객체지향 개발 방법론

DVM Project OOI Cycle3

#4조

201710240 이해림

201811217 이해인

201711836 송호영

Index

Activity Cycle3. Review & Changes

‘spec’

V&V

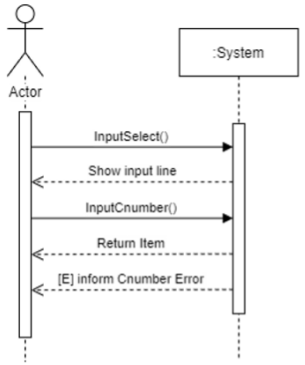
389	수정 사항 누락 - Stage2030_v4 34p use case 4 name

수정 전

Use Case 4. 사용자 인증번호 / 바코드 입력

(A) : Actor (S) : System

1. (A) : 사용자가 DVM에 인증번호 입력 버튼을 선택
2. (S) : 시스템이 이를 인식 후 입력 화면을 출력
3. (A) : 사용자가 이전에 출력 받은 인증번호를 입력
4. (S) : 시스템이 인증 번호에 따른 제품을 전달

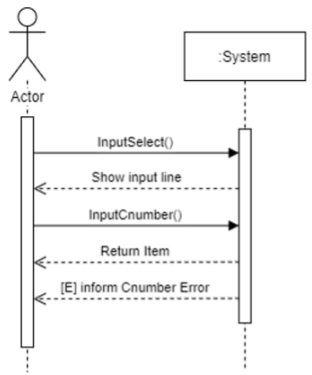


수정 후

Use Case 4. 사용자 인증번호 입력

(A) : Actor (S) : System

1. (A) : 사용자가 DVM에 인증번호 입력 버튼을 선택
2. (S) : 시스템이 이를 인식 후 입력 화면을 출력
3. (A) : 사용자가 이전에 출력 받은 인증번호를 입력
4. (S) : 시스템이 인증 번호에 따른 제품을 전달



Activity Cycle3. Review & Code Change
'PMD'

V&V

1	BeanMembers ShouldSerialize	logic/CNumber.java, 12	member 변수에 transient, static을 붙여주거나 get/set 함수가 필요함.

수정 전

```
public int getTitleId() {
  return titleId;
}

public void setTitleId(int titleId) {
  this.titleId = titleId;
}

public int getDvmID() {
  return DvmID;
}

public void setDvmID(int dvmID) {
  DvmID = dvmID;
}

public int getCNumberT() {
  return cNumberT;
}

public void setCNumberT(int cNumberT) {
  this.cNumberT = cNumberT;
}
```

수정 후

```
public int getTitleId() { return titleId; }

public void setTitleId(int titleId) { this.titleId = titleId; }

public int getDvmID() { return DvmID; }

public void setDvmID(int dvmID) { DvmID = dvmID; }

public int getCNumberT() { return cNumberT; }

public void setCNumberT(int cNumberT) { this.cNumberT = cNumberT; }

public int getcNumberT() { return cNumberT; }

public void setcNumberT(int cNumberT) { this.cNumberT = cNumberT; }

public static Random getRand() { return rand; }

public static void setRand(Random rand) { CNumber.rand = rand; }
```

Activity Cycle3. Review & Code Change
‘PMD’

V&V

2	NullAssignment	logic/Controller.java, 34 logic/Controller.java, 549 logic/Payment.java, 59	변수를 선언하고 있는 상황이 아닐 때, 변수에 null을 할당하는 것은 일반적으로 잘못된 형식임.

수정 전

```
32    public Controller() {
33      this.basket = -666;
34      this.payment = null;
546   public void init() {
547     basket = -666;
548     dvmStack.clear();
549     payment = null;
550   }
56  public void init() {
57    this.titleId = -1;
58    this.DVMId = -1;
59    this.errorLog = null;
60  }
```

수정 후

```
32    public Controller() {
33      this.basket = -666;
```

```
596   public void init() {
597     basket = -666;
598     dvmStack.clear();
599   }
```

```
56    public void init() {
57      this.titleId = -1;
58      this.DVMId = -1;
59      this.errorLog = "";
60    }
```

Activity Cycle3. Review & Code Change
‘PMD’

V&V

3	AvoidLiteralsInIfCondition	logic/Controller.java, 144 logic/Controller.java, 220 logic/Controller.java, 221 logic/Controller.java, 254 logic/Controller.java, 277 logic/Controller.java, 283 logic/Controller.java, 310 logic/Controller.java, 314 logic/Controller.java, 357 logic/Controller.java, 400 logic/Controller.java, 467 logic/Controller.java, 491	조건문에서 하드 코딩된 리터럴을 사용하는 것은 좋지 않음.
		logic/Message.java, 35 logic/Message.java, 38 logic/Message.java, 48 logic/Message.java, 58 logic/Message.java, 69 logic/Message.java, 78 logic/Message.java, 89 logic/MessageQueue.java, 76 logic/MessageQueue.java, 79 logic/MessageQueue.java, 82 logic/MessageQueue.java, 85 logic/MessageQueue.java, 88 logic/MessageQueue.java, 91 logic/MessageQueue.java, 94 logic/MessageQueue.java, 153 logic/MessageQueue.java, 159 logic/MessageQueue.java, 190 logic/MessageQueue.java, 194 logic/MessageQueue.java, 196 logic/MessageQueue.java, 201 logic/MessageQueue.java, 205 logic/MessageQueue.java, 207 logic/MessageQueue.java, 211 logic/Title.java, 21	

