

소프트웨어 설계 A조

카카오톡 선물하기
기능 구현

설계 : D조

구현 : A조

A조 조원 : 정금종, 강유빈, 손호빈

목 차

01 | Intro

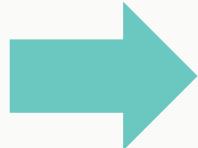
02 | 코드 설명

03 | 오류 수정 및 추가

04 | 구현시 어려웠던
점

05 | Q & A

01 | Intro



구현



카카오톡 선물하기

- 선물함
- 교환
- 환불
- 상품 목록
- 상세 보기
- 결제

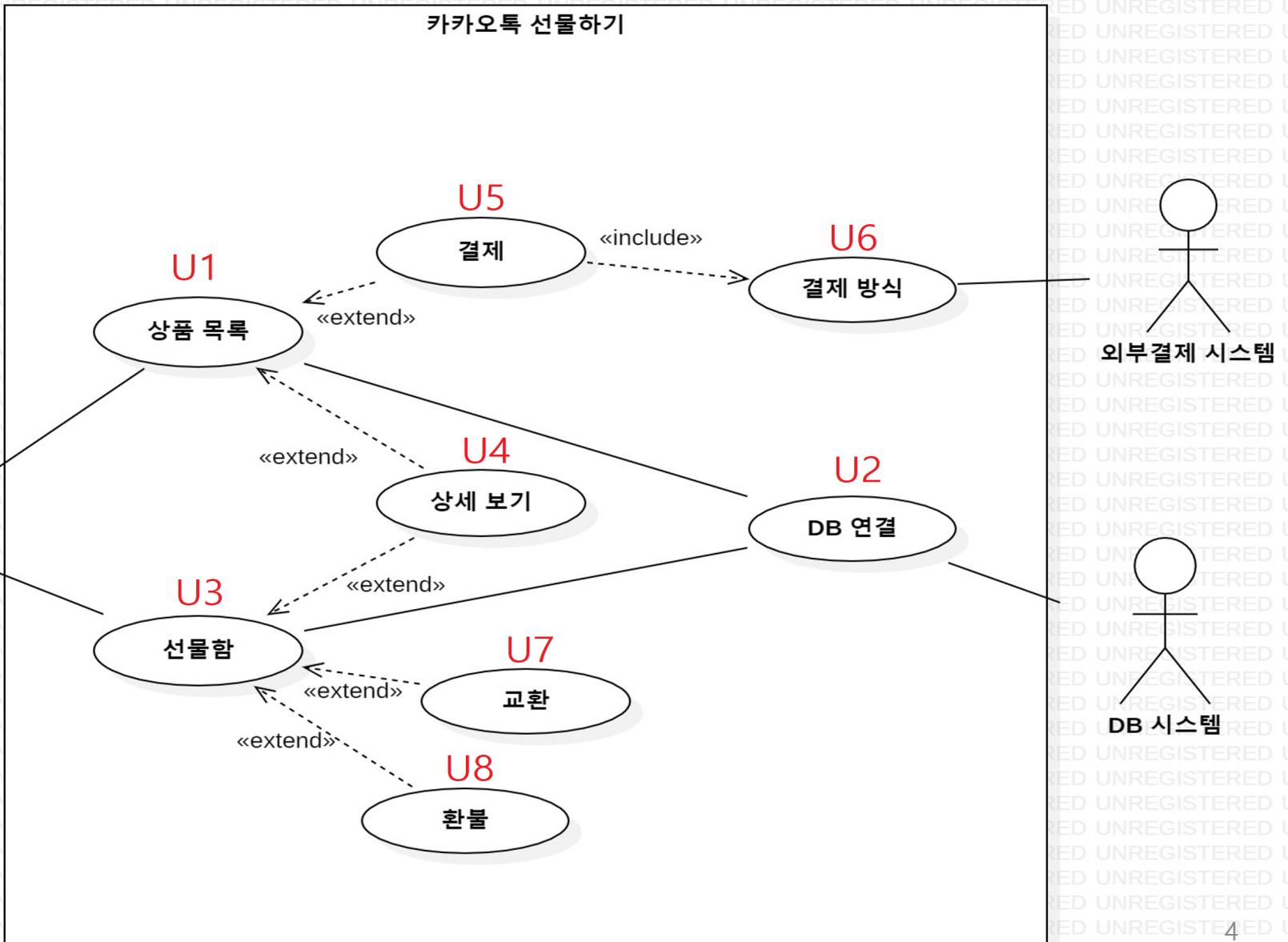
개발 언어 : JAVA , mongoDB

개발 환경 : 이클립스

추가 라이브러리 : JSON-SIMPLE

02 | 코드 설명

-유스케이스



02 | 코드 설명

-클래스 다이어그램

1. 상품 리스트

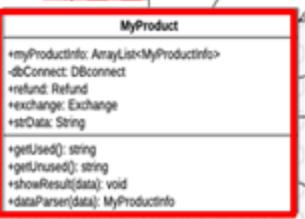


메인

4. 세부정보 보여주기



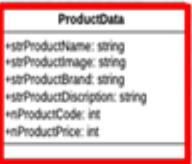
3. 내선율



9. Struct



9. Struct



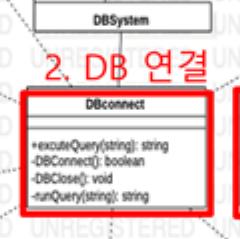
9. Struct



5. 결제

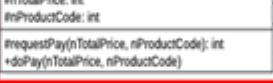


2. DB 연결



7. 교환

6. 결제수단



8. 환불



CreditCardPay

MobilePay

KakaoPay

Creditcard Payment Sys

Mobile Payment Sys

Kakao Payment Sys

02 | 코드 설명

-ProductList Diagram

C0. Main - main()

```
public class Main { //class main
    public static void main() throws ParseException{ // add : main
        try (Scanner menu = new Scanner(System.in)) {
            int click = menu.nextInt();

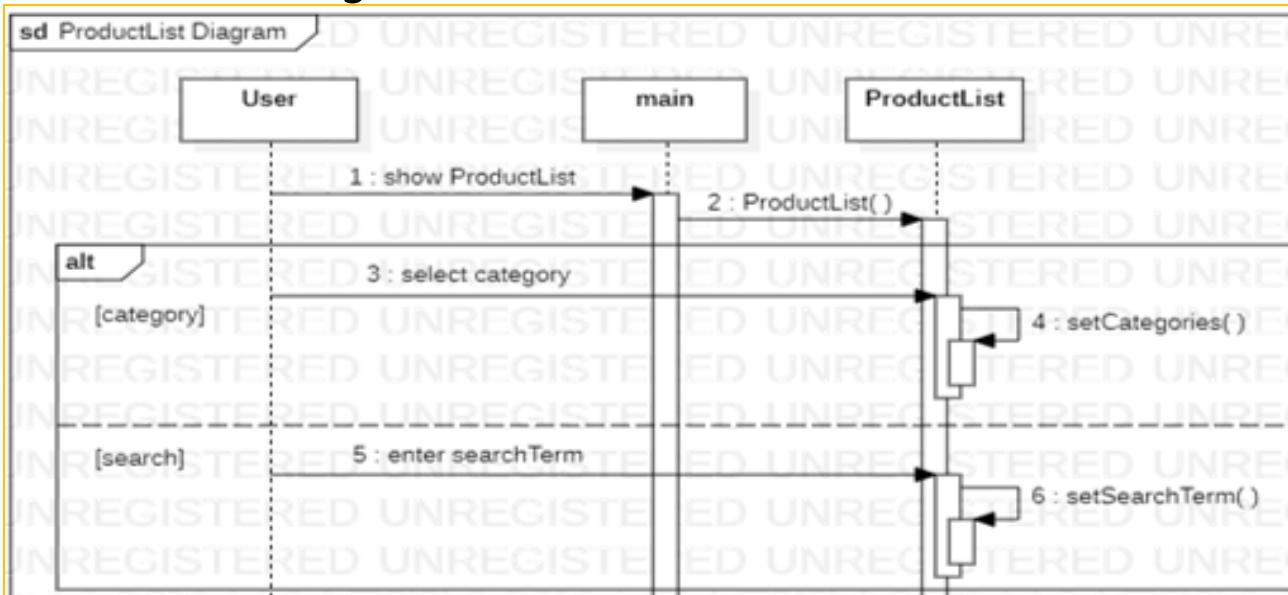
            1, 2 if(click == 1) { //U1.SeqDiagram : 1
                ProductList(); //U1.SeqDiagram : 1
            }
        }
    }
}
```

*main() 메서드 추가

C0. Main - ProductList()

```
