Distributed Vending Machine

OOPT : OOI (The 2nd Cycle)

TEAM 1

정연수 김민환 이승헌 조벽정 황유란

INDEX

- Spec Review
- Interaction Diagram
- Test Case
- Unit Testing
- Class Diagram

Non Functional Requirements

👉 "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.

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

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

Performance Requirement

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

Performance Requirements

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

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.

Record Term In Glossary

💡 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.

Verification Code	선구매 인증번호
Count	item의 수량
Mode	사용자에게 제공하는 구매 유형.

Define Business 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.

Ref	Function	Category
1.1	Set Up All	Hidden

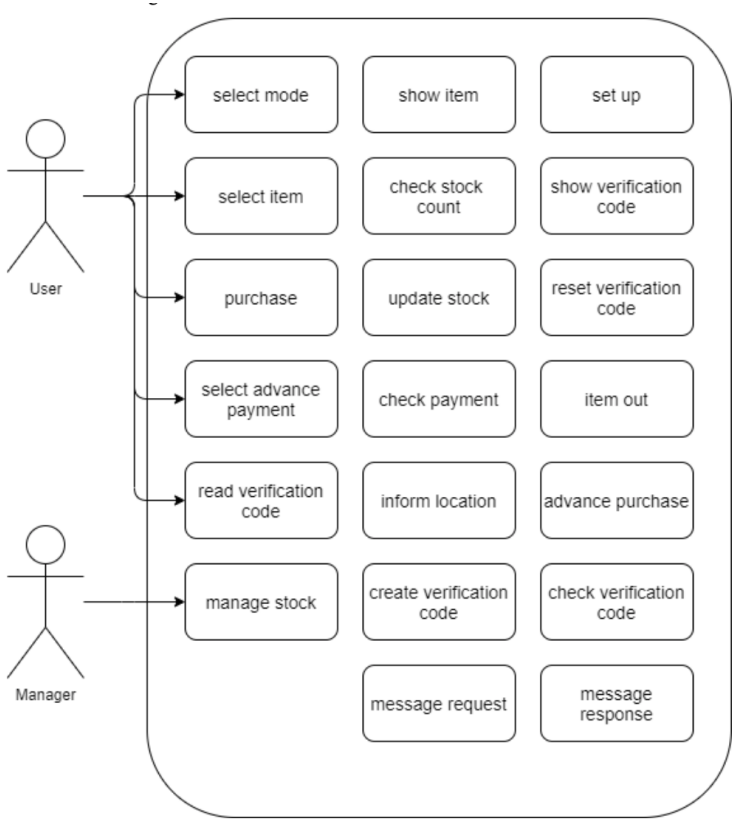
Identify use cases

- 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

Use Case Diagram

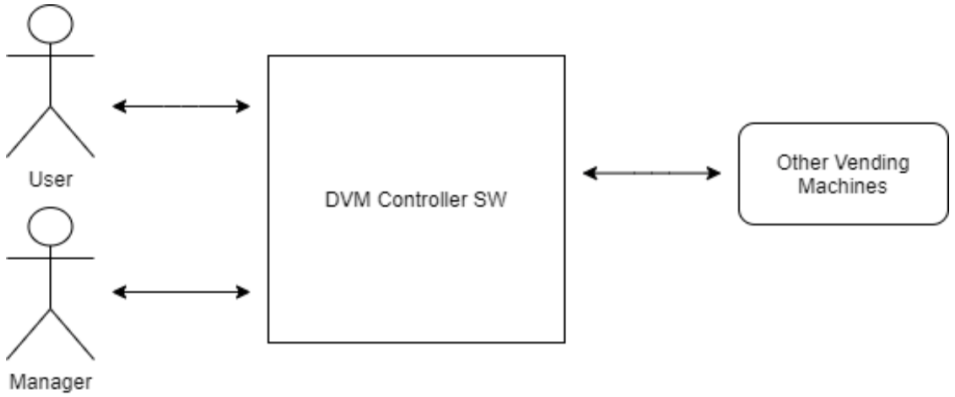
👉 'set up'은 hidden use case로 actor가 없어야 하는데 diagram 상에서는 있는 것으로 그려져 있습니다.

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



Use Case Diagram

👉 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.



Describe Use Cases

👉 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.

Use Case	7. Update Stock
Actor	None
Description	- 재고의 증감을 하기 위한 use case. - 사용자가 자판기에 음료를 구매하면 해당 음료의 재고를 감소시키거나 관리자가 음료의 재고를 보충하면 해당 음료 재고의 수를 증가시킨다.

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

1	Set up all	1. 전원 공급: 네트워크 및 재고 초기화	1. 전기가 방전: 시스템이 강제 종료됨

Describe Essential Use Cases

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

💡 "R4.1"이 중복되었으므로 삭제 부탁드립니다.

💡 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.

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

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 Use Case: “Message Request”, “Update Stock”, “Select Mode”
Pre-Requisites	N/A
Typical Courses of Events	(A1): User, (A2): Manager, (A3): Other DVM, (S): System 1. (S): 다른 DVM의 ID 를 받아온다. 2. (S): 다른 DVM의 재고 정보를 받아온다. 3. (S): 모든 자판기의 재고 정보를 업데이트 한다. 4. (S): “Select Mode”를 실행한다.
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

Describe Essential Use Cases

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.

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

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.**

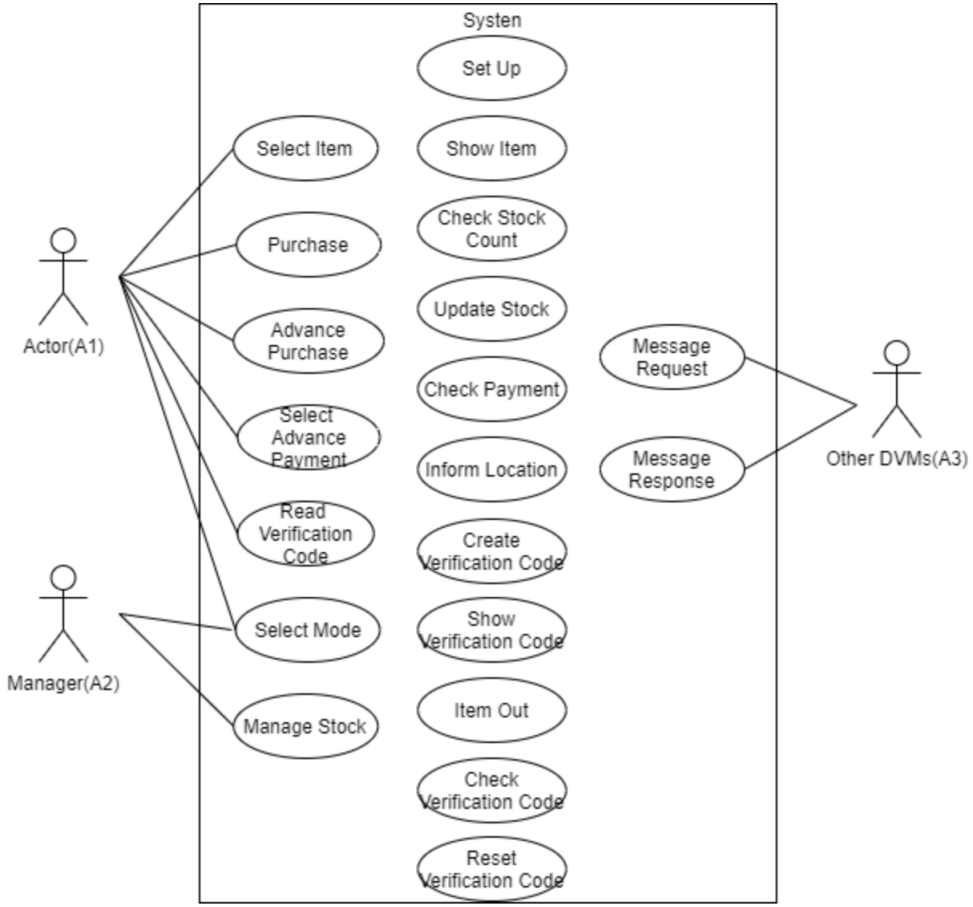
해당 내용들은 또 다른 성공 시나리오 이므로 Alternative Courses of Events로 옮기는 것이 좋습니다.

Use Case	11. Select Advance Payment
Actor	User
Purpose	(business use case에 포함)
Overview	(business use case에 포함)
Type	Evident
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.

Refine Use Cases Diagram

(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.