수정 후

```
public static void dequeue() {
  while (msgQueue.size() > 0) {
    Message rm = msgQueue.poll();
    if (rm.getType() == 1) {
      Message sm = new Message(DVM.getCurrentID());
      sm.setMsg(rm.getMyId(), type: 2, Controller.getTitleList().get(rm.getTitle() - 1).checkStock());
      //System.out.println("재고 요청 응답 완료");
    } else if (rm.getType() == 2) {
      if(!stkMsgQueue.offer(rm))
        throw new NullPointerException();
    } else if (rm.getType() == 3) {
      CNumber rc = new CNumber(rm.getTitle(), rm.getMyId());
      rc.setCNumberT(rm.getCNumber());
      Controller.getCm().addCNumber(rc);
      Controller.getTitleList().get(rm.getTitle() - 1).updateStock( id: 1, ifHold: true);
    } else if (rm.getType() == 4) {
      Message sm = new Message(DVM.getCurrentID());
      sm.setMsg(rm.getMyId(), type: 5, DVM.getCurrentX(), DVM.getCurrentY());
      //System.out.println("위치 요청 메시지 응답 완료");
    } else if (rm.getType() == 5) {
      if(!locMsgQueue.offer(rm))
        throw new NullPointerException();
    } else if (rm.getType() == 6) {
      Message sm = new Message(DVM.getCurrentID());
      int data = Controller.getCm().checkCNumber(rm.getCNumber());
      sm.setMsg(rm.getMyId(), type: 7, rm.getCNumber(), data2: data != -1);
    } else if (rm.getType() == 7) {
      if(!cNMsgQueue.offer(rm))
        throw new NullPointerException();
    } else {
      System.out.println("메시지 오류");
    }
  }
```

Activity Cycle3. Review & Code Change
‘PMD’

V&V

4	CloseResource	logic/MessageQueue.java, 49 logic/MessageQueue.java, 126	BufferedReader를 사용하고 close를 하지 않았음.
5	CloseResource	logic/MessageQueue.java, 52	ObjectInputStream를 사용하고 close를 하지 않았음.
6	CloseResource	logic/MessageQueue.java, 50 logic/MessageQueue.java, 53 logic/MessageQueue.java, 127	PrintWriter를 사용하고 close를 하지 않았음.

수정 전

```
printWriter.write("1");
printWriter.flush(); //메시지 정상 전송을 클라이언트에게 알려줌
socket.close(); // 소캣을 종료시켜 접속된 클라이언트 종료시킴.
dequeue();
```

```
socket.close();
```

수정 후

```
printWriter.write( s: "1");
printWriter.flush(); //메시지 정상 전송을 클라이언트에게 알려줌
in.close();
printWriter.close();
socket.close(); // 소캣을 종료시켜 접속된 클라이언트 종료시킴.
dequeue();
```

```
socket.close();
in.close();
out.close();
```

Activity Cycle3. Review & Code Change
‘PMD’

V&V

7	DataflowAnomalyAnalysis	logic/Controller.java, 232-235 logic/Controller.java, 235-235 logic/MessageQueue.java, 34-37 logic/MessageQueue.java, 35-38 logic/MessageQueue.java, 47-63 logic/MessageQueue.java, 48-56 logic/MessageQueue.java, 125-133 logic/MessageQueue.java, 218-225 logic/MessageQueue.java, 225-225	(수정 필수 x) 함수 내에서 정의한 변수가 재정의가 된 것은 좋은 것은 아니나 버그까진 아님.

수정 후

```
Socket socket = null;                  //Client와 통신하기 위한 Socket
ServerSocket server_socket = null;  //서버 생성을 위한 ServerSocket
BufferedReader in;                  //Client로부터 데이터를 읽어들이기 위한 입력스트림
int port = myId + 50000;
PrintWriter printWriter; // 값을 전달할때 사용

try {
  server_socket = new ServerSocket(port); //서버 소캣 생성
} catch (IOException e) {
  System.out.println(port + "번 포트 사용 불가");
}
try {
  while (!this.isInterrupted()) {
    assert server_socket != null;
    socket = server_socket.accept();  //서버 오픈 ,클라이언트 접속 대기.
    printWriter = new PrintWriter(
        new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8)));
    in = new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));
    String msg = in.readLine();
    if (msg == null || msg.length() == 0)
      throw new NullPointerException();
    String[] temp = msg.split( regex: ",");
```

Activity Cycle3. Review & Code Change
‘PMD’

V&V

8	DataflowAnomalyAnalysis	logic/Controller.java, 470-505 logic/Controller.java, 489-505 logic/MessageQueue.java, 50-122 logic/MessageQueue.java, 129-174 logic/MessageQueue.java, 187-253	(수정 필수 x) 함수 내에서 정의한 변수가 함수 내에서 사용되지 않았음. 삭제하거나 변수를 사용하는 것을 권장.

수정 전

```
public void msgReceive(int myId) {
  Socket socket = null;                //Client와 통신하기 위한 Socket
  ServerSocket server_socket = null;  //서버 생성을 위한 ServerSocket
  BufferedReader in;             //Client로부터 데이터를 읽어들이기 위한 입력스트림
  PrintWriter out = null;                //Client로 데이터를 내보내기 위한 출력 스트림
  int port = myId + 50000;
  ObjectInputStream objectInputStream; // 직렬화된 객체를 읽어올때 사용
  PrintWriter printWriter; // 값을 전달할때 사용
```

수정 후

```
public void msgReceive(int myId) {
  Socket socket = null;                //Client와 통신하기 위한 Socket
  ServerSocket server_socket = null;  //서버 생성을 위한 ServerSocket
  BufferedReader in;             //Client로부터 데이터를 읽어들이기 위한 입력스트림
  int port = myId + 50000;
  PrintWriter printWriter; // 값을 전달할때 사용
```

Activity Cycle3. Review & Code Change
‘PMD’

V&V

9	NonStaticInitializer	logic/CNumber.java, 14	초기화 블록이 생성자가 생성될 때마다 실행됨. 틀린 구문은 아니지만 거의 사용 되지 않고 혼란을 줄 수 있음. static으로 수정하거나 삭제하는 것을 권장.

수정 전

```
12   private Random rand;
13
14   {
15     try {
16       new SecureRandom();
17       rand = SecureRandom.getInstanceStrong();
18     } catch (NoSuchAlgorithmException e) {
19       throw new RuntimeException("Failed to instantiate random number generator", e);
20     }
21   }
```

수정 후

```
11   private static Random rand = new SecureRandom();
12
13   /*
14   {
15     try {
16       new SecureRandom();
17       rand = SecureRandom.getInstanceStrong();
18     } catch (NoSuchAlgorithmException e) {
19       throw new RuntimeException("Failed to instantiate random number generator", e);
20     }
21   }
22   */
```

Activity Cycle3. Review & Code Change

‘Spotbugs’

V&V

4,5	RV_RETURN_VALUE_IGNORED_BAD_PRACTICE: Method ignores exceptional return value	MessageQueue.java :[line 75, 195, 206, 212]	offer함수의 return값을 검사하여 오류가 발생하였는지 확인이 필요하다.