public static void ProductList() throws ParseException{ //U1.SeqDiagram : 1
    ProductList productList = new ProductList(); //U1.SeqDiagram : 2
}
```

U1. ProductList Diagram : 1 ~ 6



[1]

사용자가
상품 목록 요청

[2]

ProductList() 실행

C0.main에서 ProductList 객체
생성

02 | 코드 설명

-ProductList Diagram

C0. Main - ProductList()

```
if(click == "Category"){
    productList.setCategories();
}
else if(click == "searchTerm"){
    productList.setSearchTerm();
}
```

3, 4 //U1.SeqDiagram : 3
//U1.SeqDiagram : 4

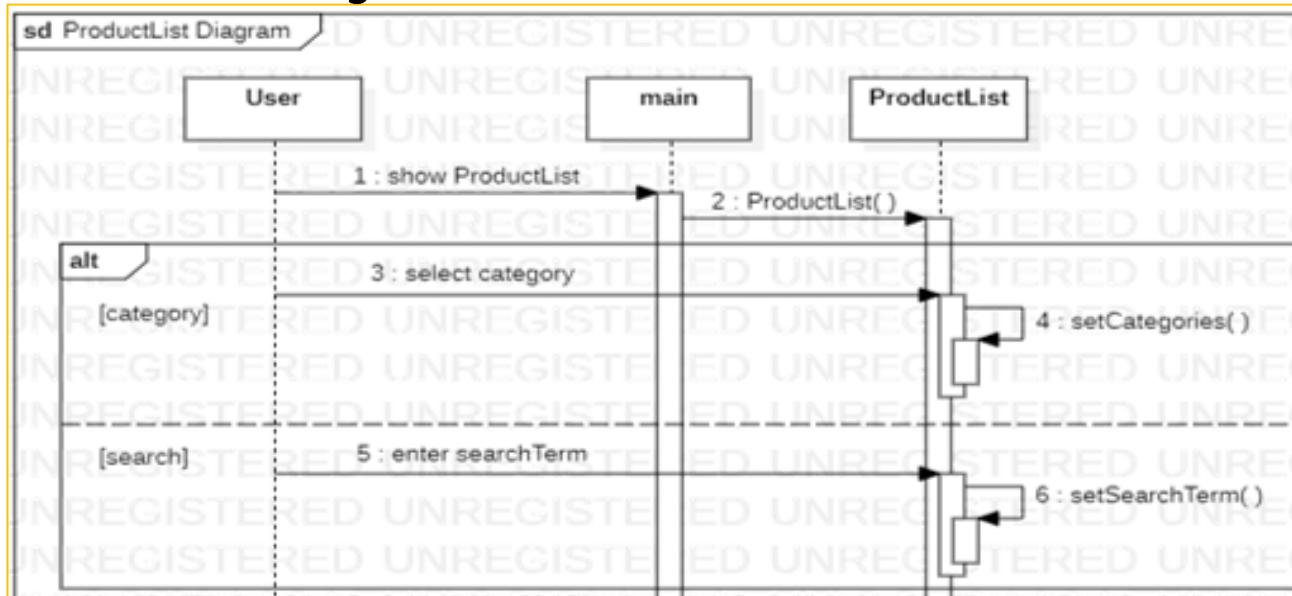
5, 6 //U1.SeqDiagram : 5
//U1.SeqDiagram : 6

[3, 4]

사용자가 카테고리를 선택

setCategories()를 통해
변수 초기화

U1. ProductList Diagram : 1 ~ 6



[5, 6]

사용자가 검색어를 선택

setSearchTerm()을 통해
변수 초기화

02 | 코드 설명

-ProductList Diagram

C1. ProductList

```
535~ public void setCategories() { //U1.SeqDiagram : 4, private -> public
536     try (Scanner scanner = new Scanner(System.in)) {
537         System.out.print("input category ");
538         this.Categories = scanner.nextLine();
539     }
540 }
```

4

* 두 메서드의 접근 지정자 변경
private -> public

```
553~ public void setSearchTerm() { //U1.SeqDiagram : 6, private -> public
554     try (Scanner scanner = new Scanner(System.in)) {
555         System.out.print("input searchTerm");
556         this.SearchTerm = scanner.nextLine();
557     }
558 }
```