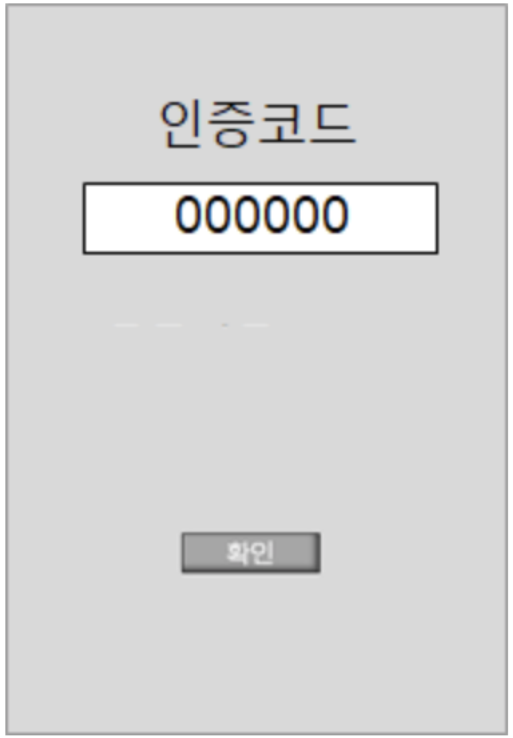


* (A1): User, (A2): Manager, (A3): Other DVMs

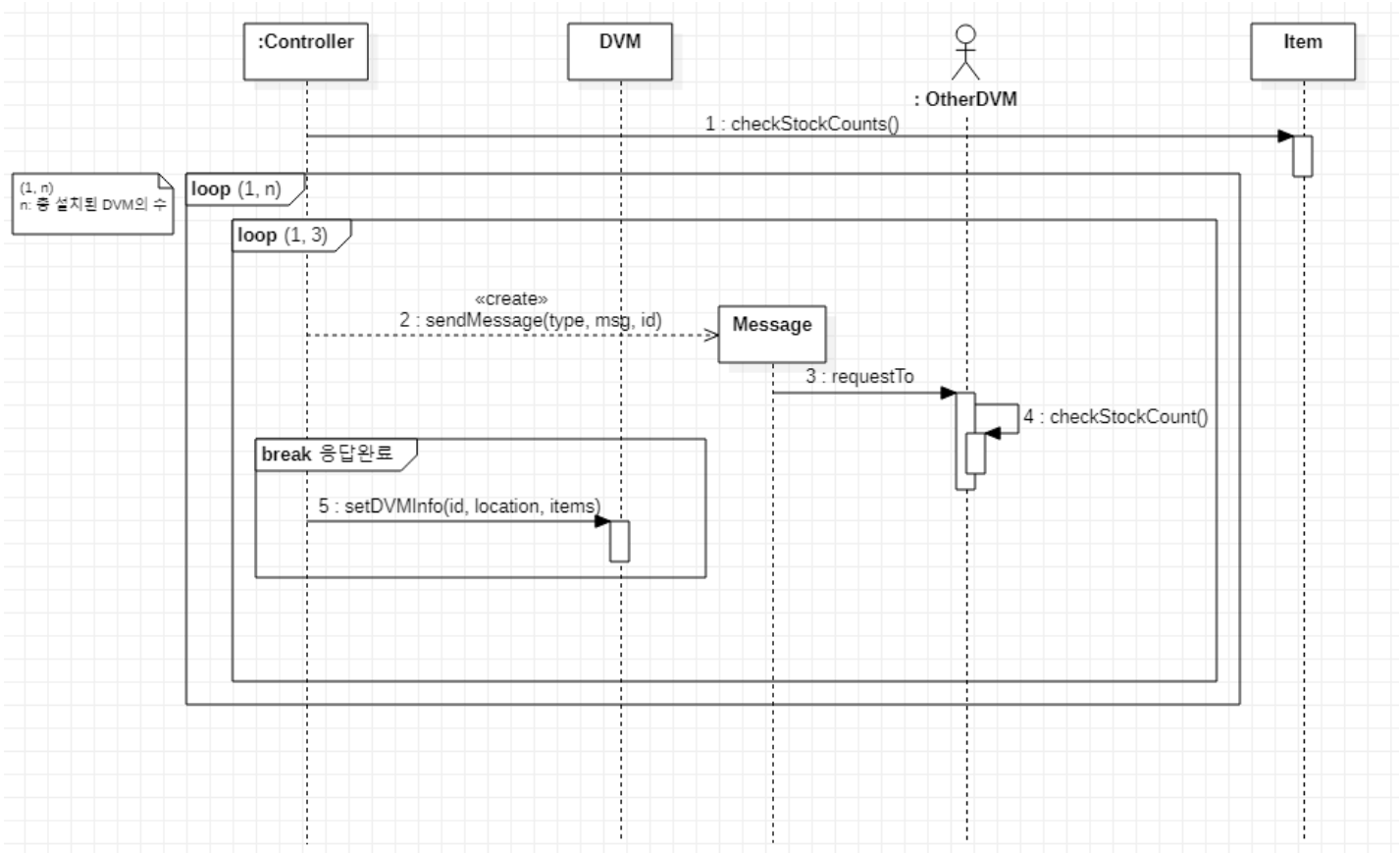
Define UI

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

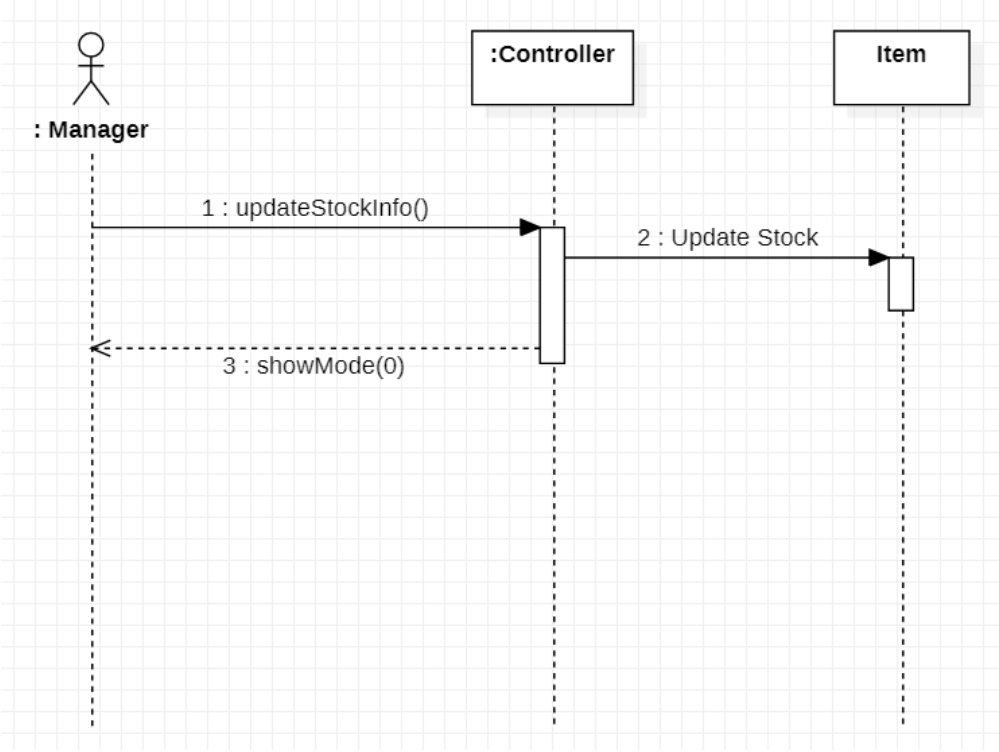
👉 Q. 남은 시간에 관한 정의는 이전 문서에는 없었는데 UI에서는 표현이 되어 있었으나 추후 문서에서는 수정되었습니다.