수정 전

```
75          msgQueue.offer(message);
195         stkMsgQueue.offer(rm);
206         locMsgQueue.offer(rm);
212         cNMsgQueue.offer(rm);
```

수정 후

```
61          if (msg == null || msg.length() == 0)
62            throw new NullPointerException();
```

```
176         if(!stkMsgQueue.offer(rm))
177           throw new NullPointerException();
```

```
188         if(!locMsgQueue.offer(rm))
189           throw new NullPointerException();
```

```
195         if(!cNMsgQueue.offer(rm))
196           throw new NullPointerException();
```

Activity Cycle3. Review & Code Change
‘Spotbugs’

V&V

- Internationalization Warnings

Index	Warning	Code	Warning 발생 이유
6~9	DM_DEFAULT_ENCODING: Reliance on default encoding	MessageQueue.java :[line 65, 66, 135, 136]	생성자 인자에 인코딩 방식을 추가로 전달해야 시스템 종속적인 인코딩이 시행되지 않는다. `new InputStreamReader(socket.getInputStream(),StandardCharsets.UTF_8)`

수정 전

```
printWriter = new PrintWriter(
    new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())));
in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
```

```
in = new BufferedReader(
    new InputStreamReader(socket.getInputStream())); //서버로부터 메시지를 받기위한 버퍼
out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())));
```

수정 후

```
printWriter = new PrintWriter(
    new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8)));
in = new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));
```

```
in = new BufferedReader(
    new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8)); //서버로부터 메시지를 받기위한 버퍼
out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8)))
```

Activity Cycle3. Review & Code Change
‘Spotbugs’

V&V

10	SBSC_USE_STRINGBUFFER_CONCATENATION: Method concatenates strings using + in a loop	CNumber.java :[line 70]	Loop안에서 String concatenate를 할 경우에는 StringBuffer을 사용하여 append하고 Loop 종료 후에 String으로 변환하는 것이 좋다. 다만 Loop를 3번만 도는 것으로 확인되어 이를 꼭 수정할 필요는 없을 것으로 생각된다.

수정 후

```
public String randNumber() {
  int len = 6;
  String numStr = ""; //난수가 저장될 변수
  String ran = Integer.toString( i: rand.nextInt( bound: 9) + 1);  //첫번째 숫자 0이 아님
  numStr += ran;
  numStr += Integer.toString( i: DVM.getCurrentID() - 1); // 두번째 자릿수 => DVMID
  numStr += Integer.toString( i: this.DvmID - 1);
  for (int i = 3; i < len; i++) {
    //0~9 까지 난수 생성
    ran = Integer.toString(rand.nextInt( bound: 10));
    numStr += ran;
  }
  return numStr;
}
```

Activity Cycle3. Review & Code Change
‘Spotbugs’

V&V

11	SIC_INNER_SHOULD_BE_STATIC_ANON: Could be refactored into a named static inner class	Title.java :[line 48]	Outer Class의 reference를 이용하지 않아 익명 inner class보다 static inner class를 따로 선언하여 사용하는 것이 좋다. 하지만 IntelliJ에서 제안하는대로 `Comparator.comparingInt(Item::getExpirationDate)` 로 코드를 바꾸는 것이 더 좋아보인다.

수정 전

```
public void addItem(Item item) {
  this.itemList.add(item);
  this.itemList.sort(new Comparator<Item>() {
    @Override
    public int compare(Item o1, Item o2) {
      return o1.getExpirationDate() - o2.getExpirationDate();
    }
  });
}
```

수정 후

```
public void addItem(Item item) {
  this.itemList.add(item);
  this.itemList.sort(Comparator.comparingInt(Item::getExpirationDate));
}
```

Activity Cycle3. Review & Code Change

‘Spotbugs’

V&V

12~ 30	BC_UNCONFIRMED_CAST: Unchecked/unconfirmed cast	Controller.java: [line 339, 349, 169, 520, 133, 91, 254,278,271,258, 228,245,264 214,284, 185, 444, 478, 460, 73, 427, 528, 301, 391, 381]	JFrame객체를 gui의 각 Class로 cast할 때 castable한지 확인하는 부분이 존재하지 않아 발생한 경고이다.

수정 전

```
del = ((MainMenu) k).getReturnValue();

del = ((InputLine) k).getReturnValue();

del = ((InputLine) k).getReturnValue();

del2 = ((InfoUI) k).getReturnValue();
```

수정 후

```
if (k instanceof MainMenu) {
  del = ((MainMenu) k).getReturnValue();
}

if (k instanceof Sleep) {
  del = ((Sleep) k).getReturnValue();
}

if (k instanceof  InputLine) {
  del = ((InputLine) k).getReturnValue();
}

if (k instanceof InfoUI) {
  del2 = ((InfoUI) k).getReturnValue();
}
```

Activity Cycle3. Review & Code Change
‘Spotbugs’

V&V

31, 32	NP_DEREFERENCE_OF_READLINE_VALUE: Dereference of the result of readLine() without nullcheck	MessageQueue.java :[line 68, 147]	readLine()의 결과에 대해 null checking이 필요하다.

