# 3rd Cycle

객체지향개발방법론

201711300 류창현
201711314 송주한
201915005 남승협
201911231 강현수

# 목차

Table of contents

# System Testing

## Brute Force Testing

| ID | testing result | bug issue | ID | testing result | bug issue |
|---|---|---|---|---|---|
| 2nd testing fail 이슈들 | | | 2nd testing 은 pass 했지만 버그 발생 | | |
| BFT-17 | fail >> fail | VMP-167 | BFT-43 | pass >> pass | VMP-165 |
| BFT-37 | pass >> fail | VMP-168 | BFT-48 | fail >> pass | VMP-166 |
| BFT-40 | fail >> fail | VMP-169 | | | |

# System Testing

## Brute Force Testing

### BFT-17 테스팅 버그

**설명**

카드 잔액이 없는 경우에서 새로운 vm을 실행시키고 제품을 샀을 경우 다시 돈이 초기화가 되어버림

**-> 현 DVM_SW에서는 DB 환경을 구축하지 않았기 때문에 소프트웨어를 동작할 때마다 돈이 초기화되는 것은 정상적인 동작이다.**

### BFT-37 테스팅 버그

첨부 하위 작업 생성 이슈 연결

**설명**

Location 변경 시 아무런 변화가 없음
사용하지 않으려면 삭제 요망

**-> 해당 기능 UI 및 Logic Layer 함수 삭제 완료.**

# System Testing

## Brute Force Testing

### BFT-40 테스팅 버그

첨부 하위 작업 생성 이슈 연결 ...

**설명**

연락처 변경을 필요 없다고 판단하여 문서에서 삭제 했으면
프로그램에서도 삭제 요망

3.2.3.4 Functional requirement 3.4 Edit Contact

- System 은 물리 키보드로 입력 받은 연락처 정보를 통해 연락처를 수정해야 한다.

**->연락처 변경의 경우 문서에서 삭제된 적이 없으며 삭제된 부분은 연락처 정보 수정에서**
**" 지정된 범위에서 벗어난 경우 " 부분이 삭제된 것이다.**

# System Testing

## Brute Force Testing

테스트 실행

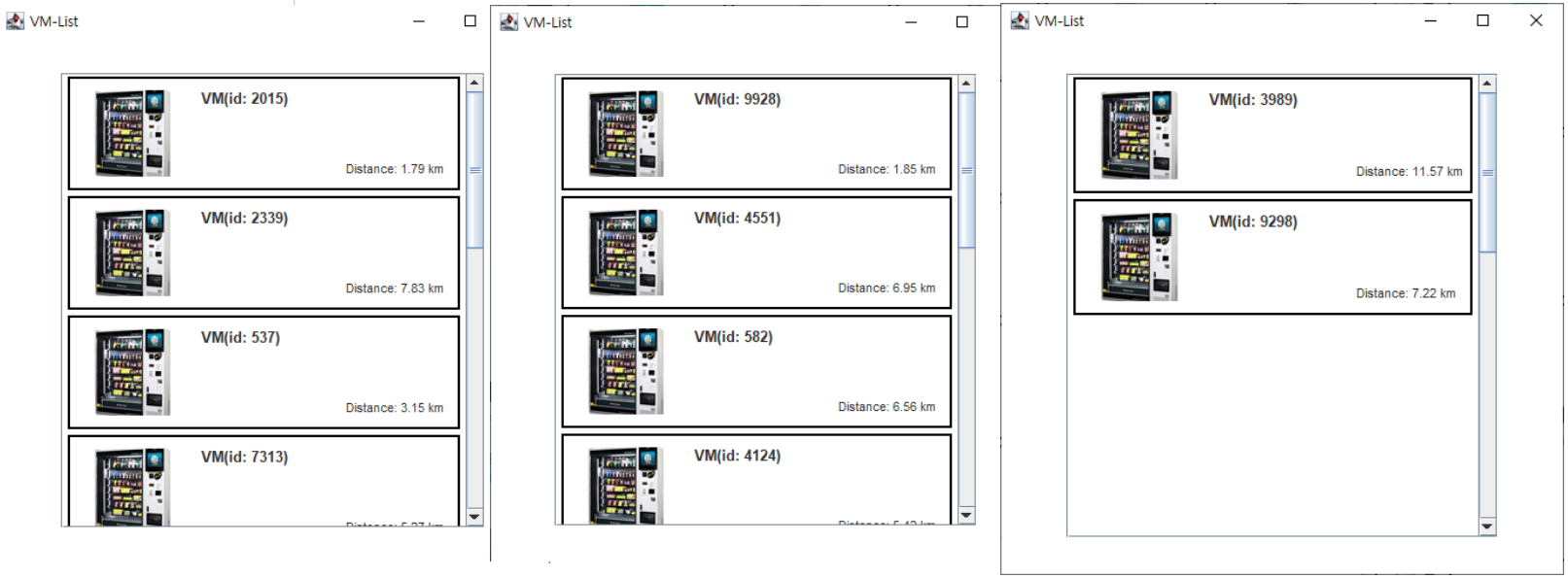
Execution Status : PASS

결함들 : VMP-165

주석 :