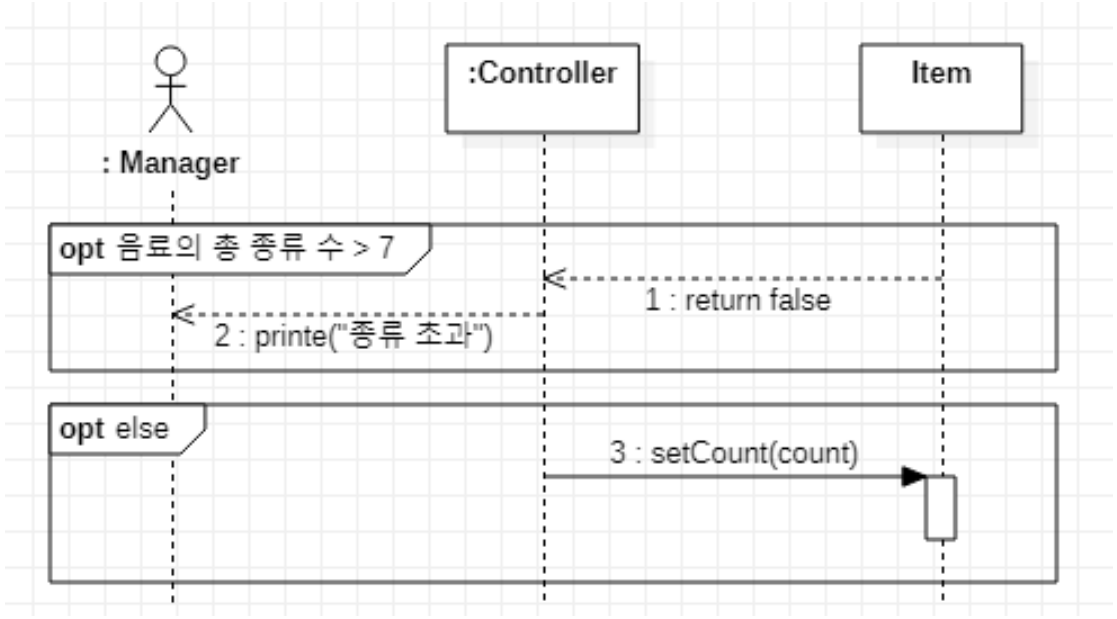
Interaction Diagram - Set up



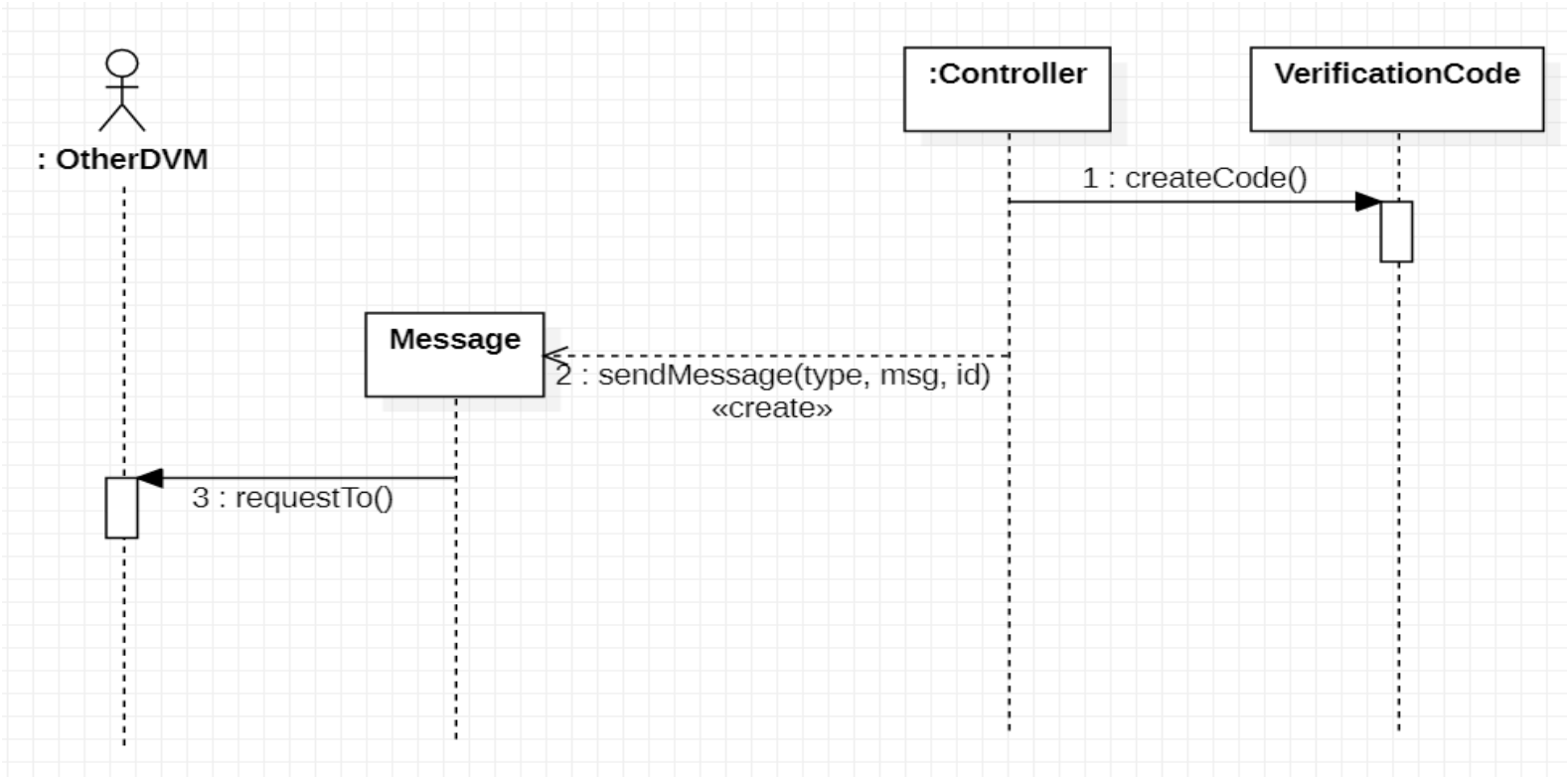
Interaction Diagram – Manage Stock



Interaction Diagram – Update Stock



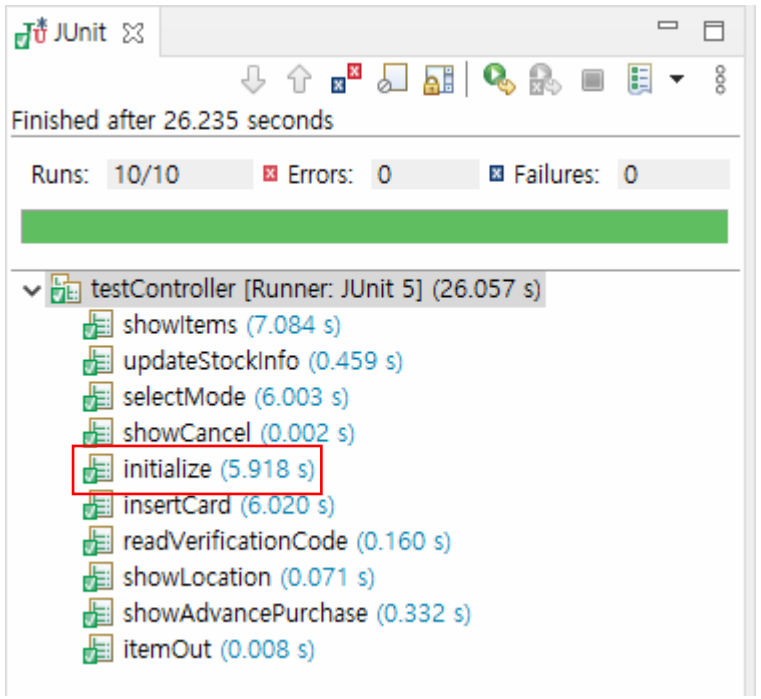
Interaction Diagram – Create Verification Code



Test Case

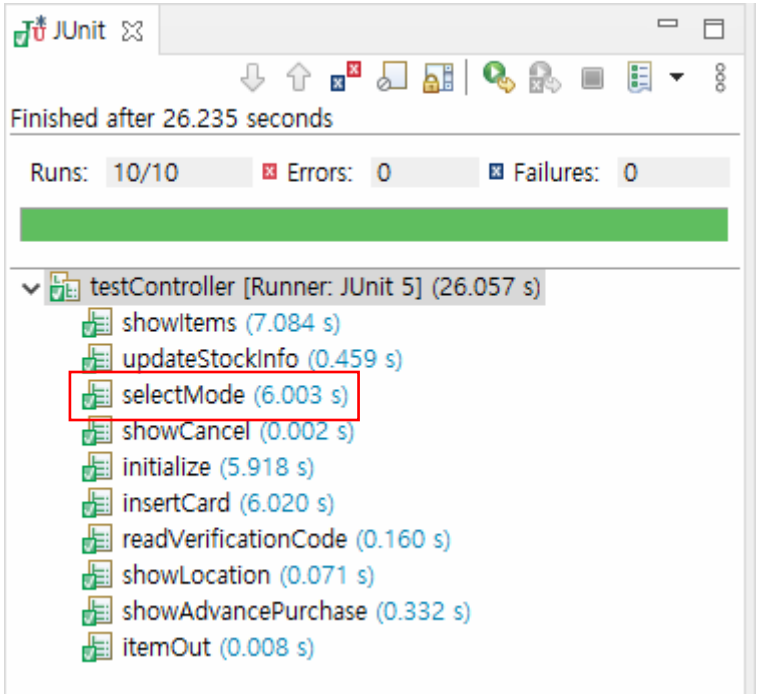
Use case	Number	Test case	Result	Modify
Select Mode	2-2	User가 여러 개의 모드를 선택했을 때 현장구매 모드가 실행되는지 확인	Fail	Removed
Show Item	4-2	User가 선택한 음료가 선택되는지 확인	Fail	Modified
Update Stock	7-2	음료 재고 정보를 주어진 값에 맞게 최신화 하는지 확인	Fail	Modified
Advance Purchase	9-2	선구매 결제에 성공했을 때 받아온 선구매 코드를 "Message Request"를 통해 broadcast 하는지 확인	Fail	Modified
	9-6	선구매 결제에 실패했을 때, 감소시켰던 재고 수량을 다시 증가시키는지 확인	Fail	Modified
	9-8	선구매 코드 입력 결제 시, 해당 item의 재고가 줄어드는지 확인	Fail	Modified
Select Advance payment	11-2	선구매 결정한 음료의 재고가 남아있다면 "Message Request"로 재고를 감소시키는지 확인	Fail	Modified
	11-3	재고를 감소시킨 후 "Advance Purchase"를 실행하는지 확인	Fail	Modified
	11-8	모드 선택과 취소를 둘다 누를 경우 사용자로부터 재입력 받는지 확인	Fail	Removed
	11-9	선구매 진행을 위한 결제 시 다른 자판기들의 재고가 바로 줄어드는지 확인	Fail	Modified
	11-10	선결제 후 선구매 코드 입력 시 해당 item의 재고가 줄어드는지 확인	Fail	Modified
Create Verification Code	13-3	선결제 코드가 여러 종류의 음료에 중복되어 생성되지 않는지 확인	Fail	Modified
Item Out	15-1	음료가 선택되어 있고, 결제가 완료되었거나 유효한 선구매 코드를 입력했을 때만 음료를 제공하는지 확인	Fail	Modified
	15-2	현재 자판기에 재고가 없을 때는, 선구매 코드를 입력해도 Item Out이 되지 않아야 함	Fail	Modified
Message Request	19-1	보내고자하는 메시지 종류를 수신 대상에게 잘 보내는지 확인	Fail	Modified

Unit Test – Controller(initialize)



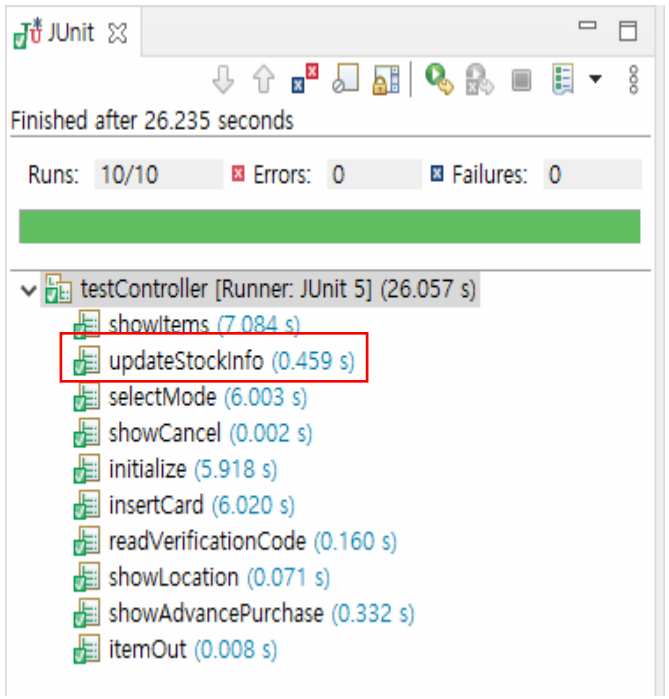

```
79    @Test
80    public void initialize() {
81        Integer myId = 0;
82        List<DVM> dvms = new ArrayList<>();
83        List<Item> items = new ArrayList<>();
84
85        String[] locations = {"신공학관 1층", "새천년관1층", "학생회관1층", "법학관1층", "도서관1층"};
86        for(int i = 0; i < 5; i++) {
87            dvms.add(null);
88        }
89        assertNotNull(dvms);
90        assertNotNull(items);
91        this.dvms = dvms;
92        this.Items = items;
93
94        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
95
96        JFrame UI = new JFrame();
97        UI.setTitle("DVM " + locations[myId] + "(id: " + myId + ")");
98        UI.setSize(370,450);
99        UI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
100
101       selectMode();
102   }
```

Unit Test – Controller(selectMode)



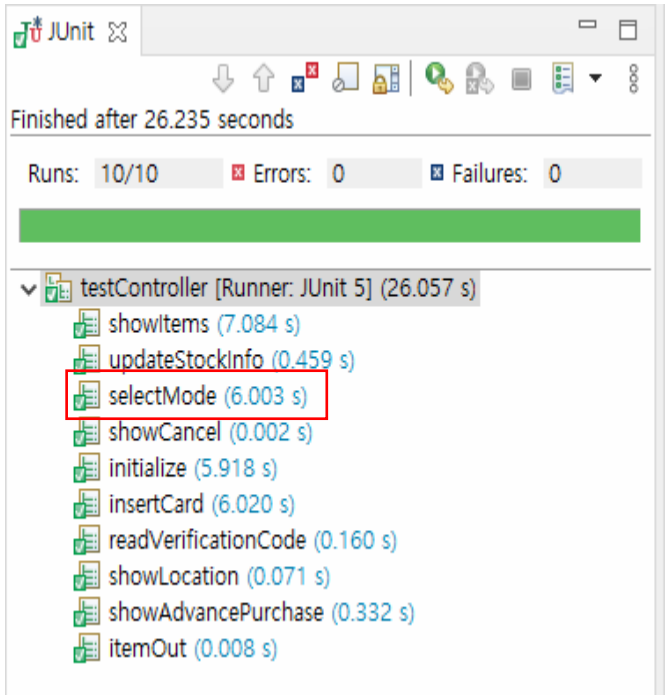

```
114    @Test
115    public void selectMode() {
116        JFrame UI = new JFrame();
117        JPanel Panel = new JPanel();
118        int mode = new Random().nextInt(4);
119
120        if(Panel != null) {
121            Panel.setVisible(false);
122            System.out.println(mode + " 클릭");
123        }
124        assertNotNull(mode);
125
126        if(mode == 0) {
127            showMode();
128        }
129        else if(mode == 1) {
130            showItems();
131        }
132        else if(mode == 2) {
133            readVerificationCode();
134        }
135        else if(mode == 3) {
136            updateStockInfo();
137        }
138    }
```