수정 전

```
out.println(msg);                          //서버로 데이터 전송
out.flush();                          //서버로 데이터 전송
String returnMsg = in.readLine();
//객체 정리하는 부분
socket.close();
//서버에서 확인메시지 리시브 및 완료시 브레이크
if (returnMsg.equals("1")) {
  break;
}
```

수정 후

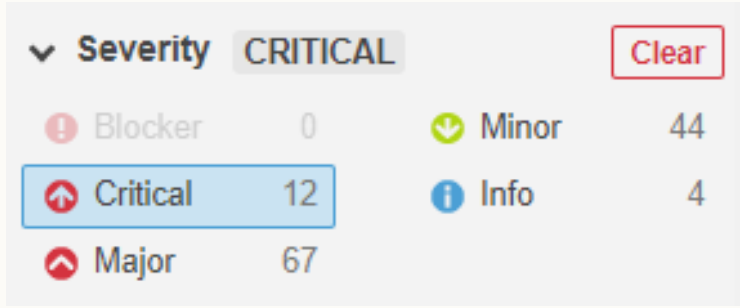
```
out.println(msg);                          //서버로 데이터 전송
out.flush();                          //서버로 데이터 전송
String returnMsg = in.readLine();
if (returnMsg == null || returnMsg.length() == 0)
  throw new NullPointerException();
//객체 정리하는 부분
socket.close();
in.close();
out.close();
//서버에서 확인메시지 리시브 및 완료시 브레이크
if (returnMsg.equals("1")) {
  break;
}
```

Activity Cycle3. Review & Code Change

‘SonarQube’

수정 전

수정 후



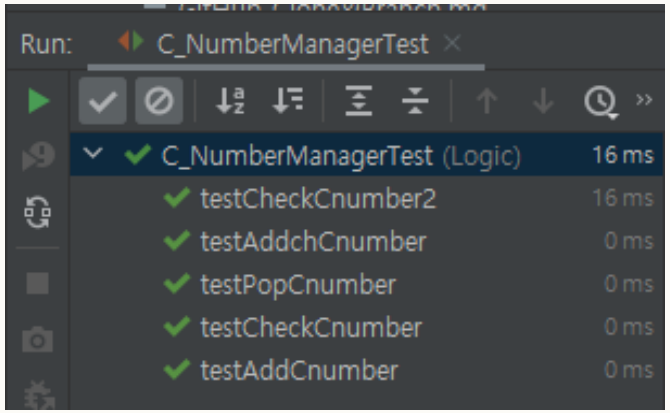
Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: C_NumberManager

```
@Test
public void testAddCNumber() {
  Cn.setCNumberT(971125);
  CM.addCNumber(Cn);
  int ExpectedResult = 1;
  int ActualResult = CM.getCList().size();
  Assert.assertEquals(ExpectedResult, ActualResult);

@Test
public void testAddChCNumber() {
  Cn.setCNumberT(971125);
  CM.addChCNumber(Cn);
  int ExpectedResult = 1;
  int ActualResult = CM.getChCList().size();
  Assert.assertEquals(ExpectedResult, ActualResult);
}
@Test
public void testGetMNumber() {
  CM.setMNumber(123456);
  int ExpectedResult = 123456;
  int ActualResult = CM.getMNumber();
  Assert.assertEquals(ExpectedResult, ActualResult);
}
@Test
public void testGetCList() {
  CNumber cnumber = new CNumber( titleId: 1,  id: 1);
  cnumber.setcNumberT(111111);
  HashMap<Integer, CNumber> cList = new HashMap<>();
  cList.put(1,cnumber);
  CM.setCList(cList);
  Assert.assertEquals(cList,CM.getCList());
}

@Test
public void testGetChCList() {
  CNumber cnumber = new CNumber( titleId: 1,  id: 1);
  cnumber.setcNumberT(111111);
  HashMap<Integer, CNumber> chList = new HashMap<>();
  chList.put(1, cnumber);
  CM.setChCList(chList);
  Assert.assertEquals(chList, CM.getChCList());
}
}
```

결 과



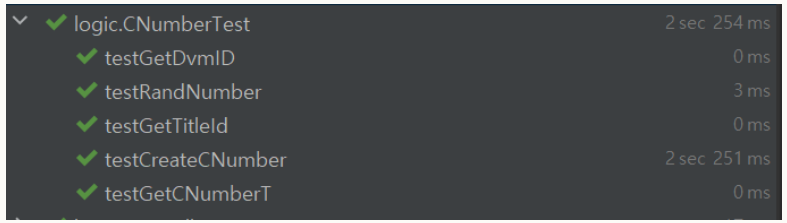
Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: C_Number

```java
@Test
public void testGetTitleId() {
  CN.setTitleId(4);
  int ExpectedResult = 4;
  int ActualResult = CN.getTitleId();
  Assert.assertEquals(ExpectedResult, ActualResult);
}

@Test
public void testGetDvmID() {
  CN.setDvmID(3);
  int ExpectedResult = 3;
  int ActualResult = CN.getDvmID();
  Assert.assertEquals(ExpectedResult, ActualResult);
}

@Test
public void testGetCNumberT() {
  CN.setCNumberT(123456);
  int ExpectedResult = 123456;
  int ActualResult = CN.getCNumberT();
  Assert.assertEquals(ExpectedResult, ActualResult);
}
```

결 과



Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: Controller

```
@Test
public void testGetDel() {
  c.setDel(0);
  Assert.assertEquals( expected: 0, c.getDel());
}

@Test
public void testGetTitleList() {
  ArrayList<Title> temp = Controller.getTitleList();
  ArrayList<Title> t = new ArrayList<>();
  t.add(new Title( name: "A", price: 700));
  Controller.setTitleList(t);
  Assert.assertEquals( expected: 1, Controller.getTitleList().size());
  Assert.assertEquals( expected: "A", Controller.getTitleList().get(0).getName());
  Assert.assertEquals( expected: 700, Controller.getTitleList().get(0).getPrice());
  Controller.setTitleList(temp);
}

@Test
public void testGetCm() {
  CNumberManager temp = Controller.getCm();
  CNumberManager cm = new CNumberManager();
  cm.setMNumber(971125);
  Controller.setCm(cm);
  Assert.assertNotNull(Controller.getCm());
  Assert.assertEquals( expected: 971125, Controller.getCm().getMNumber());
  Assert.assertEquals(cm, Controller.getCm());
  Controller.setCm(temp);
}

@Test
public void testGetMq() {
  MessageQueue temp = Controller.getMq();
  MessageQueue mq = new MessageQueue();
  Controller.setMq(mq);
  Assert.assertEquals(mq, Controller.getMq());
  Controller.setMq(temp);
}
```

결 과

✔ logic.ControllerTest	17 ms
✔ testGetCm	4 ms
✔ testGetMq	0 ms
✔ testInit	1 ms
✔ testGetTitleList	0 ms
✔ testReturnMain	12 ms
✔ testCancelPay	0 ms
✔ testCancelItem	0 ms
✔ testGetDel	0 ms