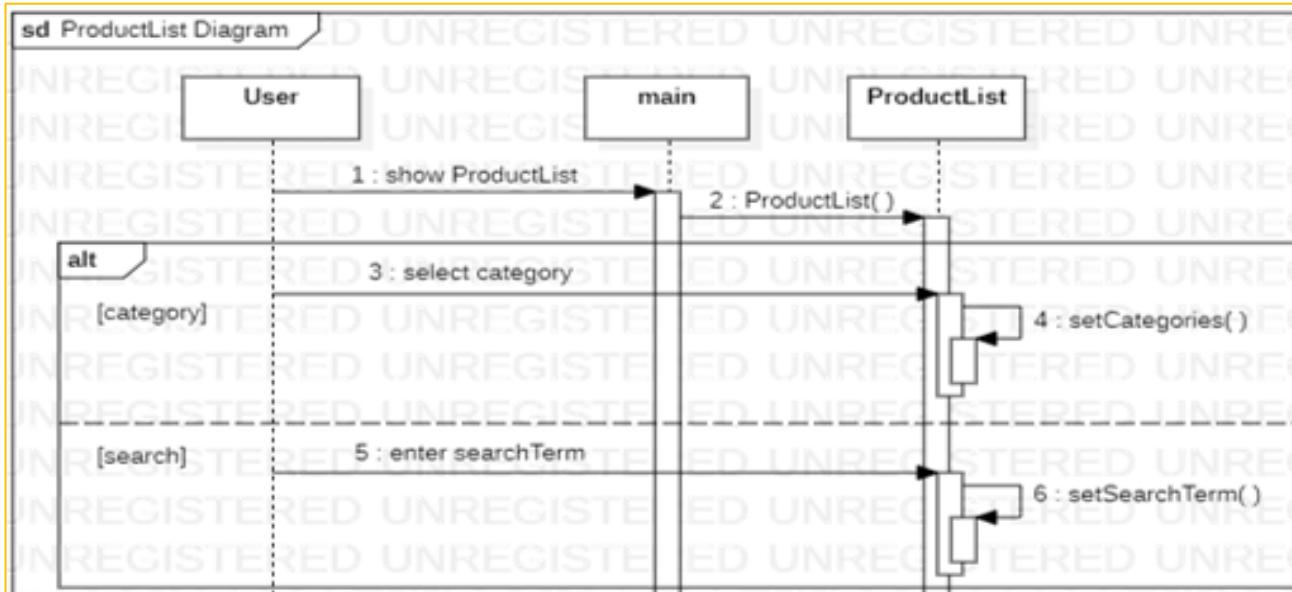
6

[3, 4]

사용자가 카테고리를 선택

setCategories()를 통해
변수 초기화

U1. ProductList Diagram : 1 ~ 6



[5, 6]

사용자가 검색어를 선택

setSearchTerm()을 통해
변수 초기화

02 | 코드 설명

-ProductList Diagram

C0.Main – ProductList()

```

productList.strData = productList.getProductList(); //U1.SeqDiagram : 7~9 , jump to C1.ProductList
if(productList.strData.contains("error")) { //U1.SeqDiagram : 10
    System.out.println("error page"); //U1.SeqDiagram : 10
}
else {
    productList.dataIndexing(productList.dataParser(productList.strData)); //U1.SeqDiagram : 11
}

```

7
10
11

* dataIndexing() 추가

C1.ProductList

```

public String getProductList() { //U1.SeqDiagram : 7
    String strGetDataQuery = " strGetDataQuery ";
    return this.dbConnect.executeQuery(strGetDataQuery); //U1.SeqDiagram : 8, jump to class main
}

```

8, 9

[7]

getProductList() 호출
DB에서 상품 목록 조회

[8, 9]

쿼리를 실행하고
문자열을 반환
*JSON 문자열

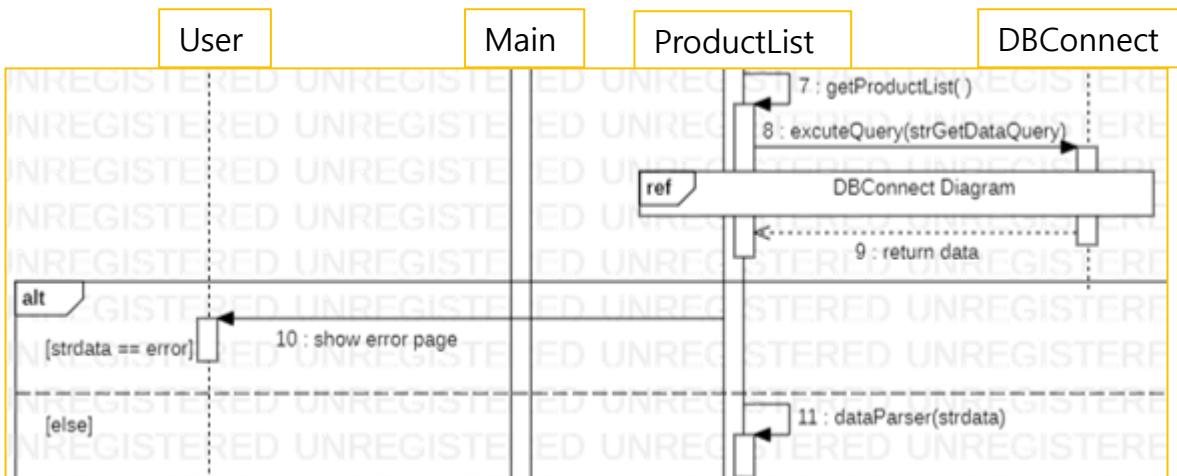
[10]

strData에 에러 문자열이 저
장되었다면 에러 페이지 출력

[11]

정상적인 경우, JSON 문자열
을 ArrayList<ProductData>
로 변환

U1. ProductList Diagram : 7 ~ 11



C1. ProductList – dataParser(String data)

```

public ArrayList<ProductData> dataParser(String data){ //U1.SeqDiagram :11 , private->public
    ArrayList<ProductData> parsedData = new ArrayList<>();
    ProductData tmpData = new ProductData();

    /*Parse from JSON String*/

    try {
        JSONParser jParse = new JSONParser();
        JSONObject pObj = (JSONObject)jParse.parse(data);
        JSONArray jArray = (JSONArray)pObj.get("ProductDataList");

        for(int jArrIndex = 0 ; jArrIndex < jArray.size();jArrIndex++) {
            JSONObject productData = (JSONObject)jArray.get(jArrIndex);

            tmpData.strProductName = (String)productData.get("strProductName");
            tmpData.strProductImage = (String)productData.get("strProductImage");
            tmpData.strProductBrand = (String)productData.get("strProductBrand");
            tmpData.strProductDescription = (String)productData.get("strProductDescription");
            tmpData.nProductCode = (Integer)productData.get("nProductCode");
            tmpData.nProductPrice = (Integer)productData.get("nProductPrice");

            parsedData.add(tmpData);
        }
    } catch(ParseException e) { e.printStackTrace(); }
    tmpData = null;
    return parsedData;
}

```

```

public void dataIndexing(ArrayList<ProductData> productDataList) {
    ArrayList<ProductData> parsedList = new ArrayList<>();
    int arrProductListIndex = 0;

    parsedList = this.dataParser(this.strData); //U3.SeqDiagram
    for(arrProductListIndex = 0; arrProductListIndex < parsedList.size(); arrProductListIndex++) {
        this.productDataList.add(parsedList.get(arrProductListIndex));
    }
    parsedList = null;
}

```

* 접근 지정자 변경
dataParser()
private -> public

1.

매개변수 String data로
JSON 문자열을 수신

2.

변환 정보를 저장할 객체 생성
parsedData

3.

JSON 문자열을 JSON 객체로
변환

4.