Unit Test – Controller(updateStockInfo)



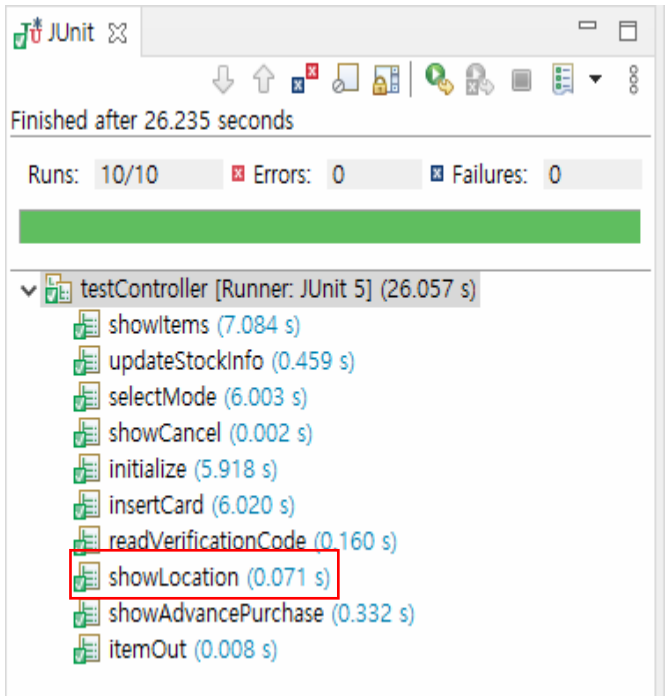

```
251        ok.addActionListener( new ActionListener(){
252            @Override
253            public void actionPerformed(ActionEvent e) {
254                // TODO Auto-generated method stub
255                List<Item> Items = new ArrayList<>();
256                for(int i = 0; i < 20; i++) {
257                    Items.add(new Item(i, 1, Item.prices[i], myId));
258                }
259                Integer myId = new Random().nextInt(5);
260                Integer target = new Random().nextInt(20);
261                Item selectedItem = Items.get(target);
262                assertEquals(selectedItem.getName(), Item.names[target]);
263
264                int count = Integer.parseInt(countItem.getText());
265                Integer oldCount = selectedItem.getCount(myId);
266                oldCount = (oldCount == null ? 0 : oldCount);
267
268                if(oldCount + count < 0)
269                    printe("잘못된 값입니다.");
270                else {
271                    Integer targetCount = oldCount + count;
272                    selectedItem.setCount(myId, targetCount);
273                    System.out.println("old: " + oldCount + " | count: " + count + " | Sum: " + (targetCount) + " | Actual: " + selectedItem.getCount(myId));
274                    assertThat(selectedItem.getCount(myId), is(targetCount));
275                    int cnt = 0;
276                    for(int i=0;i< 20;i++) {
277                        Integer cCount = 0;
278                        if(Items.get(i) != null) {
279                            cCount = Items.get(i).getCount(myId);
280                            cCount = (cCount == null ? 0 : cCount);
281                        }
282                        if(cCount != null && cCount > 0)
283                            cnt++;
284                    }
285                    if( cnt > 7 ) {
286                        selectedItem.setCount(myId, 0);
287                        assertThat(selectedItem.getCount(0), is(0));
288                        JOptionPane.showMessageDialog(null,
289                                ("7종류의 음료만 판매할 수 있습니다."), "Message",
290                                JOptionPane.ERROR_MESSAGE);
291                    } else {
292                    //  manageStock.setVisible(false);
293                    //  showMode();
294                    }
295                }
296            }
```

Unit Test – Controller(showMode)



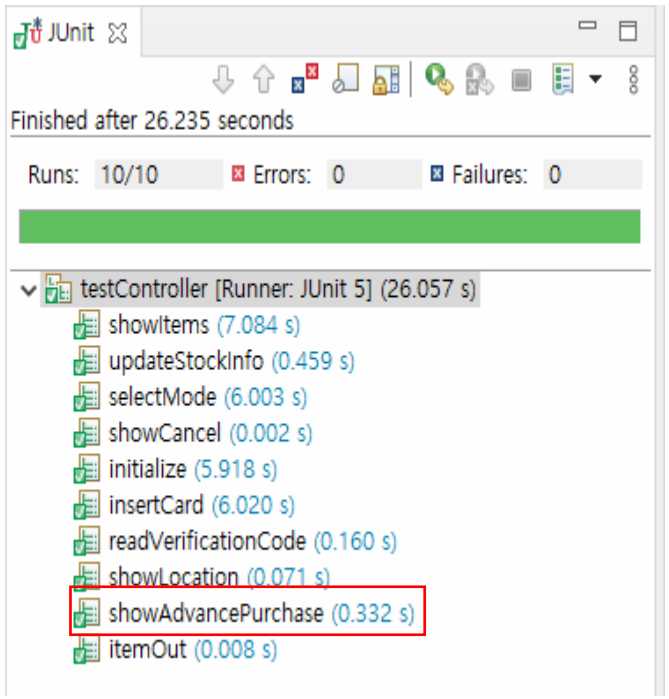

```
308     public void showMode() {
309         JFrame UI = new JFrame();
310         JPanel showMode = new JPanel();
311         showMode.setLayout(new FlowLayout());
312
313         JButton goShowItem = new JButton("현장 구매");
314         JButton goReadVcode = new JButton("선결제코드 입력");
315         JButton goManageStock = new JButton("재고 관리");
316
317         showMode.add(goShowItem);
318         showMode.add(goReadVcode);
319         showMode.add(goManageStock);
320
321         UI.add(showMode);
322         UI.setVisible(true);
323
324         boolean goshowClicked = false;
325         boolean goreadClicked = false;
326         boolean gomanageClicked = false;
327
328         goShowItem.addActionListener( new ActionListener(){
329             @Override
330             public void actionPerformed(ActionEvent e) {
331                 testController.getInstance(myId).selectMode();
332             }
333         });
334         goReadVcode.addActionListener( new ActionListener(){
335             @Override
336             public void actionPerformed(ActionEvent e) {
337                 testController.getInstance(myId).selectMode();
338             }
339         });
340         goManageStock.addActionListener( new ActionListener(){
341             @Override
342             public void actionPerformed(ActionEvent e) {
343                 testController.getInstance(myId).selectMode();
344             }
345         });
346     }
```

Unit Test – Controller(showLocation)




```
364         JPanel underLayer1 = new JPanel(new FlowLayout());
365         JPanel underLayer2 = new JPanel(new FlowLayout());
366         JPanel underLayer3 = new JPanel(new FlowLayout());
367
368         JLabel locate = new JLabel("자판기 위치");
369         JLabel locationSet = new JLabel();
370         String dvmLocations = null;
371         String dvm_ids = "0^1^2^3^4^";
372         String[] dvms = null;
373         if(dvm_ids != null) {
374             if(dvmLocations == null)
375                 dvmLocations = "";
376             dvms = dvm_ids.split("\\^");
377         }
378
379         String[] alpha = {"0", "1", "2", "3", "4"};
380         assertThat(dvms, is(alpha));
381
382         if(dvms != null) {
383             String[] locations = {"신공학관 1층", "새천년관1층", "학생회관1층", "법학관1층", "도서관1층"};
384             for(int i = 0; i < dvms.length; i++) {
385                 Integer dvmId = Integer.parseInt(dvms[i]);
386                 dvmLocations += locations[dvmId];
387
388                 if(i < dvms.length-1)
389                     dvmLocations += ",";
390             }
391         } else {
392             dvmLocations = "없음";
393         }
394         assertNotNull(dvmLocations);
395
396         //      System.out.println("dvm location: " + this.dvm.get(dvm_id).getLocation());
397         locationSet.setText(dvmLocations);
398         ok = new JButton("확인");
```

Unit Test – Controller(showAdvancePurchase(1))

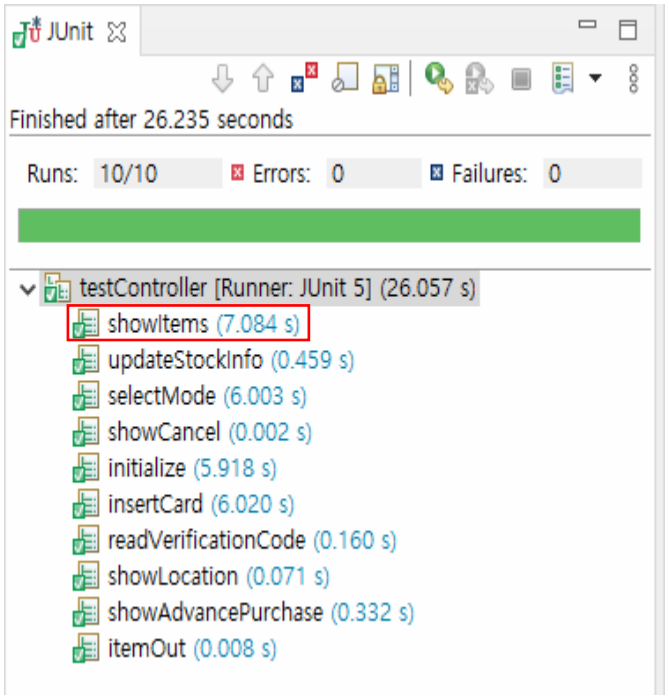



```
476     public void showAdvancePurchase(/*JFrame UI*/) { // (Item item)
477         JFrame UI = new JFrame();
478         JPanel showAdvancePurchase1 = new JPanel(new GridLayout(3,1));
479         JPanel itemPanel = new JPanel(new GridLayout(0,1));
480         JPanel itemLabel = new JPanel(new FlowLayout());
481         JLabel item = new JLabel("음료 :");
482         Item currentItem = new Item(0, 1, Item.prices[0], myId);
483         JLabel itemName = new JLabel(currentItem.getName());
484         assertThat(currentItem.getName(), is(Item.names[0].toString()));
485         itemLabel.add(item);
486         itemLabel.add(itemName);
487 
488         JPanel priceLabel = new JPanel(new FlowLayout());
489         JLabel price = new JLabel("가격 :");
490         JLabel setPrice = new JLabel(currentItem.getPrice().toString() + " 원");
491         priceLabel.add(price);
492         priceLabel.add(setPrice);
493 
494         itemPanel.add(itemLabel);
495         itemPanel.add(priceLabel);
496 
497         JPanel apbPanel = new JPanel(new FlowLayout());
498         JButton AdvancePurcahaseButton = new JButton("선구매");
499         apbPanel.add(AdvancePurcahaseButton);
500 
501         JPanel buttonPanel = new JPanel();
502         cancel = new JButton("취소");
503         JButton location = new JButton("위치보기");
504 
505         buttonPanel.add(cancel);
506         buttonPanel.add(location);
507 
508         showAdvancePurchase1.add(itemPanel);
509         showAdvancePurchase1.add(apbPanel);
510         showAdvancePurchase1.add(buttonPanel);
511 
512         UI.add(showAdvancePurchase1);
513         showAdvancePurchase1.setVisible(true);
514         assertTrue(showAdvancePurchase1.isVisible());//Test Code
```

