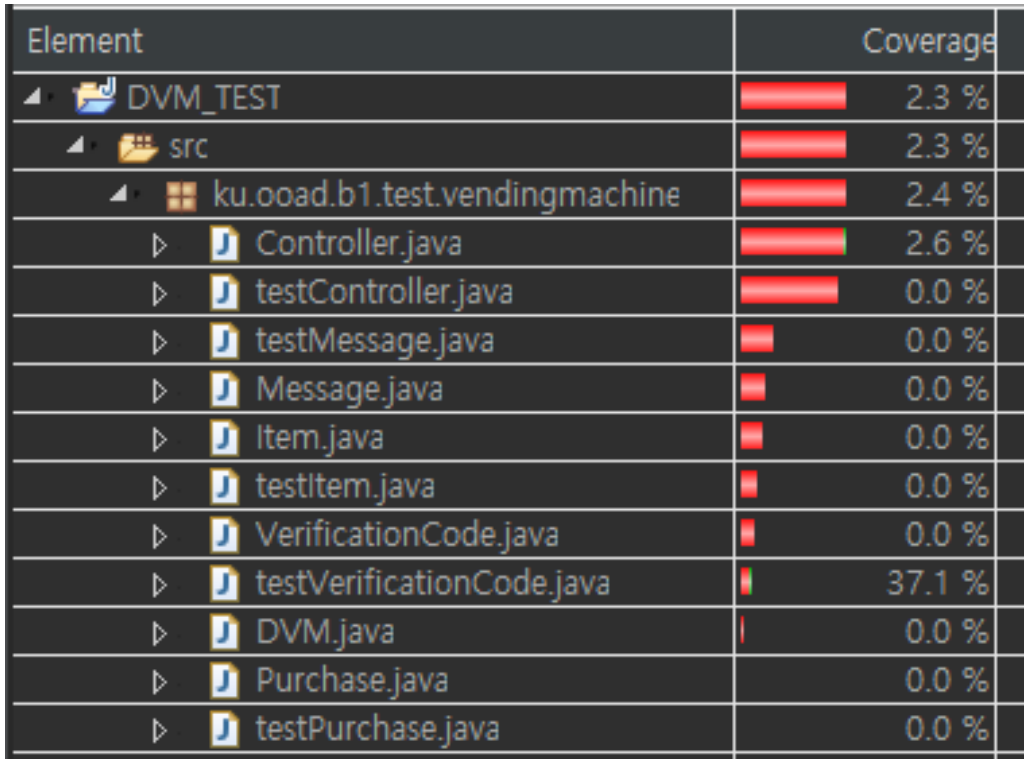


testItem



Static Code Analysis

testVerification



Element	Coverage
DVM_TEST	2.3 %
src	2.3 %
ku.ooad.b1.test.vendingmachine	2.4 %
Controller.java	2.6 %
testController.java	0.0 %
testMessage.java	0.0 %
Message.java	0.0 %
Item.java	0.0 %
testItem.java	0.0 %
VerificationCode.java	0.0 %
testVerificationCode.java	37.1 %
DVM.java	0.0 %
Purchase.java	0.0 %
testPurchase.java	0.0 %

→ testVerification의 경우, 키가 존재하는지 찾는 부분에서 HashMap 내부에 key가 존재하는지 while문으로 탐색하는데 해당 부분에서 조건 분기에 의해 coverage가 낮은 것을 확인하였고, map을 임의로 설정한뒤 테스트하는건 무의미하다 판단하여 따로 수정하지 않음

끝으로...