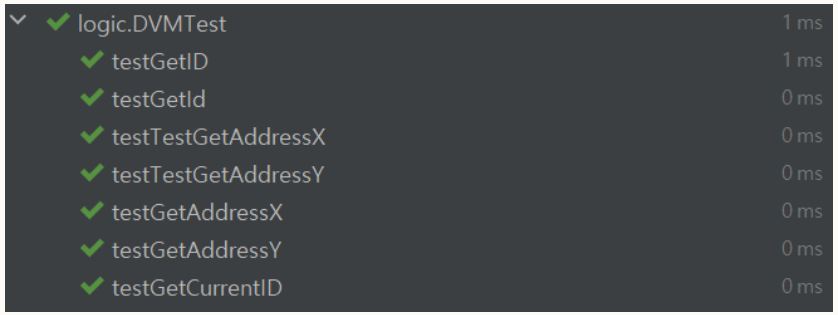
Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: DVM

```
@Test
public void testGetID() {
  dvm.setId(7);
  int ExpectedResult = 7;
  int ActualResult = dvm.getId();
  Assert.assertEquals(ExpectedResult, ActualResult);
}


@Test
public void testGetAddressX() {
  dvm.setAddressX(1.23456);
  Double ExpectedResult = 1.23456;
  Double ActualResult = dvm.getAddressX();
  Assert.assertEquals(ExpectedResult, ActualResult);
}

@Test
public void testGetAddressY() {
  dvm.setAddressY(2.45678);
  Double ExpectedResult = 2.45678;
  Double ActualResult = dvm.getAddressY();
  Assert.assertEquals(ExpectedResult, ActualResult);
}

@Test
public void testGetCurrentID() {
  DVM.setCurrentID(1);
  Assert.assertEquals( expected: 1, DVM.getCurrentID());
}
@Test
public void testGetId() {
  DVM.setCurrentID(1);
  Assert.assertEquals( expected: 1,DVM.getCurrentID());
}
@Test
public void testTestGetAddressX() {
  DVM.setCurrentX(1.0);
  Assert.assertEquals(Double.toString( d: 1.0),Double.toString(DVM.getCurrentX()));
}

@Test
public void testTestGetAddressY() {
  DVM.setCurrentY(1.0);
  Assert.assertEquals(Double.toString( d: 1.0),Double.toString(DVM.getCurrentY()));
}
```

결 과



Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: MessageQueue

```
@Test
public void testMsgReceive() {
  Thread thread = new Thread(() -> queue.msgReceive( myId: 1));
  MessageQueue.getStkMsgQueue().clear();
  thread.start();
  msg.setMsg( id: 1, type: 2,  data: true);
  while(MessageQueue.getStkMsgQueue().isEmpty()){}
  thread.interrupt();
  Message rm = MessageQueue.getStkMsgQueue().poll();
  Assert.assertEquals(msg.getTargetId(), rm.getTargetId());
  Assert.assertEquals(msg.getType(), rm.getType());
  Assert.assertEquals(msg.isBoolData(), rm.isBoolData());
  Assert.assertEquals(msg.getCNumber(), rm.getCNumber());
  Assert.assertEquals(msg.getMyId(), rm.getMyId());
  Assert.assertEquals(msg.getTitle(), rm.getTitle());
  Assert.assertEquals(Double.toString(msg.getXAddress()), Double.toString(rm.getXAddress()));
  Assert.assertEquals(Double.toString(msg.getYAddress()), Double.toString(rm.getYAddress()));
}

@Test
public void testMsgSend() {
  Controller.getTitleList().get(0).addItem(new Item( expirationDate: 20201125));
  queue.start();
  msg.setTargetId(1);
  msg.setType(3);
  msg.setTitle(1);
  msg.setCNumber(971125);
  MessageQueue.msgSend(msg);
  queue.interrupt();
  while(queue.isAlive()){}
  Assert.assertEquals( expected: 1, Controller.getCm().checkCNumber(msg.getCNumber()));
  Assert.assertEquals( expected: false, Controller.getTitleList().get(msg.getTitle()-1).checkStock());
  Assert.assertEquals( expected: 1, (int)(Controller.getTitleList().get(msg.getTitle()-1).getHold()));
  Assert.assertEquals(msg.getTitle(), Controller.getCm().popCNumber(msg.getCNumber()));
}
```

결 과

✔ logic.MessageQueueTest	13 ms
✔ testGetIP	3 ms
✔ testGetStkMsgQueue	1 ms
✔ testGetCNum	0 ms
✔ testGetMsgQueue	0 ms
✔ testDequeue	9 ms
✔ testGetLoc	0 ms
✔ testGetStk	0 ms
✔ testGetLocMsgQueue	0 ms

Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: Message

```
@Test
public void testSetMsg1() {
  msg.setMsg( id: 2,  type: 1,  data: 1);
  Message testMsg = new Message(DVM.getCurrentID());
  Assert.assertEquals(testMsg.getMyId(),msg.getMyId());
  testMsg.setTargetId(2);
  Assert.assertEquals(testMsg.getTargetId(),msg.getTargetId());
  testMsg.setType(1);
  Assert.assertEquals(testMsg.getType(),msg.getType());
  testMsg.setXAddress(-1);
  Assert.assertEquals(testMsg.getXAddress(),msg.getXAddress(), delta: 1);
  testMsg.setYAddress(-1);
  Assert.assertEquals(testMsg.getYAddress(),msg.getYAddress(), delta: 1);
  testMsg.setTitle(1);
  Assert.assertEquals(testMsg.getTitle(),msg.getTitle());
  testMsg.setCNumber(-1);
  Assert.assertEquals(testMsg.getCNumber(),msg.getCNumber());
  testMsg.setBoolData(false);
  Assert.assertEquals(testMsg.isBoolData(),msg.isBoolData());
}
```

결 과

Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

Class: Payment

```
@Test
public void testGetTitleId() {
  payment.setTitleId(4);
  int ExpectedResult = 4;
  int ActualResult = payment.getTitleId();
  Assert.assertEquals(ExpectedResult, ActualResult);
}

@Test
public void testGetDvmId() {
  payment.setDVMId(3);
  int ExpectedResult = 3;
  int ActualResult = payment.getDVMId();
  Assert.assertEquals(ExpectedResult, ActualResult);
}

@Test
public void testGetErrorLog() {
  payment.setErrorLog("hello");
  String ExpectedResult = "hello";
  String ActualResult = payment.getErrorLog();
  Assert.assertEquals(ExpectedResult, ActualResult);
}
```