결함 발견

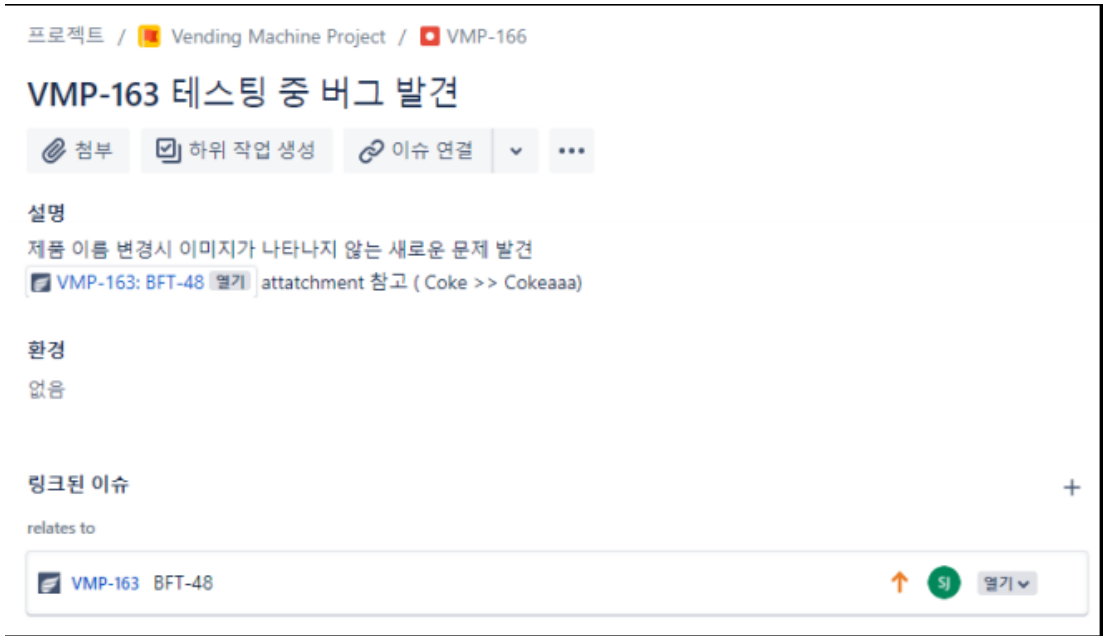
vm 리스트에 같은 vm 들이 두 개씩 나타나는 화면을 확인함




-> 해당 내용에 대하여 확인해봤으나 해당 결함을 찾지 못함.

# System Testing

## Brute Force Testing




-> 구현한 SW에서 이미지는 해당 제품 이름을 통해 맞는 이미지가 있으면 제품 이미지를 띄워주는 기능입니다.
따라서, “Cokeaaa” 라는 이름의 제품 이미지가 준비되어 있지 않아 나타나는 현상입니다.
없는 이미지의 경우, default 이미지를 만들어서 출력해주도록 변경했습니다.

# System Testing

## Category Partitioning Testing

| ' | 1st testing | 2nd testing | ID | 1st testing | 2nd testing |
|---|---|---|---|---|---|
| CPT-1 | pass | pass | CPT-21 | fail | pass |
| CPT-2 | fail | pass | CPT-22 | fail | pass |
| CPT-3 | pass | pass | CPT-23 | blocked | |
| CPT-4 | fail | pass | CPT-24 | blocked | |
| CPT-5 | pass | pass | CPT-25 | pass | pass |
| CPT-6 | pass | pass | CPT-26 | pass | pass |
| CPT-7 | fail | fail | CPT-27 | pass | pass |
| CPT-8 | pass | pass | CPT-28 | pass | pass |
| CPT-9 | fail | pass | CPT-29 | blocked | |
| CPT-10 | fail | pass | CPT-30 | blocked | |
| CPT-11 | pass | pass | CPT-31 | pass | pass |
| CPT-12 | pass | pass | CPT-32 | pass | pass |
| CPT-13 | pass | pass | CPT-33 | pass | pass |
| CPT-14 | pass | pass | CPT-34 | fail | pass |
| CPT-15 | pass | pass | CPT-35 | blocked | |