Unit Test – Controller(showAdvancePurchase(2))

```
517         AdvancePurcahaseButton.addActionListener( new ActionListener(){
518             @Override
519             public void actionPerformed(ActionEvent e) {
520                 showAdvancePurchase1.setVisible(false);
521                 assertFalse(showAdvancePurchase1.isVisible());//Test Code
522                 assertNotNull(UI); // Test Code
523                 insertCard();
524             }
525         });
526         location.addActionListener( new ActionListener(){
527             @Override
528             public void actionPerformed(ActionEvent e) {
529                 showAdvancePurchase1.setVisible(false);
530                 assertFalse(showAdvancePurchase1.isVisible());//Test Code
531                 //current Item을 갖고 있는 dvm id 구하기
532                 Integer myId = new Random().nextInt(5);
533                 HashMap<Integer, Integer> otherStocks = new HashMap<>();
534                 assertNotNull(otherStocks);//Test Code
535                 String dvms = null;
536 
537                 for(int td = 0; td < 5; td++) {
538                     if(otherStocks.get(td) != null) {
539                         if(otherStocks.get(td) > 0) {
540                             if(dvms == null)
541                                 dvms = "";
542                             dvms += td + "^";
543                             assertNotNull(dvms);
544                         }
545                     }
546                 }
547                 showLocation();
548             }
549         });
550 
551         cancel.addActionListener( new ActionListener(){
552             @Override
553             public void actionPerformed(ActionEvent e) {
554                 showAdvancePurchase1.setVisible(false);
555                 showMode();
556             }
557         });
558         AdvancePurcahaseButton.doClick();
559         location.doClick();
560         cancel.doClick();
561         showAdvancePurchase1.setVisible(false);
562     }
```

Unit Test – Controller(showItems)

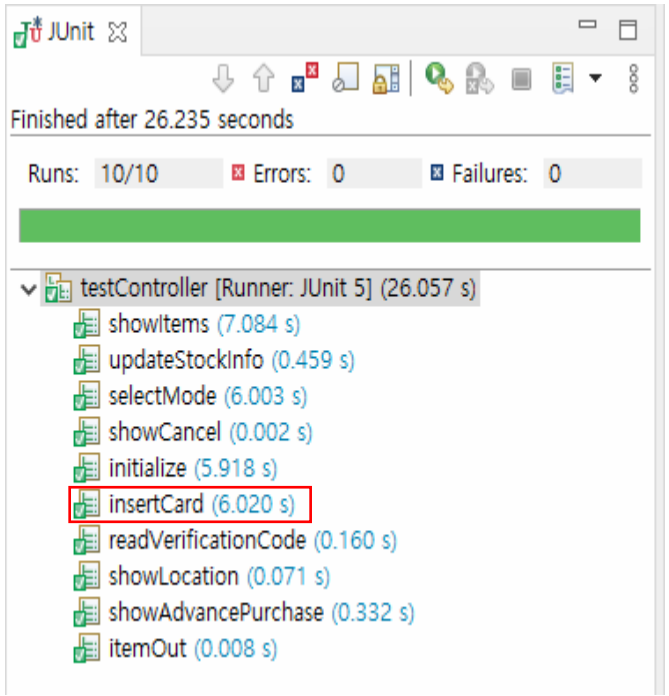



```
581         JButton[] n = new JButton[21];
582         assertNotNull(n); //Test Code
583
584         for(int i=0;i<=20;i++) {
585             if (i == 20) {
586                 n[i] = new JButton();
587                 assertNotNull(n[i]); //Test Code
588                 n[i].setText("취소");
589             } else {
590                 n[i] = new JButton();
591                 assertNotNull(n[i]); //Test Code
592                 n[i].setText(Integer.toString(i+1) + ":" + Item.names[i]);
593             }
594             showItemUnder.add(n[i]);
595         }
596
597         showItems.add(showItemUnder);
598
599         UI.add(showItems);
600         showItems.setVisible(true);
601         assertTrue(showItems.isVisible());
```

Unit Test – Controller(showItems(2))

```
603     List<Item> Items = new ArrayList<>();
604     for(int i = 0; i < 20; i++) {
605         Items.add(new Item(i, 1, Item.prices[i], myId));
606     }
607
608     for(int i=0;i<=20;i++) {
609         n[i].addActionListener( new ActionListener(){
610             @Override
611             public void actionPerformed(ActionEvent e) {
612                 String title = e.getActionCommand();
613                 assertThat(title, is(e.getActionCommand()));
614                 //  테스트 위한 가상 변수
615                 int exIt1 = 0;
616                 int exIt2= 1;
617
618                 if(title.equals("취소")) { //  취소 클
619                     showItems.setVisible(false);
620                     assertNotNull(UI); //Test Code
621                     //showCancel(UI);
622                 } else {
623                     int itemNumOnDisplay = Integer.parseInt(title.split(":")[0]);
624                     int itemid_int = (itemNumOnDisplay)-1;
625
626                     currentItem = Items.get(itemid_int);
627                     assertThat(currentItem.getName(), is(Item.names[itemid_int]));
628                     int stockOK = -1;
629                     if(currentItem == null) {
630
631                     } else if(currentItem.getCount(myId) == null || currentItem.getCount(myId) <= 0)
632                         HashMap<Integer, Integer> otherStocks = new HashMap<>();
633                         assertNotNull(otherStocks); //Test Code
634                         Set<Integer> keys = otherStocks.keySet();
635                         assertNotNull(keys); //Test Code
636                         Iterator<Integer> iter = keys.iterator();
637                         assertNotNull(iter); //Test Code
638
639                         while(iter.hasNext()) {
640                             Integer key = iter.next();
641                             assertNotNull(key); //Test Code
642                             if(otherStocks.get(key) != 0) {
643                                 stockOK = 1;
644                                 break;
645                             }
646                         }
651                         if (stockOK == 0) {
652                             showItems.setVisible(false);
653                             assertNotNull(currentItem); //Test Code
654                             purchase = new Purchase(false, currentItem);
655                             assertThat(purchase.IsAdvanced(), is(false)); //Test Code
656                             insertCard();
657                         } else if (stockOK == 1) {
658                             showItems.setVisible(false);
659                             assertNotNull(currentItem); //Test Code
660                             purchase = new Purchase(true, currentItem);
661                             assertThat(purchase.IsAdvanced(), is(true)); //Test Code
662                             showAdvancePurchase();
663                         } else {
664                             showItems.setVisible(false);
665                             printe("품절입니다.");
666                             showCancel();
667                         }
668                     }
669                 }
670             });
671             assertNotNull(n[i].getActionListeners()); //Test Code
672             n[i].doClick();
673         }
674         showItems.setVisible(false);
675     }
```

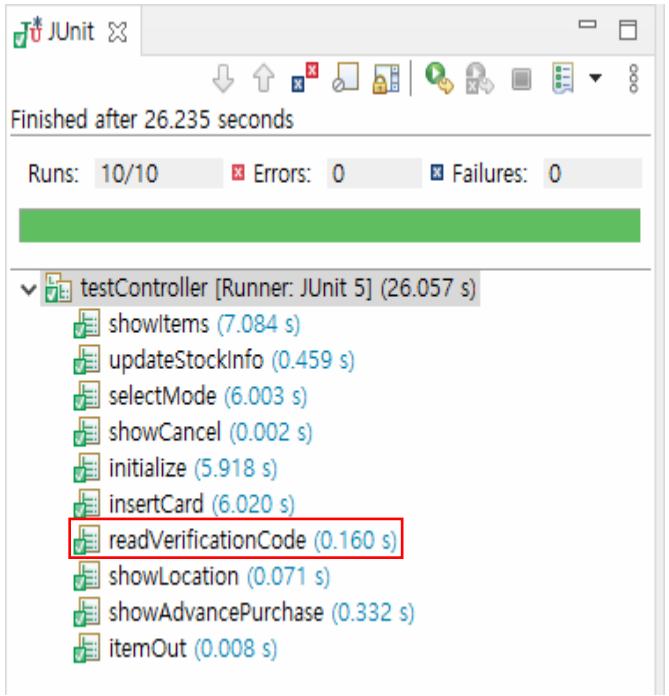
Unit Test – Controller(insertCard)




```
681    public void insertCard(/*JFrame UI*/) { //
682        JFrame UI = new JFrame();
683        SimpleDateFormat format2 = new SimpleDateFormat ( "yyyy년 MM월dd일 HH시mm분ss초");
684        Date time = new Date();
685        String time2 = format2.format(time);
686        if(currentItem == null) {
687            selectMode();
688            System.out.println("음료가 선택되지 않았습니다.");
689            return;
690        }
691        int itemPrice = currentItem.getPrice();
692        assertThat(itemPrice, is(currentItem.getPrice())); //Test Code
693        JPanel insertCard = new JPanel();
694        insertCard.setLayout(new GridLayout(4,0));
695
696        JPanel buttonPanel = new JPanel(new FlowLayout());
697        JPanel itemAndPrice = new JPanel(new FlowLayout());
698
699        JLabel insert = new JLabel();
700        JLabel price = new JLabel();
701        JLabel setPrice = new JLabel();
702        JLabel date = new JLabel();
703        cancel = new JButton("취소");
704        JButton purchaseResult = new JButton("결제 테스트(1% 실패)");
705        JButton failTest = new JButton("실패 테스트");
706
707        insert.setText("카드를 삽입해주세요.");
708        price.setText("결제금액 :");
709        setPrice.setText(Integer.toString(itemPrice) + " 원");
710        date.setText(time2);
711
712        itemAndPrice.add(price);
713        itemAndPrice.add(setPrice);
714        assertNotNull(itemAndPrice.getComponents()); //Test Code
715        buttonPanel.add(cancel);
716        buttonPanel.add(purchaseResult);
717        buttonPanel.add(failTest);
718        assertNotNull(buttonPanel.getComponents()); //Test Code
719
720        insertCard.add(insert);
721        insertCard.add(itemAndPrice);
722        insertCard.add(date);
723        insertCard.add(buttonPanel);
724        assertNotNull(insertCard.getComponents()); //Test Code
```

```
733        boolean validation = true;
734        assertTrue(validation);//Test Code
735        List<Item> Items = new ArrayList<>();
736        for(int i = 0; i < 20; i++) {
737            Items.add(new Item(i, 1, Item.prices[i], myId));
738        }
739        purchaseResult.addActionListener( new ActionListener(){
```