결 과

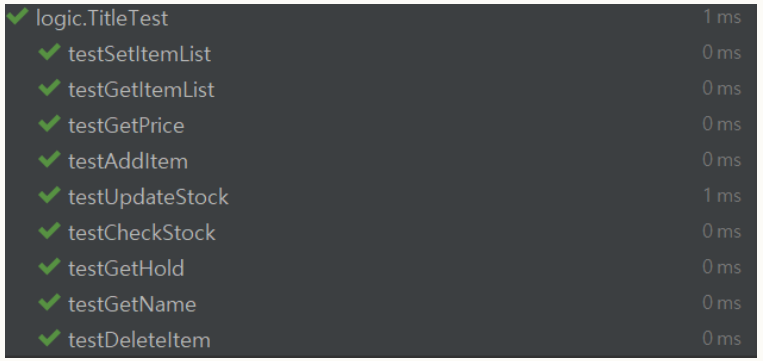
Activity 2055. Write Unit Test Code & Unit Testing

추가된 메소드에 대한 Unit Test Code 작성

Class:Title

```
@Test
public void testCheckStock() {
  t.addItem(new Item( expirationDate: 20201025));
  Assert.assertTrue(t.checkStock());
  t.updateStock( id: 0,  ifHold: false);
  Assert.assertFalse(t.checkStock());
}
@Test
public void testGetName() {
  t.setName("코카콜라");
  Assert.assertEquals( expected: "코카콜라",t.getName());
}

@Test
public void testGetPrice() {
  t.setPrice(1000);
  Assert.assertEquals( expected: 1000,t.getPrice());
}

@Test
public void testSetItemList() {
  ArrayList<Item> i = new ArrayList();
  t.setItemList(i);
  Assert.assertEquals(i, t.getItemList());
}

@Test
public void testGetHold() {
  t.setHold(1);
  Assert.assertEquals( expected: 1, t.getHold());
}
```

결 과



Activity 2055. Write Unit Test Code & Unit Testing

Unit Test Code 추가

결 과

Package logic

all > logic

63	0	0	21.185s
tests	failures	ignored	duration

100% successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
CNumberManagerTest	8	0	0	0.007s	100%
CNumberTest	5	0	0	2.190s	100%
ControllerTest	8	0	0	0.013s	100%
DVMTest	7	0	0	0.003s	100%
ItemTest	2	0	0	0s	100%
MessageQueueTest	10	0	0	0.520s	100%
MessageTest	8	0	0	14.346s	100%
PaymentTest	6	0	0	4.105s	100%
TitleTest	9	0	0	0.001s	100%

Activity 2055. Write Unit Test Code & Unit Testing

Test Coverage

수정 전

logic

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Controller		13%		0%	127	136	358	398	37	46	0	1
MessageQueue		24%		25%	44	56	136	185	12	19	0	1
Message		81%		57%	11	31	17	85	5	24	0	1
Title		81%		85%	8	20	11	39	6	13	0	1
CNumber		93%		71%	6	16	6	48	2	9	0	1
Payment		89%		100%	3	16	6	41	3	10	0	1
DVM		81%		n/a	4	14	6	26	4	14	0	1
CNumberManager		91%		100%	2	14	4	27	2	12	0	1
Item		100%		n/a	0	3	0	6	0	3	0	1
Title.new Comparator() {...}		100%		n/a	0	2	0	2	0	2	0	1
Total	1,973 of 3,109	36%	247 of 312	20%	205	308	544	856	71	152	0	10

수정 후

logic

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Controller		14%		0%	141	160	367	424	27	46	0	1
MessageQueue		90%		75%	19	59	23	179	0	21	0	1
Message		100%		57%	6	31	0	85	0	24	0	1
CNumber		100%		71%	4	19	0	47	0	12	0	1
Title		100%		92%	1	20	0	39	0	13	0	1
Payment		100%		100%	0	16	0	41	0	10	0	1
CNumberManager		100%		100%	0	14	0	27	0	12	0	1
DVM		100%		n/a	0	14	0	26	0	14	0	1
Item		100%		n/a	0	3	0	6	0	3	0	1
Total	1,477 of 3,162	53%	258 of 362	28%	171	336	390	874	27	155	0	9

Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R1.1	1. 자판기 음료 종류 출력	- 20가지 메뉴가 정상적으로 출력되는지 확인	**P**
R1.2.1	2. 구매할 음료 입력	- 사용자가 선택한 음료가 정확히 기록되는지 확인	**P**
R.1.2.2	3. 사용자 선택 취소	- 사용자 선택 취소 시 메인 화면으로 돌아가는 지 확인	**P**
R1.2.3	4. 사용자 인증번호/바코드 입력	- 유효하지 않은 인증번호가 입력된 경우 오류 메시지를 출력하는지 확인	**P**
		- 올바른 인증번호가 입력된 경우 보관된 제품 중 인증번호에 맞는 제품을 전달하는지 확인	**P**
R1.3	5. 재고가 부족한 제품 안내	- '재고가 부족한 제품' 이라는 안내문구를 정확히 출력하는지 확인	**P**
R1.4	6. 구매 가능한 자판기 안내	- 정확한 위치정보를 출력하는지 확인	**P**
		- 지정된 물품의 재고가 정확히 존재하는지 확인	**P**
		- 지정된 물품의 재고가 존재하는 자판기가 모두 출력되는지 확인	**P**