# System Testing

## Category Partitioning Testing

Admin Mode 에서 VM 정보를 갱신할 때 잘못된 값을 넣으면 갱신되지 않는지를 테스팅하는 케이스

결과 >> VM ID 를 00 으로 입력해도 정보가 갱신됨 (Test Fail)

### 3.2.3.3 Functional requirement 3.3 Edit VM's Info

- System 은 물리 키보드로 입력 받은 자판기 정보를 통해 자판기 정보를 수정해야 한다.
- VM ID 의 경우 0~999 사이의 정수로만 수정이 가능하며 실제로 적용되는 ID 는 관리자가 입력한 정수 + 프로그램 시작 사전작업에서 설정한 Port 번호의 마지막 번호로 변경된다.

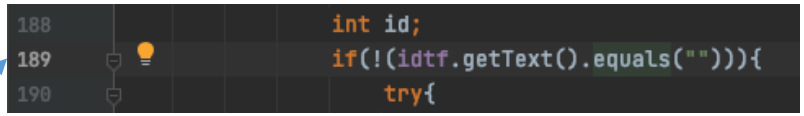
-> 00을 넣어도 정보가 갱신되는 이유는 SRS에 0~999사이의 정수로 수정이 가능하다고 언급이 되어있기 때문

# Code Smell

## Bad practice Warnings

- **Bad practice Warnings**

| Dir. | Line | Review |
|---|---|---|
| AdminFrame | 188 | == 이나 =! 대신 .equlas() 사용 권장 |
| | 247 | |
| Message | 60-73 | 함수 이름의 첫번째 알파벳은 소문자 작성 권장 |
| CheckStock | 215-221 | |
| NotifyVMsInfo | 329-364 | |
| RespondSell | 274-277 | |





**AdminFrame, Message, VM Classes에 대하여 위와 같이 수정함.**

# Code Smell

## Performance Warnings

- **Performance Warnings**

| Dir. | Line | Review |
|---|---|---|
| VM | 11 | 사용하지 않는 field variable 지우는 것을 권장 |
| | 12 | |
| | 16 | |
| | 26 | |
| ReceieveMail | 42 | vlaueof -> parseInt()로 바꾸는 것을 권장 |
| StartDialog | 47 | |
| VM | 321 | |

**삭제 완료.**

```
323         tempD[0] = Double.parseDouble(tempS[0]);
324         tempD[1] = Double.parseDouble(tempS[1]);
325         int markID = Integer.parseInt(tempS[2]);
326         controller.showVMFrame(new VM(mailBox.get(0).getSrc_id(),markID, tempD));
327         mailBox.remove( index: 0);
```

**RecieveMail, StartDialog, VM Classes에 대하여 위와 같이 수정함.**

# Code Smell

## PMD

총 251 개의 Warning 을 찾았으며 아래에는 필요한 Review 를 정리해 놓음.

| Dir. | Line | Review |
|---|---|---|
| AdminFrame<br>CardListFrame<br>VM<br>VMFrame<br>VMProject | - | 사용하지 않는 Import 를 지우는 것을 권장 |
| Message | 21, 24 | 사용하지 않는 Local Variable 지우는 것을 권장 |
| VM | 399 | 사용하지 않는 Local Variable 지우는 것을 권장 |
| DrinkPanel | 117 | Switch 문에는 default 가 없는 것은 권장하지 않음 |
| VM | 102 | Switch 문에는 default 가 없는 것은 권장하지 않음 |
| Item<br>VM | - | 사용하지 않는 Private Field 를 지우는 것을 권장 |
| VMProject | 15 | 사용하지 않는 Private Method 를 지우는 것을 권장 |
| VM | 277 | void 형의 method 에서 return 값을 삭제 |
| VM | 281 | void 형의 method 에서 return 값을 삭제 |
| VM | 279 | 사용하지 않는 method 이면 지우는 것을 권장 |

```
114                 case "Mint Coffee":
115                     icon = new ImageIcon(imgpath[19]);
116                     break;
117                 default:
118                     break;
119         }
```

```
91              default: // Default is equal to case 0
92                  for (int i = 0; i < 7; i++){
93                      itemArray[i] = new Item(drinkArray[i].getName(), drinkArray[i].getPrice(), stock: 10);
94                  }
95                  break;
96          }
```