Unit Test – Controller(readVerificationCode(1))



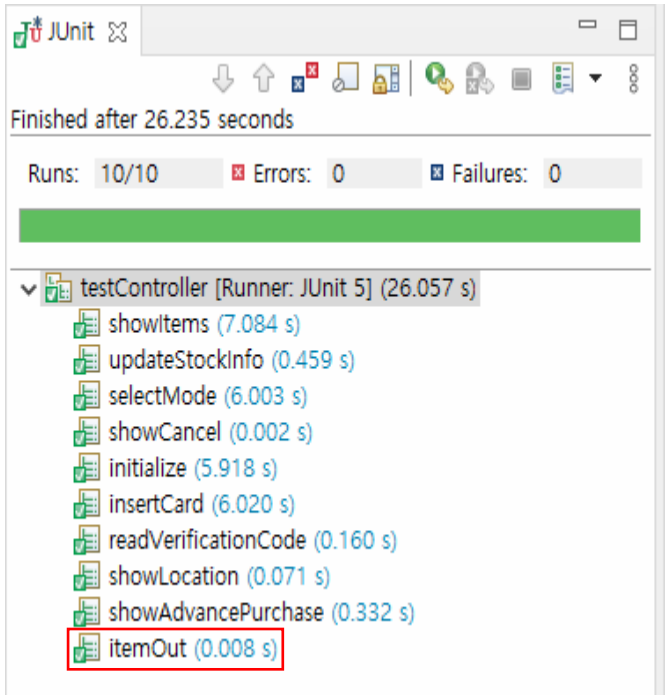

```
824         for(int i =1; i<=10; i++) {
825             int realvalue = i;
826             if(i == 10) {
827                 realvalue = 0;
828             }
829             keypad[realvalue] = new JButton(Integer.toString(realvalue));
830             assertNotNull(keypad[realvalue]); //Test Code
831             numbers.add(keypad[realvalue]);
832 
833             keypad[realvalue].addActionListener( new ActionListener(){
834 
835                 @Override
836                 public void actionPerformed(ActionEvent e) {
837                     String preset = "인증번호를 입력하세요.";
838                     String vode = e.getActionCommand();
839 
840                     if(vcode.getText().equals(preset)) {
841                         vcode.setText(vode);
842                     } else if(!vcode.getText().equals(preset)) {
843                         if(vcode.getText().length() < 6) {
844                             vode = vcode.getText() + vode;
845                             vcode.setText(vode);
846                         }
847                     }
848                 }
849             });
850             assertNotNull(keypad[realvalue].getActionListeners()); //Test Code
851         }
```

Unit Test – Controller(readVerificationCode(2))

```
853         readVcode.add(txtPanel);
854         readVcode.add(numbers);
855         readVcode.add(button);
856         UI.add(readVcode);
857         readVcode.setVisible(true);
858         assertTrue(readVcode.isVisible());//Test Code
859
860         vcode.setHorizontalAlignment(JLabel.CENTER);
861
862         cancel.addActionListener( new ActionListener(){
863             @Override
864             public void actionPerformed(ActionEvent e) {
865                 readVcode.setVisible(false);
866                 //showCancel(UI);
867             }
868         });
869         assertNotNull(vcode.getActionListeners()); //Test Code
870
871         ok.addActionListener( new ActionListener(){
872             @Override
873             public void actionPerformed(ActionEvent e) {
874                 String preset = "인증번호를 입력하세요.";
875                 vcode.setText(new Random().nextInt(2) == 0 ? "123456" : new Random().nextInt(999999) + "");
876                 if(!vcode.getText().contentEquals(preset)) {
877                     int inputCode = 0;
878                     try {
879                         inputCode = Integer.parseInt(vcode.getText());
880                         Integer myId = new Random().nextInt(5);
881                         int codeItem = VerificationCode.getInstance(myId).checkCode( inputCode );
882                         if(codeItem > -1) {
883                             readVcode.setVisible(false);
884                             currentItem = Items.get(codeItem);
885                             assertNotNull(currentItem);
886                             VerificationCode.getInstance(myId).resetCode(inputCode, true);
887                             itemOut();
888                             showMode();
889                         } else {
890                             printe("잘못된 코드입니다.");
891                         }
892                     } catch (NumberFormatException er) {
893                         printe("숫자만 입력 가능합니다.");
894                     }
895                 } else {
896                     printe("코드를 입력해주세요.");
897                 }
```

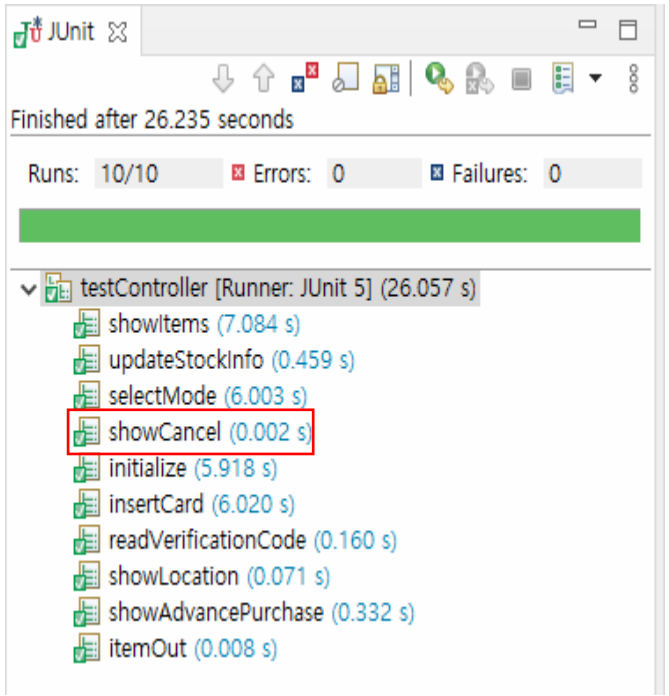
```
901         assertNotNull(ok.getActionListeners()); //Test Code
902
903         cancel.doClick();
904         ok.doClick();
905         readVcode.setVisible(false);
906     }
```

Unit Test – Controller(itemOut)



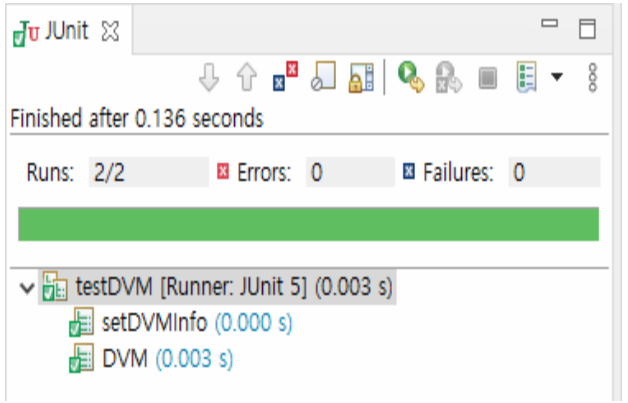

```
912     public void itemOut() {
913         //print(currentItem.getName() + "가 나왔습니다.");
914         //      JOptionPane.showMessageDialog(null,
915         //              (currentItem.getName() + "가 나왔습니다."), "Message",
916         //              JOptionPane.ERROR_MESSAGE);
917         currentItem = null;
918         assertNull(currentItem); //Test Code
919         purchase = null;
920         assertNull(purchase); //Test Code
921     }
```

Unit Test – Controller(showCancel)




```
924     public void showCancel(/*JFrame UI*/) {
925         // TODO implement here
926         JFrame UI = new JFrame();
927 
928         currentItem = null;
929         assertNull(currentItem); //Test Code
930         purchase = null;
931         assertNull(purchase); //Test Code
932     }
```

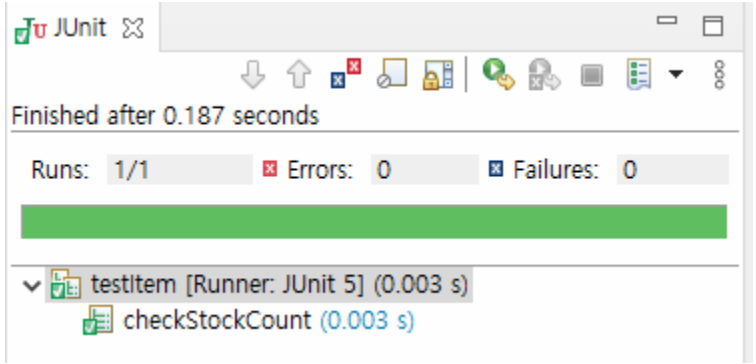
Unit Test – DVM




```java
77    @Test
78    public void setDVMInfo() {
79        System.out.println("called set dvm info");
80        if(Objects.isNull(id) && Objects.isNull(location)) {
81            //id, location 중복체크
82            assertNotNull(id);
83            assertNotNull(location);
84            assertNotNull(items);
85            this.id = id;
86            this.location = location;
87            this.items = items;
88            System.out.println("set done");
89
90        } else {
91
92        }
93    }
```