끝으로... – CTIP 소감

이름	소감
정연수	CTIP 환경의 경우 온라인 상에서 팀원들과 원활히 소통하고 협업할 수 있었고 분업이 보다 수월하게 이루어졌다. 평소 git을 사용하고 svn은 처음 사용해봤지만 git보다 직관적으로 사용할 수 있었다. commit과 update만 사용한다면 svn이 사용하기도 편하고 처음 배우는 사람도 쉽게 사용할 수 있다고 생각한다. Trello 사용으로 이슈 트래킹과 피드백이 직관적으로 쉽게 이루어졌으며 젠킨스를 활용한 테스팅도 인상깊었다. 개인이 머리속에 기억하거나 따로 메모해서 저장하지 않더라도 확인 및 피드백하기 좋은 서비스라고 생각한다. 슬랙 또한 모든 내용을 체크할 수 있었지만 소통같은 경우는 확실히 대면으로 의사소통하거나 회의롤 통한 것이 가장 효과적이라는 생각이 들었다.
김민환	CTIP에서 Trello를 통해 Issue에 대해 보기 쉽게 관리할 수 있는게 좋았다. 특히 이슈마다 현황과 내용을 한눈에 볼 수 있고, 주제마다 대화를 나눌 수 있는게 좋았다. 단점으로, 익숙하지 않아서 그런지 사용성이 떨어지는 것 같다.
이승헌	장점 : 프로젝트를 버전 별로 관리할 수 있어서 여러사람이 함께 개발하는데 이점을 주었다.또 한 테스트를 통해서 나온 이슈들을 한 눈에 파악 할 수 있고 어떻게 이슈들을 고치고 해결했는지 쉽게 알 수 있어서 구현을 하는데 훨씬 수월했다 / 단점 : 처음 사용해보는 툴들이 많아서 어떻게 활용해야 되는지 잘 몰라서 기능을 제대로 다 못 쓴거 같다. 툴의 사용목적과 메뉴얼이 있었으면 좋겠다.
조벽정	개인적으로 어렵다.다른 팀원들의 도움하에.이 일을 하고도 많은 경험과 지식을 얻었습니다.온라인강의는 실천과 조작이 아주 불편하고 또한 나에게 많은 어려움과 문제를 주었다.기타 학우들의 도움으로 전공 지식에 대해 부동한 인식을 가지기 시작했다.
황유란	4학년 테스트팀이 제공한 CTIP을 통해 우리가 찾기 못했던 오류나 모순된 부분들을 많이 찾았으며 관련 이슈들을 한 번에 볼 수 있어 개발에 많이 도움이 되었다. 이번 학기는 비대면이어서 팀원들과 만날 수 있는 시간이 부족했다. 다음에 대면수업을 하게 된다면 팀원들과 만나서 얘기할 수 있는 시간이 늘어나 의사소통이 훨씬 효과적일 것이라고 생각한다

끝으로... – UP 소감

이름	소감
정연수	UP 의 경우 단계별로 시간을 많이 투자했지만 그 만큼 협업과 개발을 했을 시 시간이 단축되었다고 생각한다. 협업하기에 매우 좋은 방법론인 것 같고 UP의 완성도에 따라 실제 개발을 담당하는 사람들의 시간이 좌우된다는 것을 확인했다. UP를 완벽하게 한다면 개발하는 과정에서 시간 단축 및 기획자와의 소통이 최소화되겠지만 그게 아니라면 지속적인 소통 또는 디자인을 변경해야되는 것을 깨달았다. 처음 analysis과 elaboragtion부터 시스템 내부의 operation 등 완벽하게 하려고 했었지만 디자인 단계로 가며 왜 블랙박스로 생각하고 진행하는지 생각해 보는 계기가 되었다. 학습의 목적 상 정해진 시간에 완료하려고 하니 여러번 iteration을 돌리지 못했지만 본 목적처럼 여러번 iteration이 돌아간다면 UP가 더욱 효과적으로 진행될 것이라고 생각한다.
김민환	컴퓨팅적 사고를 통해 Waterfall 방식의 개발 방법론을 배웠었다. 당시에 프로젝트에 대한 설계를 끝내고 개발하면 편하다는 방식이 충격이었는데, UP의 여러 번에 걸쳐 개발 한다는 내용은 더 인상깊었다. 특히 전체 프로세스를 위해 iteration을 나누고, 각 iteration마다 주제를 정하고 개발하는 점이 새로웠다. 이번 학기에서는 1st cycle부터 DVM 전체내용을 완성해야 해서 그런지 모르겠지만 UP의 이론적인 내용을 직접 경험하기에는 좀 부족했던 것 같다.
이승헌	장점 : 설계 부터 꼼꼼하게 계획하기 때문에 구현하고 테스트 하는데 큰 힘이 들지 않았다. 그리고 Use Case를 통해 시나리오를 예상 해 봄으로써 System이 어떻게 동작하는지에 대한 청사진이 머리속에 있어서 좋았다. / 단점 : 시간이 많이 들고 객체들간의 독립성을 확보하면서 설계하는것이 꽤 까다롭고 개발하는 중간중간 전체 문서를 수정해야 되는 경우가 많아서 문서 관리하는 것이 어려웠다.
조벽정	구현과 테스트의 시작은 아주 순조로웠다.초기 단계 계획의 중요성도 느꼈다.사전 설계 토론은 반드시 상세해야 한다.뒷부분에 개발 부분, 테스트 부분이 깔려있습니다.
황유란	UP기반으로 기획부터 테스트까지 단계별로 프로젝트를 진행하면서 초반 기획,디자인 파트에서 문서화의 중요성을 느꼈다. 간단해 보였지만 하나하나 고려하다보니 생각보다 시간이 오래걸렸고, 이 때 잘 해놓으면 이후 개발, 테스트 파트에서도 훨씬 수월해진다는 것을 깨달았다.

Thank you