**DrinkPanel, VM Classes에 대하여 위와 같이 수정함.**

**이외의 삭제 요구 사항을 코드에 반영함.**

# Code Smell

## Code Smell (Blocker)

**Code Smell (Blocker)**

| Dir. | Line | Review |
|---|---|---|
| AdminDialog | 16 | 변수 명을 parent 로 설정 할 경우 Component 내 field 변수 parent 와 겹치므로 변경 해야함 |
| AdminFrame | 29 | |
| CardDialog | 7 | |
| CodeDialog | 15 | |
| DrinkDialog | 10 | |
| PaymentDialog | 12 | |
| StartDialog | 16 | |
| VMFrame | 14 | |
| VM | 131,131 | Return 이 무조건 true 로 가게 되어있다. 사용하지 않는 함수 일 경우 삭제 바람 |

```
MainFrame parent = null;
```

```
MainFrame mainframe = null;
```

**위의 그림처럼 변수명을 parent로 설정한 경우가 있었음.**
**-> 아래 그림처럼 변수명을 모두 수정함.**

**사용되지 않는 함수로 삭제.**

# Code Smell

## Code Smell (Critical)

**Code Smell(Critical)**

| Dir. | Line | Review |
|---|---|---|
| VM | 86-101 | Switch 문을 사용할 때 default case 를 추가하는 것을 권장 |

```
switch (division){
    case 0:
        for (int i = 0; i < 7; i++){
            itemArray[i] = new Item(drinkArray[i].getName(), drinkArray[i].getPrice(), stock: 10);
        }
        break;
    case 1:
        for (int i = 0; i < 7; i++){
            itemArray[i] = new Item(drinkArray[i+7].getName(), drinkArray[i+7].getPrice(), stock: 10);
        }
        break;
    case 2:
        for (int i = 0; i < 7; i++){
            itemArray[i] = new Item(drinkArray[i+13].getName(), drinkArray[i+13].getPrice(), stock: 10);
        }
        break;
    default: // Default is equal to case 0
        for (int i = 0; i < 7; i++){
            itemArray[i] = new Item(drinkArray[i].getName(), drinkArray[i].getPrice(), stock: 10);
        }
        break;
}
```

**해당 부분 default case 추가.**

# Code Smell

## Code Smell (Major)

**Code Smell(Major)**

| Dir. | Line | Review |
|---|---|---|
| CardDialog | 83 | Field 이름과 겹치므로(재선언됨) 변경 바람 |
| | 85 | |
| | 130 | If 문을 두번 사용할 필요가 없으므로 합치기 바람 |
| Code | 2 | Class 이름과 겹치는 field 는권장하지 않음 |
| DrinkPanel | 131 | Field 이름과 겹치므로(재선언됨) 변경 바람 |
| | 138 | |
| MainFrame | 162 | Field 이름과 겹치므로(재선언됨) 변경 바람 |
| PaymentDialog | 82-83 | If 문을 두번 사용할 필요가 없으므로 합치기 바람 |
| ReceiveMail | 88 | Field 이름과 겹치므로(재선언됨) 변경 바람 |
| VM | 125 | Field 이름과 겹치므로(재선언됨) 변경 바람 |
| | 216,217 | If 문을 두번 사용할 필요가 없으므로 합치기 바람 |
| | 335,336 | |

**Field명이 겹쳐 재선언 되는 경우,**
**다른 변수인 경우 변수 이름 변경, 같은 변수인 경우 재선언 되지 않도록 변경**

```
public class Code {
    private int code_num;
    private String name;
```

```
    @Override
    public void run(){
        BufferedReader in = null;
        ServerSocket listener = null;
        Socket socket_h = null;
        while(true){
```

```
if (itemName.equals(itemArray[i].getName()) && itemArray[i].getStock() > 0) {
    return true;
}
```

**CardDialog, Code, DrinkPanel, MainFrame, PaymentDialog ,VM Classes에 대하여 위와 같이 수정함.**

# JUnit

## Admin, Card, Code, Drink Class, Item

### ● Admin

- getContact() 함수가 테스팅 되지 않음

```
Assertions.assertEquals(ad.editContact("010-4017-4928"), "010-4017-4928");
Assertions.assertEquals(ad.getContact(), "010-4017-4928");
```

**-> getContact() 를 테스팅 하는 함수 추가**

### ● Card

- getBalance() 함수가 테스팅 되지 않음

```
cd.setBalance(4900);
Assertions.assertEquals(cd.getBalance(), 4900);
```

**-> getBalance() 를 테스팅 하는 함수 추가**

### ● Code, Drink, Item

- 기본 생성자를 사용하지 않는다면 삭제하는 것이 좋음

**-> 해당 내용 삭제**

### ● Item

- editName(String) 함수에서 name 의 길이가 100 보다 긴 예외 경우를 테스트하는 test case 가 없음. 추가하는 것이 좋음