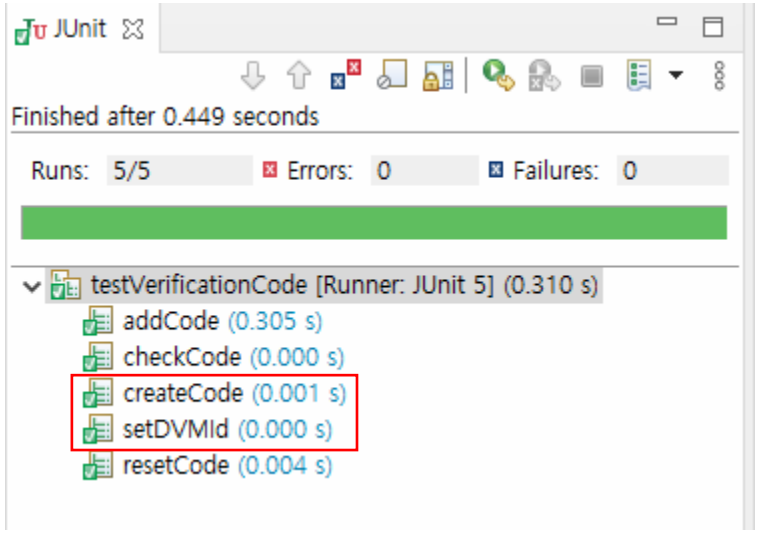
```java
22    public void DVM() {
23        this.id = id;
24        this.location = location;
25        this.items = items;
26    }
```

Unit Test – Item




```
164     public void checkStockCount() {
165
166         Map<Integer, Integer> map=new HashMap<Integer, Integer>(); // <DVM_ID, 음료개수>
167         Map<Integer, String> tmap = new HashMap<Integer, String>();
168         int dvm_Id = new Random().nextInt(5);
169         if(getCount(dvm_Id) == null || getCount(dvm_Id) <= 0) {
170             Message msg = new Message();
171             //broadcast
172             tmap = msg.sendMessage(1, "" + this.id, dvm_Id, dvm_Id);
173
174             if(tmap.size() != 0) {
175                 String responseMessage = tmap.get(2);
176
177                 assertNotNull(responseMessage);
178
179                 if(responseMessage.contains("#")) {
180
181                     String[] msg_body = tmap.get(2).split("#");
182                     for(int i=0;i<msg_body.length;i++) {
183
184                         String targetItemId_str = msg_body[i].split("\\^")[0];
185                         String targetCode_str = msg_body[i].split("\\^")[1];
186
187                         assertNotNull(targetItemId_str);
188                         assertNotNull(targetCode_str);
189
190                         map.put(Integer.parseInt(targetItemId_str), Integer.parseInt(targetCode_str));
191                     }
192                 } else {
193                     String targetItemId_str = responseMessage.split("\\^")[0];
194                     String targetCode_str = responseMessage.split("\\^")[1];
195
196                     assertNotNull(targetItemId_str);
197                     assertNotNull(targetCode_str);
198                     map.put(Integer.parseInt(targetItemId_str), Integer.parseInt(targetCode_str));
199                 }
200             }
201         }
202         else {
203             map.put(0,getCount());
204         }
205         assertNotNull(map);
206     }
207 }
```

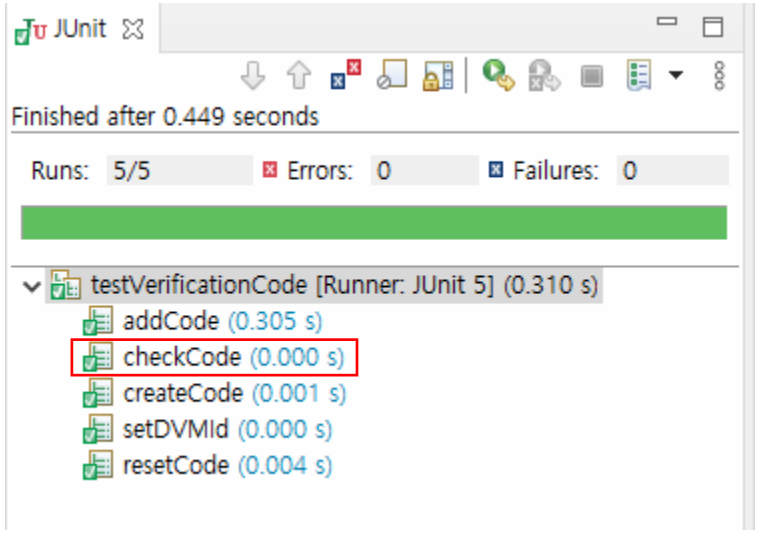
Unit Test – VerificationCode




```
18    @Test
19    public void setDVMId() {
20        Random rand=new Random();
21        assertNotNull(rand);
22        int i=rand.nextInt(5);
23        assertNotNull(i);
24        dvmId = i;
25        assertNotNull(dvmId);
26    }
```

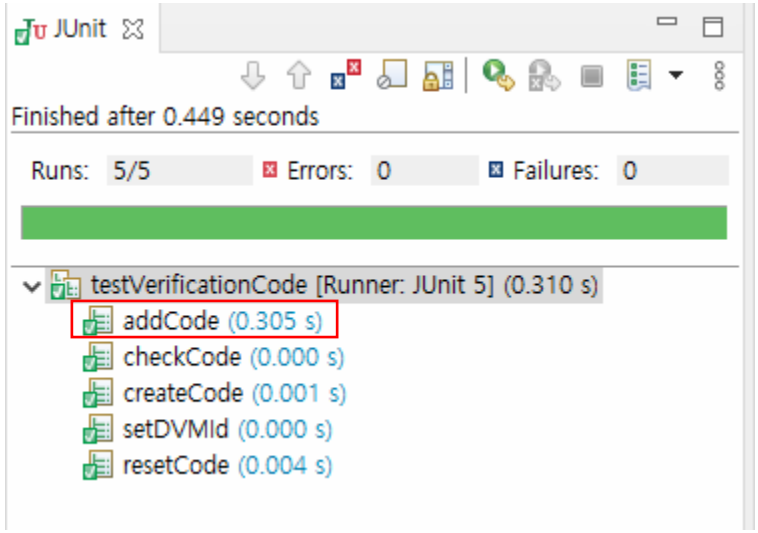
```
58    @Test
59    public void createCode() {
60        // TODO implement here
61        Random rand = new Random();
62        assertNotNull(rand);
63        Integer code=rand.nextInt(999999);
64        assertNotNull(code);
65    }
```

Unit Test – VerificationCode



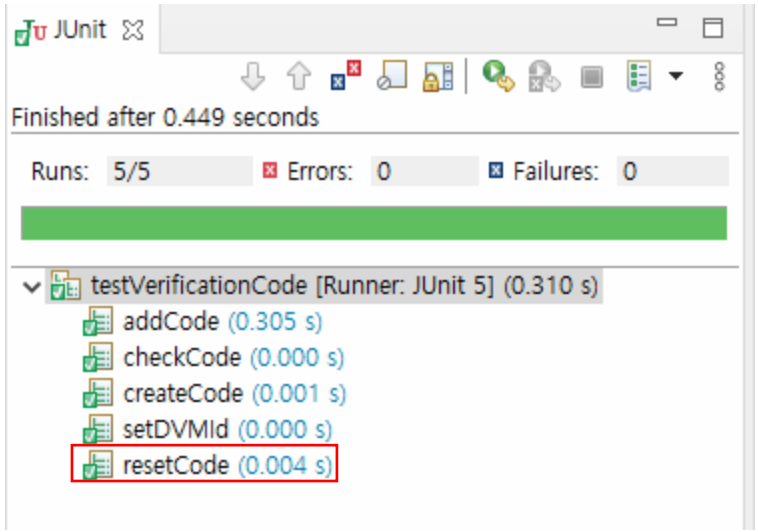

```
70    @Test
71    public void checkCode() {
72        // TODO implement here
73        Integer code= new Random().nextInt(999999);
74        assertNotNull(code);
75
76        Set<Integer> itemIds = codeList.keySet();
77        assertNotNull(itemIds);
78        Iterator<Integer> iter = itemIds.iterator();
79        assertNotNull(iter);
80        int isDone = -1;
81        while(iter.hasNext()) {
82            Integer key = iter.next();
83            assertNotNull(key);
84            ArrayList<Integer> itemCodes = codeList.get(key);
85            assertNotNull(itemCodes);
86            for(int i = 0; i < itemCodes.size(); i++) {
87                if(itemCodes.get(i).equals(code)) {
88                    assertEquals(itemCodes.get(i),code);
89                    isDone = key;
90                    assertNotNull(isDone);
91                    break;
92                }
93            }
94            assertNotNull(isDone);
95            if(isDone > -1)
96                break;
97        }
98        assertNotNull(isDone);
99    }
```

Unit Test – VerificationCode



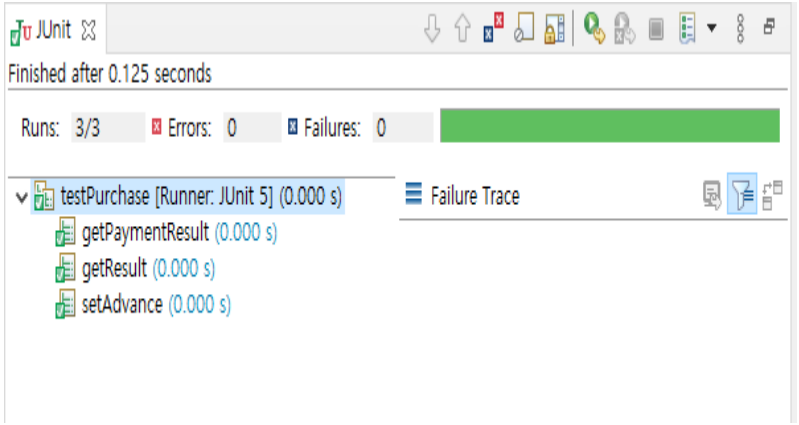

```
105    @Test
106    public void addCode() {
107        // TODO implement here
108        Random random=new Random();
109        Integer itemid=random.nextInt(20);
110        assertNotNull(itemid);
111        Integer code=random.nextInt(999999);
112        assertNotNull(code);
113        boolean flag=true;
114        try {
115            ArrayList<Integer> oldList = (codeList.get(itemid) == null ? new ArrayList<Integer>() : codeList.get(itemid));
116            assertNotNull(oldList);
117            Item item = Controller.getInstance(dvmId).getItems(dvmId).get(itemid);
118            if(item != null) {
119                assertNotNull(item);
120                Integer count = item.getCount(dvmId);
121                if(count != null && count > 0) {
122                    assertNotNull(count);
123                    oldList.add(code);
124                    codeList.put(itemid, oldList);
125                    assertNotNull(codeList);
126                    item.setCount(dvmId, count-1);
127                }
128            }
129            assertTrue(flag);
130        } catch (Exception e) {
131            System.err.println("Error occured while adding code");
132            flag=false;
133            assertFalse(flag);
134        }
135    }
```