Activity 2063. System Testing

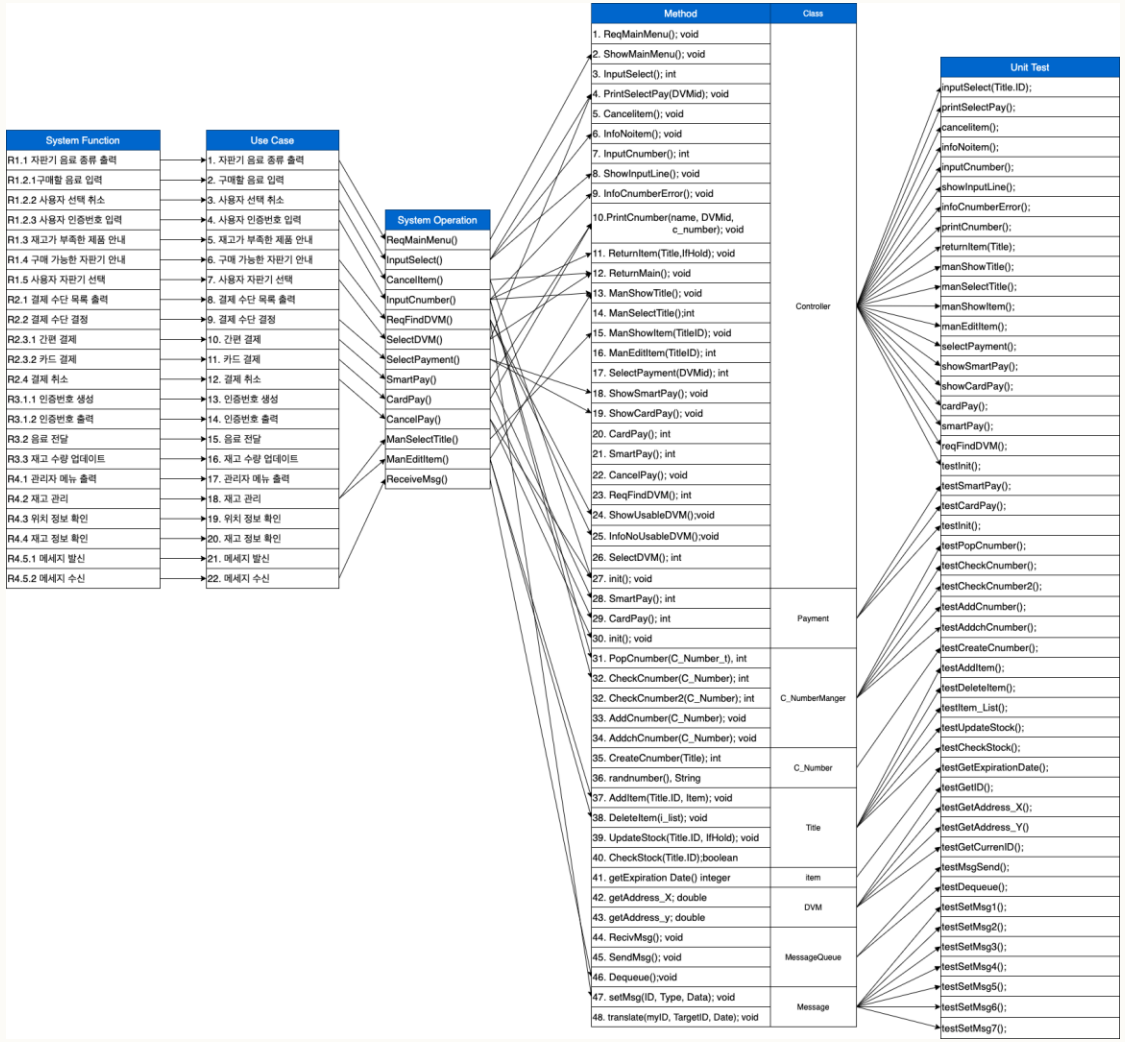
Ref	Use Case name	Test Description	P/F
R1.5	7. 사용자 자판기 선택	- 사용자가 선택한 DVM의 id가 정확히 기록되는 지 확인	**P**
R2.1	8. 결제 수단 목록 출력	- 카드결제, 간편결제 버튼이 모두 출력되는지 확인. - 사용자가 선택한 제품 이름과 가격이 정상적으로 출력되는지 확인	**P**
R2.2	9. 결제 수단 결정	- 사용자가 선택한 결제 수단에 따른 결제 화면으로 넘어가는지 확인	**P**
R2.3.1	10. 간편 결제	- 입력된 정보에 따라 정상적으로 성공/실패 여부가 출력되는지 확인	**P**
R2.3.2	11. 카드 결제		
R2.4	12. 결제 취소	- 결제 취소 시 메인 화면으로 돌아가는 지 확인	**P**
R3.1.1	13. 인증번호/바코드 생성	- 인증번호가 중복되지 않게 생성되는지 확인	**P**
R3.1.2	14. 인증번호/바코드 출력	- 생성된 인증번호가 정상적으로 출력되는지 확인	**P**