JSON 객체 배열의 정보를
parsedData에 복사 후 반환

5.

dataIndexing()를 통해
productDataList에 저장

02 | 코드 설명

-ProductList Diagram

C0.Main – ProductList()

```

else {
    productList.dataIndexing(productList.dataParser(productList.strData)); //U1.SeqDiagram : 11
    click = menu.nextLine();
    if(click == "PriceRange") { //U1.SeqDiagram : 12
        productList.setPriceRange(); //U1.SeqDiagram : 13
    }
    else if(click == "setSortingType") { //U1.SeqDiagram : 14
        productList.setSortingType(); //U1.SeqDiagram : 15
    }
}

```

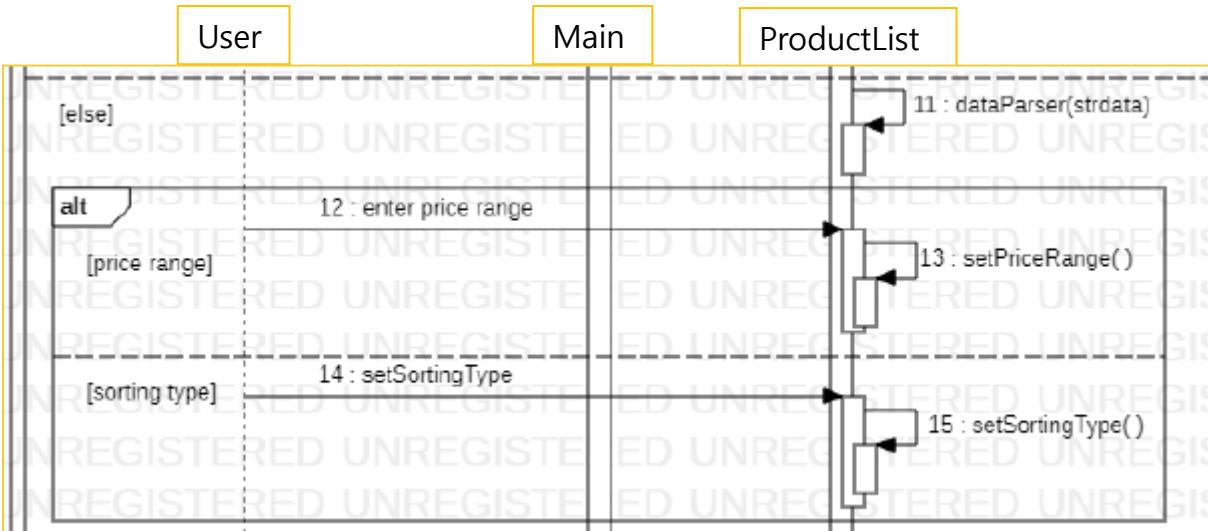
12, 13
14, 15

- * ProductDataList 정렬 기준 : 가격
- * 유스케이스 명세서 정렬 기준 : 추천, 가격, 할인율, 최근 등록

[12, 13]

사용자가 가격대를 입력하면 setPriceRange()로 변수 초기화, 리스트 초기화

U1. ProductList Diagram : 12 ~ 15



[14, 15]

사용자가 정렬기준을 선택하면 setSortingType()로 변수 초기화, 리스트 초기화

C1.ProductList – setPriceRange()

```

public void setPriceRange() { //U1.SeqDiagram : 13, private -> public
    try (Scanner scanner = new Scanner(System.in)) {
        System.out.print("input price range (0, 1, 2, 3)");
        this.nPriceRange = scanner.nextInt();
    }
    if(this.nPriceRange == 0) {// 1 ~ 9,999
        for(int nReIndex = 0 ; nReIndex < this.productDataList.size(); nReIndex++) {
            if(this.productDataList.get(nReIndex).nProductPrice > 0 && this.productDataList.get(nReIndex).nProductPrice < 10000) {
                //print
            }
        }
    }
    else if(this.nPriceRange == 1) { // 10,000 ~ 29,999
        for(int nReIndex = 0 ; nReIndex < this.productDataList.size(); nReIndex++) {
            if(this.productDataList.get(nReIndex).nProductPrice >= 10000 && this.productDataList.get(nReIndex).nProductPrice < 30000) {
                //print
            }
        }
    }
    else if(this.nPriceRange == 2) { // 30,000 ~ 49,999
        for(int nReIndex = 0 ; nReIndex < this.productDataList.size(); nReIndex++) {
            if(this.productDataList.get(nReIndex).nProductPrice >= 30000 && this.productDataList.get(nReIndex).nProductPrice < 50000) {
                //print
            }
        }
    }
    else if(this.nPriceRange == 3) { // 50,000 ~
        for(int nReIndex = 0 ; nReIndex < this.productDataList.size(); nReIndex++) {
            if(this.productDataList.get(nReIndex).nProductPrice > 50000) {
                //print
            }
        }
    }
}

```

*접근 지정자 변경
private -> public

1.

1 만원 미만 상품 출력

2.

1-2만원 대 상품 출력

3.

3-4만원 대 상품 출력

4.

5만원 이상 상품 출력

13

C1.ProductList – setSortingType()

```

public void setSortingType() { //U1.SeqDiagram : 15, private -> public
    try (Scanner scanner = new Scanner(System.in)) {
        System.out.print("input sortingType : asc, desc");
        this.strSortInfo = scanner.nextLine();
    }
    if(this.strSortInfo == "asc") {
        ArrayList<Integer> ASC = new ArrayList<>();
        ArrayList<Integer> nIndex = new ArrayList<>();
        int nTmp, nTmpIdx;
        int nListIndex, nSubListIndex;

        for(nListIndex = 0 ; nListIndex < this.productDataList.size(); nListIndex++) {
            ASC.add(this.productDataList.get(nListIndex).nProductPrice);
            nIndex.add(nListIndex);
        }
        for(nListIndex = 0 ; nListIndex < this.productDataList.size(); nListIndex++) {
            for(int subListIndex = 0 ; nListIndex < this.productDataList.size()-nListIndex-1;subListIndex++) {
                if(ASC.get(subListIndex) > ASC.get(subListIndex+1)) {
                    nTmp = ASC.get(subListIndex);
                    nTmpIdx = nIndex.get(subListIndex);
                    ASC.set(subListIndex, ASC.get(subListIndex+1));
                    nIndex.set(subListIndex, nIndex.get(subListIndex)+1);
                    ASC.set(subListIndex+1, nTmp);
                    nIndex.set(subListIndex+1, nTmpIdx);
                }
            }
        } ASC = null;
        for(nListIndex = 0 ; nListIndex < this.productDataList.size();nListIndex++) {
            //print this.productDataList.get(nIndex.get(nListIndex))
        } nIndex = null;
    }
    else if(this.strSortInfo == "desc") {

```

*접근 지정자 변경
private -> public

1.

가격과 리스트 인덱스로
오름차순 또는 내림차순
버블 정렬 실행

2.