Unit Test – VerificationCode




```
148             Set<Integer> itemIds = codeList.keySet();
149             Iterator<Integer> iter = itemIds.iterator();
150             assertNotNull(itemIds);
151             assertNotNull(iter);
152             while(iter.hasNext()) {
153                 Integer key = iter.next();
154                 assertNotNull(key);
155                 ArrayList<Integer> itemCodes = codeList.get(key);
156                 assertNotNull(itemCodes);
157                 for(int i = 0; i < itemCodes.size(); i++) {
158                     if(itemCodes.get(i).equals(code)) {
159                         assertEquals(itemCodes.get(i),code);
160                         itemid = key;
161                         assertNotNull(itemid);
162                         itemCodes.remove(i);
163                         codeList.put(itemid, itemCodes);
164                         assertNotNull(codeList);
165                         isDone = true;
166                         break;
167                     }
168                 }
169                 assertNotNull(isDone);
170                 if(isDone)
171                     break;
172             }
173         }
174         catch(NullPointerException e) {
175             System.err.println("There is no code: " + code);
176             //return false;
177         }
178             /* broadcast */
179         if(isDone) {
180                 if(broadcast) {
181                     Message net = new Message();
182                     assertNotNull(net);
183                     //net.sendMessage(4, (itemid +"^"+ code), dvmId, dvmId);
184                 } else {
185                     Item item =Controller.getInstance(dvmId).getItems(dvmId).get(itemid);
186                     if(item != null) {
187                         assertNotNull(item);
188                         Integer count = item.getCount(dvmId);
189                         count = (count == null ? 0 : count);
190                         assertNotNull(count);
191                         item.setCount(dvmId, count+1);
```

Unit Test – Purchase

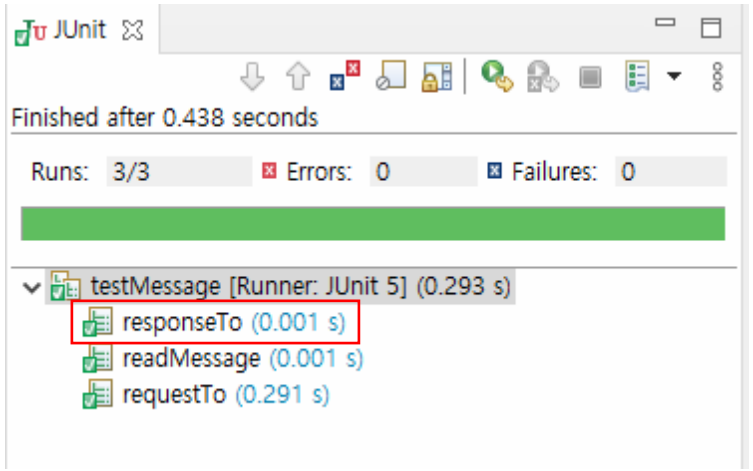



```
43      public void getResult() {
44          // TODO implement here
45          Random random = new Random();
46          assertNotNull(random);
47          int temp=random.nextInt(100)+1;
48          assertNotNull(temp);
49      }
```

```
55      public void getPaymentResult() {
56          // TODO implement here
57          this.isAdvance=true;
58          assertTrue(this.isAdvance);
59          this.isAdvance=false;
60          assertFalse(this.isAdvance);
61      }
62
```

```
76      public void setAdvance() {
77          // TODO implement here
78          this.isAdvance=true;
79          assertTrue(this.isAdvance);
80          this.isAdvance=false;
81          assertFalse(this.isAdvance);
82      }
```

Unit Test – Message(responseTo(1))

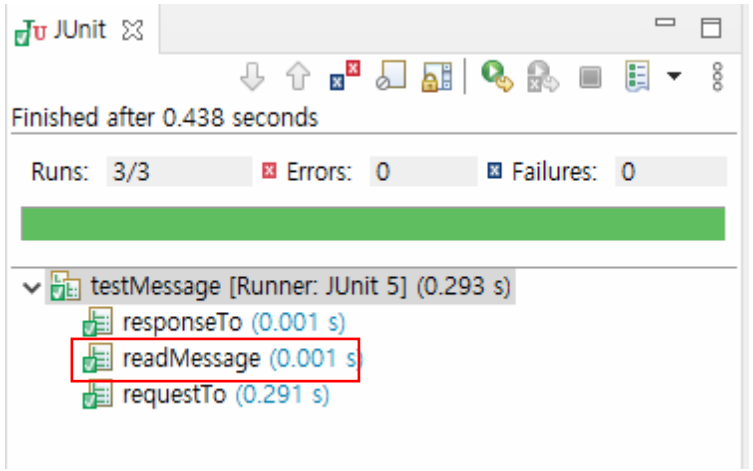



```
198         case 1:
199         {
200             Integer targetItemId = Integer.parseInt(body);
201             assertNotNull(targetItemId);
202
203             Item tarItem = new Item(targetItemId, targetItemId, targetItemId, targetItemId);
204
205             String message = dst_id + "^0";
206             assertNotNull(message);
207
208             if(tarItem != null) {
209                 Integer cCount = tarItem.getCount(dst_id);
210 //              System.out.println("cCount: "+ cCount);
211                 message = dst_id + "^" + cCount;
212             }
213
214             result.put(2, message);
215 //          System.out.println("result: " + result.toString());
216             break;
217         }
218         //선결제 코드 등록
219         case 3:
220         {
221             if(body.contentEquals("success")) {
222                 break;
223             }
224             String targetItemId_str = body.split("\\^")[0];
225             String targetCode_str = body.split("\\^")[1];
226
227             assertNotNull(targetItemId_str);
228             assertNotNull(targetCode_str);
229
230             Integer targetItemId = Integer.parseInt(targetItemId_str);
231             Integer targetItemCode = Integer.parseInt(targetCode_str);
232
233             assertNotNull(targetItemId);
234             assertNotNull(targetItemCode);
235             VerificationCode.getInstance(dst_id).addCode(targetItemId, targetItemCode);
236             result.put(3, "success");
237             break;
238         }
```

Unit Test – Message(responseTo(2))

```
240         case 4:
241         {
242             if(body.contentEquals("success")) {
243                 break;
244             }
245             Integer targetItemId = Integer.parseInt(body.split("\\^")[0]);
246             Integer targetItemCode = Integer.parseInt(body.split("\\^")[1]);
247 
248             assertNotNull(targetItemId);
249             assertNotNull(targetItemCode);
250 
251             VerificationCode.getInstance(dst_id).resetCode(targetItemCode, false);
252             result.put(4, "success");
253             break;
254         }
255         //주소 요청
256         case 5:
257         {
258             System.out.println("위치 응답 생성중");
259             String message = dst_id + "^" + Controller.getInstance(dst_id).getDVMs().get(dst_id).getLocation();
260             assertNotNull(message);
261             System.out.println("위치 응답: " + message);
262             result.put(6, message);
263             break;
264         }
265         //음료 판매 확인 : 음료 누가 파나요?~
266         case 7:
267         {
268             Integer targetItemId = Integer.parseInt(body);
269 
270             Item tarItem = items.get(targetItemId);
271             int count = tarItem.getCount(dst_id);
272 
273             String message = (count == 0 ? "-1" : ("0"));
274             assertNotNull(message);
275 
276             result.put(8, message);
277             break;
278         }
```

Unit Test – Message(requestTo(1))




```
69          if(dst_id == src_id) {
70              //
71              response = new HashMap<Integer, String>();
72              for(int i = 0; i < 5; i++) {
73                  if(i != src_id) {
74                      targetDVM = Controller.getInstance(i);
75
76                      assertNotNull(targetDVM);
77                      Map<Integer, String> retFromDVM = new HashMap<>();
78                      if(retFromDVM.get(2) != null) {
79                          int tCount = Integer.parseInt(retFromDVM.get(2).split("\\^")[1]);
80                          assertNotNull(tCount);
81                          if(tCount > 0) {
82                              if(response.get(2) == null) {
83                                  response.put(2, (retFromDVM.get(2) + "#"));
84                              } else {
85                                  String old = response.get(2);
86                                  response.put(2, old + (retFromDVM.get(2) + "#"));
87                              }
88                          } else {
89                              if(response.get(2) == null)
90                                  response.put(2, "0^0#");
91                          }
92                      } else if (retFromDVM.get(3) != null) {
93                          // success 만 올거임.
94
95                          if(!retFromDVM.get(3).contentEquals("success")) {
96                              //메시지 전달 실패이고
97                              //^ 조차 없으면(메시지 요청이 아니면)
98                              if(!retFromDVM.get(3).contains("^"))
99                                  i--;
100                         }
101
102                     } else if (retFromDVM.get(4) != null) {
103                         // success 만 올거임.
104                         if(!retFromDVM.get(4).contentEquals("success")) {
105                             //메시지 전달 실패이고
106                             //^ 조차 없으면(메시지 요청이 아니면)
107                             if(!retFromDVM.get(4).contains("^"))
108                                 i--;
109                         }
110                     } else if (retFromDVM.get(6) != null) {
111                         if(response.get(6) == null) {
112                             response.put(6, (retFromDVM.get(6) + "#"));
```

Unit Test – Message(requestTo(2))

```
133             targetDVM = Controller.getInstance(dst_id);
134
135             assertNotNull(targetDVM);
136
137             response = targetDVM.readMessage(this.type, this.body, src_id, dst_id);
138
139             //2,3,4,6,8
140             //2, dvm_id + "^" + count
141             //3, item^code / success
142             //4, item^code / success
143             //6, dvm_id + "^" + dvm_location
144             //8, -1 / dvm_id
145
146             if(response.get(2) != null) {
147                 // Nothing just return
148             } else if (response.get(3) != null) {
149                 // Nothing just return
150             } else if (response.get(4) != null) {
151                 // Ntohing just return
152             } else if (response.get(6) != null) {
153                 // Nothing just return
154             } else if (response.get(8) != null) {
155                 // Nothing just return
156             } else {
157                 response.put(0, "Unknown");
158             }
159         }
160         assertNotNull(response);
161     }
```

Class Diagram