Activity 2063. System Testing

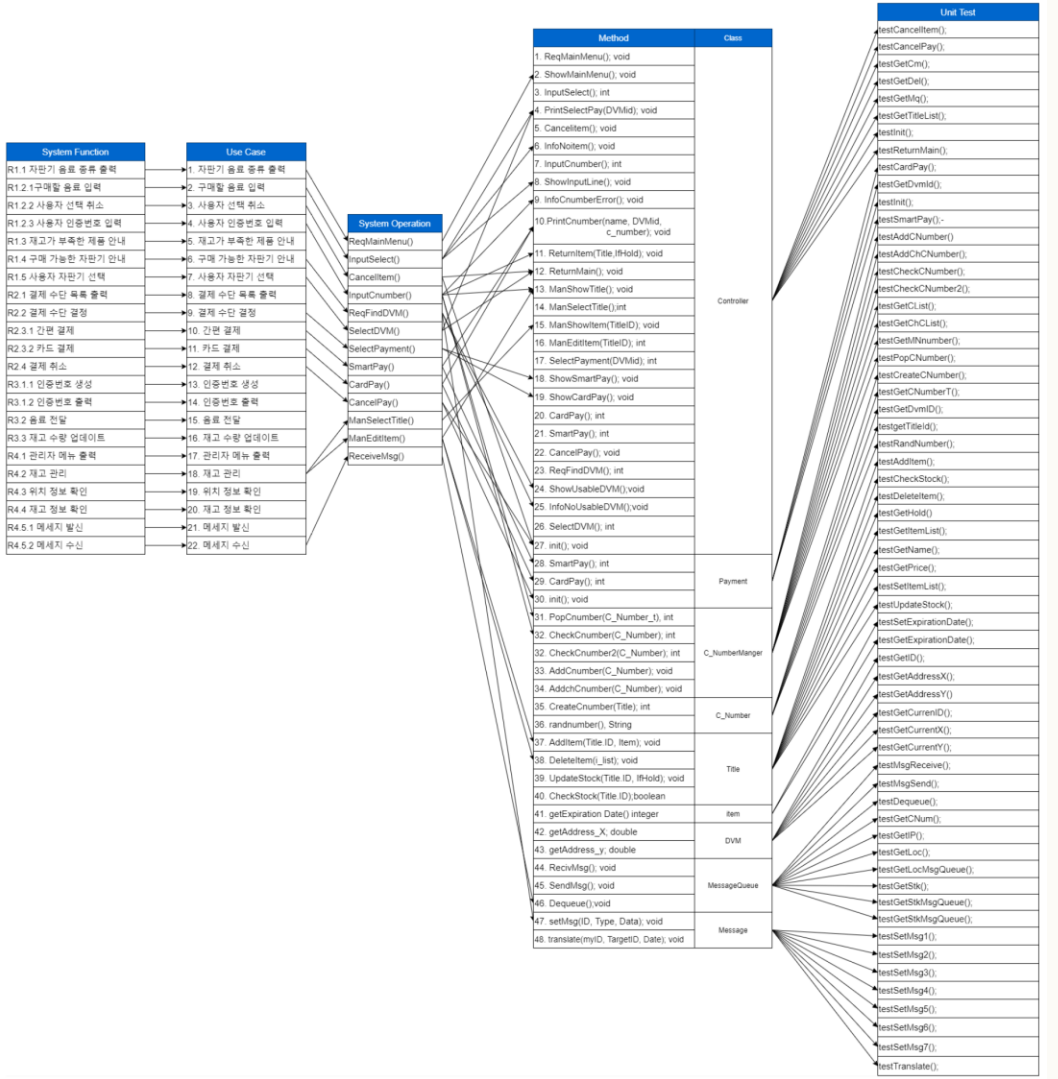
Ref	Use Case name	Test Description	P/F
R3.2	15. 음료 전달	- 사용자가 결제를 성공한 제품이 제대로 전달되는지 확인	**P**
R3.3	16. 재고 수량 업데이트	- 사용자 이용 이후 변화된 재고가 정확히 변화하는지 확인	**P**
R4.1	17. 관리자 메뉴 출력	- 인증번호 입력화면에서 관리자 고유 번호를 입력함에 따라 메뉴가 출력 되는지 확인	**P**
R4.2	18. 재고 변경/관리	- 관리자의 동작에 따라 재고의 수량이 잘 변경되는지 확인	**P**
R4.3	19. 위치 정보 확인	- 정확한 위치 정보를 반환하는지 확인	**P**
R4.4	20. 재고 정보 확인	- 정확한 재고 정보를 반환하는지 확인	**P**
R4.5.1	21. 메시지 발신	- DVM네트워크를 통한 메시지가 정확히 목표 DVM으로 전송되는지 확인	**P**
R4.5.2	22. 메시지 수신	- DVM네트워크를 통한 메시지가 정확히 수신되는지 확인	**P**

Testing Traceability Analysis

수정 전

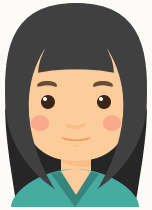


수정 후



4학년이 제공한 CTIP 환경에 대한 소감

깃헙을 이용한 협업은 처음 사용해 보았는데, 각자 짠 코드를 합치고 다시 공유하는 과정을 효율적으로 진행할 수 있어 좋았습니다. 또한 다양한 검증 툴이 바로 연동되어 코드를 추가할 때마다 발생한 문제점들을 즉각적으로 확인할 수 있어 효율적인 팀 활동이 가능하였습니다. 여러 툴을 통해 확인되는 이슈들을 해결하며 기능을 제대로 수행하는 것 외에 프로그램에서 중요한 부분들이 어떤 것인지 새로운 시각을 가질 수 있게 되었습니다

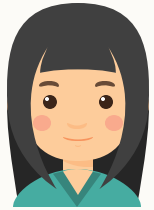


팀원 모두 깃헙 사용이 처음이라 상당한 어려움을 겪었습니다. SonarQube의 경우 퀄리티 게이트를 통해 이전 버전과 비교하여 코드를 분석하는데, 이에 따라 풀리퀘스트에서 버그가 파악되지 않는 경우가 있었습니다. 또한 이슈트래킹 툴로 지라를 사용하였는데, 이슈가 여러 단계로 링크되어 있어 피드백을 작성하는데 있어 약간의 불편함이 있었습니다.

어디까지가 4학년 팀이 제공해줄 수 있는 영역인지 파악하기 어려운데다, 비대면으로만 협업이 진행되어서 팀 간의 소통에 약간의 주저함이 있었던 것 같습니다. 다음에 또 이러한 프로젝트를 진행하게 된다면, 보다 적극적으로 소통에 임하는 것이 중요하다고 생각합니다.

UP (OOPT)를 기반으로 SW 개발한 방식에 대한 소감

개발 시 코드를 작성한 후 보고서를 작성하는 것에 익숙했는데 문서작업을 통해 필요한 기능을 꼼꼼히 생각해보고 SW를 설계하는 과정을 통해 프로그램에 대한 높은 이해도를 가지고 개발 단계에 돌입할 수 있어 좋았습니다. 또한 현업에서 실제로 사용하는 방법론을 사용하여 프로젝트를 진행하였다는 점이 뜻깊은 경험이 되었다고 생각합니다.



다만 수정사항이 발생할 경우 처음부터 문서작업을 전부 확인해가며 수정해야 한다는 점이 큰 부담으로 다가왔습니다. 각 스테이지가 독립적이지 않은 만큼 작은 수정사항이 발생해도 처음부터 리뷰 해야하는 부분은 비효율적으로 느껴지기도 하였습니다.

이번 팀 프로젝트에서는 객체 지향 개발과 도구 사용 등의 이해도가 높지 않아 팀프로젝트 초반에 약간의 차질이 있었습니다. 다음에 또 UP기반으로 소프트웨어를 개발한다면 전체 스테이지에 대한 충분한 이해를 바탕으로 프로젝트를 진행할 수 있을 것이라 생각합니다.