정렬된 리스트 인덱스 순으로
출력

15

C0.Main – ProductList()

```

productList.showProductList(productList.productDataList); 16, 17
//U1.SeqDiagram : 16, jump to C1.ProductList
if(productList.nProduct != 0) { //U1.SeqDiagram : 18
    productList.showProductDetail(productList.productDataList.get(productList.nProduct));
    //U1.SeqDiagram : 18,
} //jump to (U4) C4.ShowMyProductDetail
}

```

*showProductList()
클래스 디어그램에
표기 X

18

[16, 17]

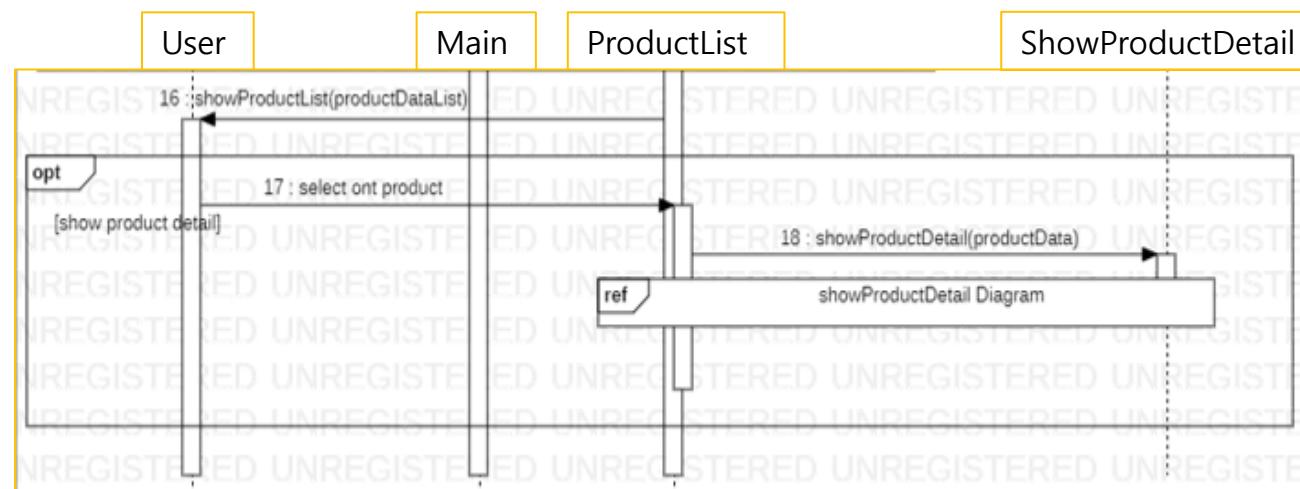
showProductList()로
상품 목록 출력
[16]의 내부에서 사용자가 상
품 선택

[18]

C4-1.ShowProductDetail의
showProductDetail()메서드
호출

U4-1.showProductDetail
Diagram으로 이동

U1. ProductList Diagram : 16 ~ 18



02 | 코드 설명

-DBConnect Diagram

C2.DBConnect – executeQuery(String query)

```
class DBConnect{ //class 2
    public String executeQuery(String query) { //U2.SeqDiagram : 1
        boolean bConnectResult = this.DBconnect(); //U2.SeqDiagram : 2

        if(bConnectResult == false) { //U2.SeqDiagram : 3
            return "ERROR"; //U2.SeqDiagram : 4
        }
    }
}
```

1, 2
3, 4

*클래스와 메서드 명칭
이 동일
-> 생성자 경고
-> 이름 수정

[1]

쿼리 전달

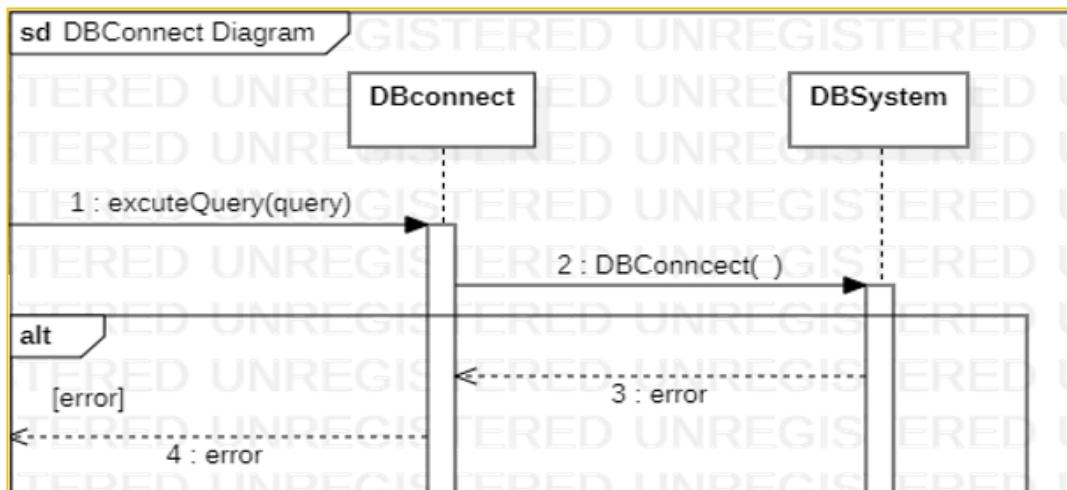
[2]

연결 실패 시 false 반환

[3, 4]

[2]의 결과로 false가 반환되면
에러 문자열 반환

U2. DBConnect Diagram : 1 ~ 4



02 | 코드 설명

-DBConnect Diagram

C2.DBConnect – executeQuery(String query)

```

else {
    String data = runQuery(query); //U2.SeqDiagram : 5, 6
    this.DBclose(); //U2.SeqDiagram : 7
    return data; //U2.SeqDiagram : 8
}
}                                5~8

```

*유스케이스 명세서
->반환 형태 : 리스트

[5, 6]

쿼리 실행으로 전달받은
JSON문자열을 data에 저장

```

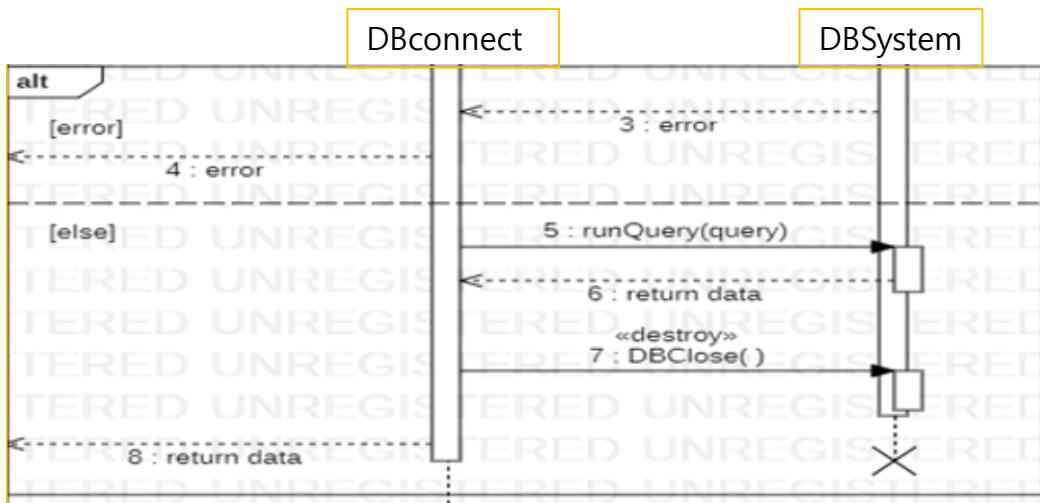
private String runQuery(String query) { //U2.SeqDiagram : 5
    return "__JSON_String__";
}
}                                5

```

[7, 8]