**-> 필요하지 않은 If문이라서 If문 자체를 삭제함**

# JUnit

## VM

### ● VM

- VM(int, double[]) 생성자와 VM(int, int, double[]) 생성자가 실제로도 사용되지 않는 생성자들이라면 삭제하는 것이 좋음

  **-> 해당 내용 삭제**

- VM에서 requestChangeName 등 테스팅이 필요해 보이는 함수들이 테스팅 되지 않음

```
@Test
void checkRequestChangeName(){
    Assertions.assertEquals(vm1.requestChangeName(1,"sprite2"),"1:sprite2");
    Assertions.assertEquals(vm1.requestChangePrice(1,800),"1:800");
}
```

```
@Test
void checkMailReceive(){
    Assertions.assertEquals(vm1.mailRecieve(new Message(9998,9999,5,"10-10-8708")),"mail receive");
    vm1.getOtherVM("Coke");
}
```

**-> 테스팅 되지 않는 함수들의 테스팅 케이스 추가 (소스코드 일부만 첨부)**

- 함수들이 테스팅이 되지만 테스트 케이스가 부족한 경우 발생

```
vm2.mailBox.add(new Message(9999,9998,8,null));
Assertions.assertEquals(vm2.receiveRequest(),8);
```

```
vm1.mailBox.add(new Message(9998,9999,2,"trash"));
Assertions.assertEquals(vm1.receiveRequest(),2);
```

```
void checkRequestPrepay(){
    Assertions.assertEquals(vm3.requestPrepay("Hot Six",9999),"99970000");
}
```

**-> 테스팅 되지 않는 부분들의 테스팅 케이스 추가 (소스코드 일부만 첨부)**

# JUnit

## VM

### ● VM

- VM(int, double[]) 생성자와 VM(int, int, double[]) 생성자가 실제로도 사용되지 않는 생성자들이라면 삭제하는 것이 좋음

  **-> 해당 내용 삭제**

- VM에서 requestChangeName 등 테스팅이 필요해 보이는 함수들이 테스팅 되지 않음

```
@Test
void checkRequestChangeName(){
    Assertions.assertEquals(vm1.requestChangeName(1,"sprite2"),"1:sprite2");
    Assertions.assertEquals(vm1.requestChangePrice(1,800),"1:800");
}
```

```
@Test
void checkMailReceive(){
    Assertions.assertEquals(vm1.mailRecieve(new Message(9998,9999,5,"10-10-8708")),"mail receive");
    vm1.getOtherVM("Coke");
}
```

**-> 테스팅 되지 않는 함수들의 테스팅 케이스 추가 (소스코드 일부만 첨부)**

- 함수들이 테스팅이 되지만 테스트 케이스가 부족한 경우 발생

```
vm2.mailBox.add(new Message(9999,9998,8,null));
Assertions.assertEquals(vm2.receiveRequest(),8);
```

```
vm1.mailBox.add(new Message(9998,9999,2,"trash"));
Assertions.assertEquals(vm1.receiveRequest(),2);
```

```
void checkRequestPrepay(){
    Assertions.assertEquals(vm3.requestPrepay("Hot Six",9999),"99970000");
}
```

**-> 테스팅 되지 않는 부분들의 테스팅 케이스 추가 (소스코드 일부만 첨부)**

# JUnit

## Frame, Panel Classes

### ● UI

- Frame, Panel 클래스들의 경우, Interface Layer 즉 UI 부분들입니다.
- 해당 클래스들의 이미지나 버튼 등이 잘 출력된다는 Junit 테스트를 만들기 애매하여 만들지 않음

### ● 변경 후 Code Coverage

| | | | |
|---|---|---|---|
| VM | 100% (1/1) | 100% (42/... | 98% (282/... |
| Item | 100% (1/1) | 100% (7/7) | 100% (18/... |
| Drink | 100% (1/1) | 100% (5/5) | 100% (10/... |
| Code | 100% (1/1) | 100% (4/4) | 100% (7/7) |
| Card | 100% (1/1) | 100% (6/6) | 100% (19/... |
| Admin | 100% (1/1) | 100% (5/5) | 100% (14/... |
| Message | 100% (1/1) | 100% (7/7) | 100% (22/... |
| RecieveMail | 100% (1/1) | 100% (2/2) | 80% (20/2... |

### ● 그 외의 내용들은 Jira의 해당 이슈에 기록

# Thank you.

Thank you for coming today.