DB 연결 종료 후 문자열 반환

U2. DBConnect Diagram : 5 ~ 8



02 | 코드 설명 - My Product Diagram

C0.Main – main()

```
public class Main { //class main
    public static void main() throws ParseException{
        try (Scanner menu = new Scanner(System.in)) {
            int click = menu.nextInt();

            if(click == 1) {    //U1.SeqDiagram : 1
                ProductList(); //U1.SeqDiagram : 1
            }
            else {           //U3.SeqDiagram : 1
                MyProduct(); //U3.SeqDiagram : 1
            }
        } } 1, 2
```

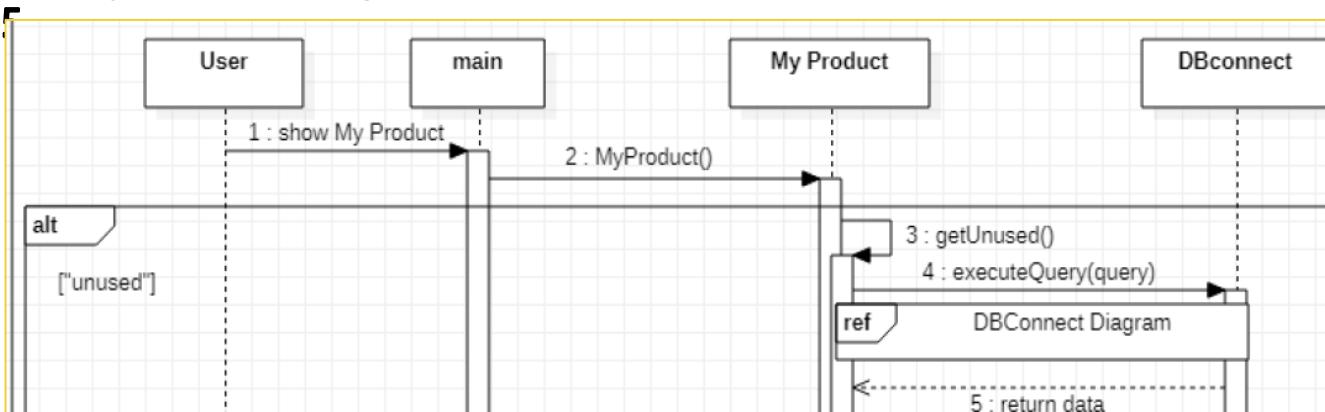
[1]

사용자가 선물함 클릭

[2]

MyProduct() 를 통해
선물함으로 이동

U3. My Product Diagram : 1 ~



02 | 코드 설명 - My Product Diagram

C0.Main – MyProduct()

```
if(click == "UNUSED") { //U3.SeqDiagram : 3  
    myProduct.strData = myProduct.getUnused(); //U3.SeqDiagram : 3, jump to C3.MyProduct  
    if(myProduct.strData.contains("ERROR")) { //U3.SeqDiagram : 6  
        myProduct.showErrorResult(myProduct.strData); //U3.SeqDiagram : 6  
    }  
    else { //U3.SeqDiagram : 7  
        myProduct.dataIndexing((ArrayList<MyProductInfo>)myProduct.dataParser(myProduct.strData)); //U3.SeqDiagram : 7  
        myProduct.showResult(myProduct.myProductInfo); //U3.SeqDiagram : 8  
    }  
}
```

3

[3]

사용 가능 선물 선택(미사용)
getUnused() 실행

C3.MyProduct – getUnused()

```
public String getUnused() { //U3.SeqDiagram : 3  
    return this.dbConnect.executeQuery("query"); //U3.SeqDiagram : 4, 5, jump to Class main  
}
```

4, 5

[4]

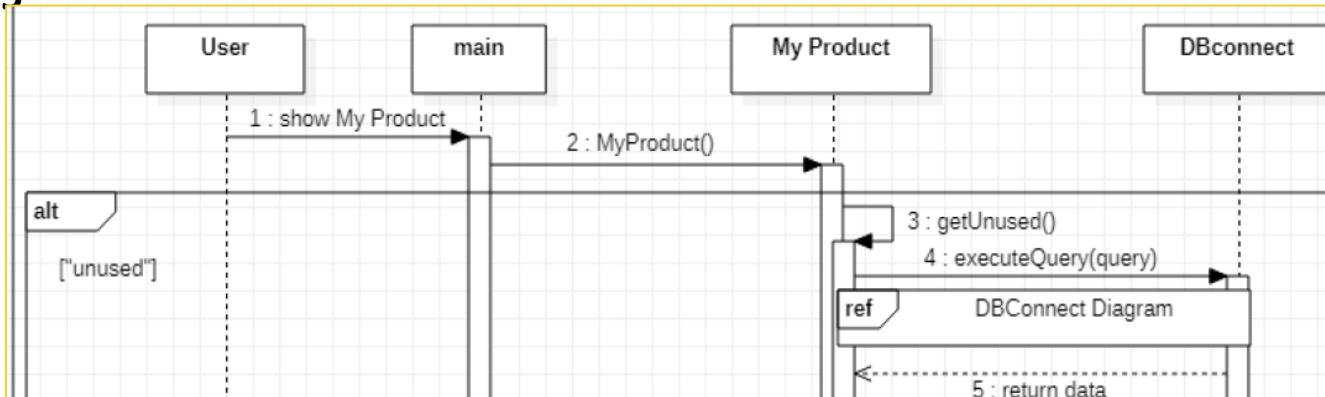
미사용 선물 조회 쿼리 실행

[5]

쿼리의 결과를 반환하여
strData에 저장

U3. My Product Diagram : 1 ~

5



02 | 코드 설명 - My Product Diagram

C0.Main – MyProduct()

```
if(click == "UNUSED") {      //U3.SeqDiagram : 3
    myProduct.strData = myProduct.getUnused(); //U3.SeqDiagram : 3, jump to C3.MyProduct
    if(myProduct.strData.contains("ERROR")) { //U3.SeqDiagram : 6
        myProduct.showErrorResult(myProduct.strData); 6 //U3.SeqDiagram : 6
    }
} else { //U3.SeqDiagram : 7
    myProduct.dataIndexing((ArrayList<MyProductInfo>)myProduct.dataParser(myProduct.strData));
    //U3.SeqDiagram : 7
    myProduct.showResult(myProduct.myProductInfo); //U3.SeqDiagram : 8
}
}
```

*showErrorResult()추가
6, 8 같은 메서드 사용
->인자가 다르고
->오버로딩 표기X

[6]

에러를 보여줌

[7]

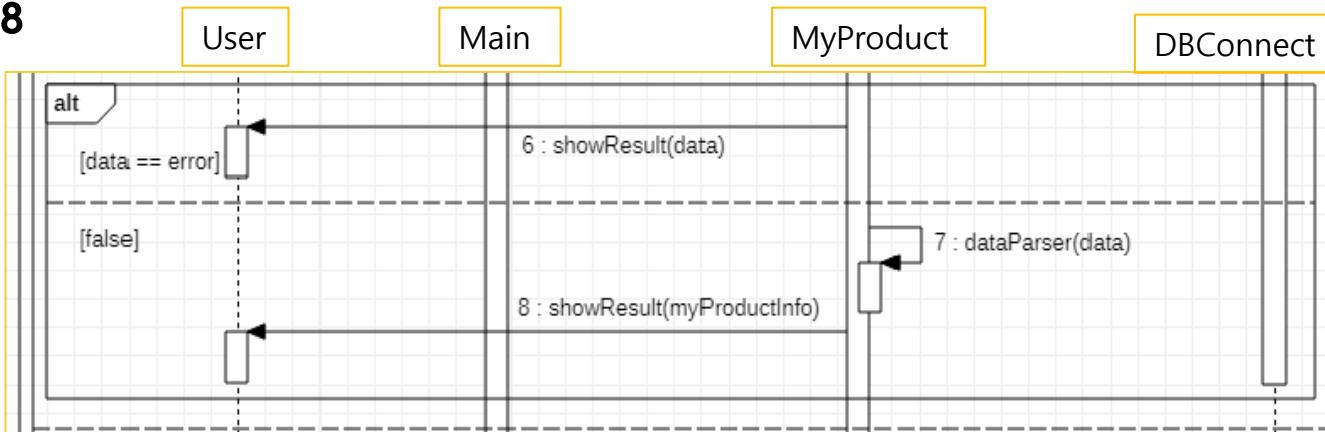
[5]로 전달받은 JSON 문자열
을
변환하여 저장

[8]

ArrayList의 myProductInfo를
보여줌

U3. My Product Diagram : 6 ~

8



02 | 코드 설명 - My Product Diagram

C3.MyProduct – dataParser(String data), dataIndexing(ArrayList)

```
public ArrayList<MyProductInfo> dataParser(String strData) { //U3.SeqDiagram 7, 13
    ArrayList<MyProductInfo> parsedData = new ArrayList<>();
    MyProductInfo tmpData = new MyProductInfo();

    /*Parse from JSON String*/
    try {
        JSONParser jParse = new JSONParser();
        JSONObject jObj = (JSONObject)jParse.parse(strData);
        JSONArray jArray = (JSONArray)jObj.get("myProducts");

        for(int jArrIndex = 0 ; jArrIndex < jArray.size();jArrIndex++) {
            JSONObject pInfo = (JSONObject)jArray.get(jArrIndex);
            tmpData.strReceiver = (String)pInfo.get("strReceiver");
            tmpData.strSender = (String)pInfo.get("strSender");
            tmpData.nPaymentCode = (Integer)pInfo.get("nPaymentCode");
            tmpData.nExpiredDate = (Integer)pInfo.get("nExpiredDate");
            tmpData.isUsed = (Boolean)pInfo.get("isUsed");
            parsedData.add(tmpData);
        }
    } catch(ParseException e) { e.printStackTrace(); }
    tmpData = null;
    return parsedData;
}

public void dataIndexing(ArrayList<MyProductInfo> myProductList) { //U3.SeqDiagram : 7
    //add : for init data
    ArrayList<MyProductInfo> parsedList = new ArrayList<>();
    int arrMyProductListIndex = 0;

    parsedList = this.dataParser(this.strData); //U3.SeqDiagram
    for(arrMyProductListIndex = 0; arrMyProductListIndex < parsedList.size(); arrMyProductListIndex++) {
        this.myProductInfo.add(parsedList.get(arrMyProductListIndex)); //U3.SeqDiagram
    }
    parsedList = null;
}
```

* dataParser 반환형 변경
MyProductInfo ->
ArrayList<MyProductInfo>

1.

매개변수 String data로
JSON 문자열을 수신

2.

변환 정보를 저장할 객체 생성
parsedData

3.

JSON 문자열을 JSON 객체로
변환

4.

JSON 객체 배열의 정보를
parsedData에 복사 후 반환

5.

dataIndexing()를 통해
myProductInfo에 저장

02 | 코드 설명 - My Product Diagram

C0.Main – MyProduct()

```

else if(click == "USED") { //U3.SeqDiagram : 9
    myProduct.strData = myProduct.getUsed(); 9 //U3.SeqDiagram : 9 , jump to C3.MyProduct
    if(myProduct.strData.contains("ERROR")) { //U3.SeqDiagram : 12
        myProduct.showErrorResult(myProduct.strData); 12 //U3.SeqDiagram : 12
    }
    else {
        myProduct.dataIndexing((ArrayList<MyProductInfo>)myProduct.dataParser(myProduct.strData)); 13 //U3.SeqDiagram : 13
        myProduct.showResult(myProduct.myProductInfo); //U3.SeqDiagram : 14
    }
}

```

14

[9~11]

사용한 선물 선택
getUsed()로 DB 조회 결과
를 반환

[12]

에러를 보여줌

[13]

[5]로 전달받은 JSON 문자열
을
변환하여 저장

[14]

ArrayList인 myProductInfo를
보여줌

U3. My Product Diagram : 6 ~

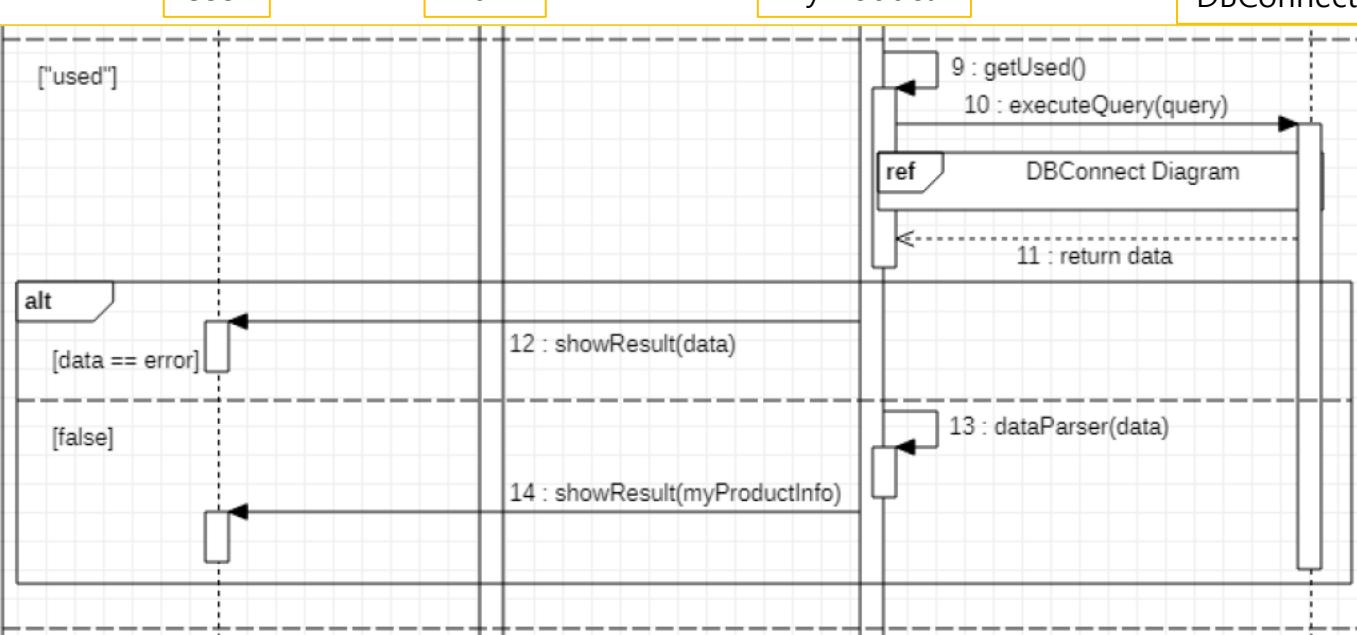
8

User

Main

MyProduct

DBConnect



02 | 코드 설명

- My Product Diagram

C0.Main – MyProduct()

```
else if(click == "REFUND") {    //U3.SeqDiagram : 15 , U8.SeqDiagram : 1  
    int nPaymentCode = menu.nextInt();  
    myProduct.refund.requestRefund(nPaymentCode);    //U3.SeqDiagram : 15, U8.SeqDiagram : 2  
}  
15  
else if(click == "EXCHANGE") {  //U3.SeqDiagram : 16, U7.SeqDiagram : 1  
    int nPaymentCode = menu.nextInt();  
    myProduct.exchange.requestExchange(nPaymentCode);    //U3.SeqDiagram : 16, U7.SeqDiagram : 2  
}  
16
```

* 인자의 출처 불분명
-> 입력으로 대체

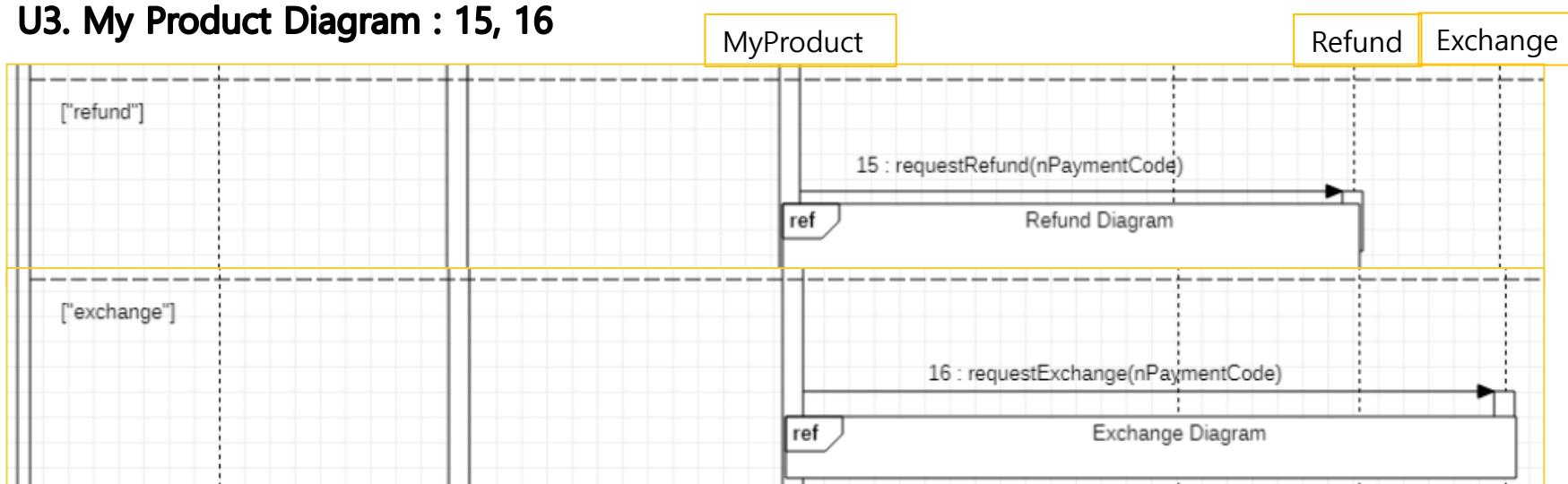
[15]

환불 선택으로 환불 메뉴로
이동
-> U8.Refund Diagram

[16]

교환 선택으로 교환 메뉴로
이동
-> U7.Exchange Diagram

U3. My Product Diagram : 15, 16



02 | 코드 설명 - My Product Diagram

C0.Main – MyProduct()

```
else if(click == "SHOW_DETAILS") { //U3.SeqDiagram : 17
    int nPaymentCode = menu.nextInt();
    //add : U3.SeqDiagram :17_1
    myProduct.strData = myProduct.dbConnect.executeQuery("__Query_ShowDetail_include_myProductInfo__");
    if(myProduct.strData.contains("ERROR")) { //add : U3.SeqDiagram :17_1
        myProduct.showErrorResult(myProduct.strData); //add : U3.SeqDiagram :17_1
    }
    else { //add : U3.SeqDiagram :15_1
        myProduct.dataIndexing((ArrayList<MyProductInfo>)myProduct.dataParser(myProduct.strData));

        for(int nIndex = 0 ; nIndex < myProduct.myProductInfo.size();nIndex++) {
            if(nPaymentCode == myProduct.myProductInfo.get(nIndex).nPaymentCode) {
                //add : U3.SeqDiagram :17_1 (user Choose routine)
                myProduct.showMyProductDetail(myProduct.myProductInfo.get(nPaymentCode)); 17
                //U3.SeqDiagram : 17, U4.SeqDiagram :16
            }
        }
    }
}
```

* 인자의 출처 불분명
-> DB조회 추가
* dbConnect
private -> public

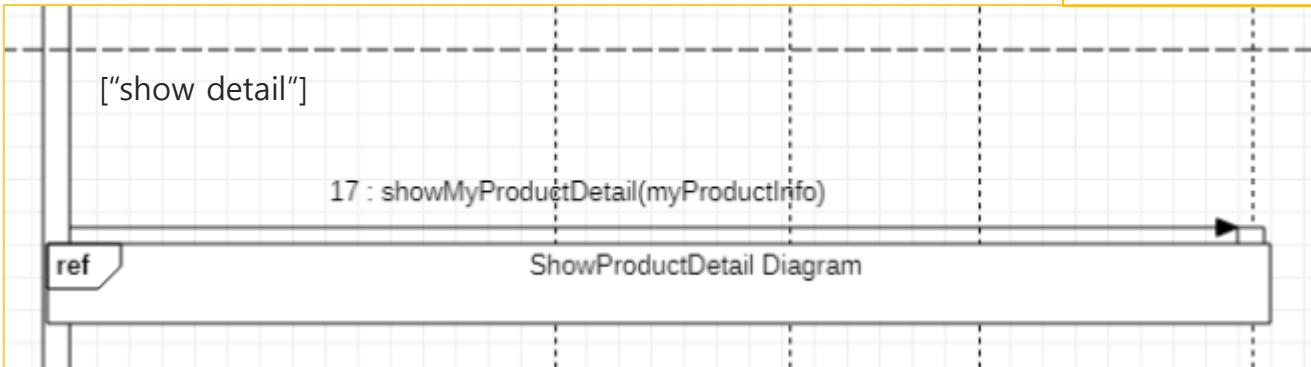
[17]

상세보기 선택으로 상세보기 페이지로 이동
-> U4-2. ShowProductDetail Diagram

U3. My Product Diagram : 17

MyProduct

ShowMyProduct Detail



02 | 코드 설명

- ShowProductDetail Diagram

C1.ProductList – showProductDetail(ChartData productData)

```

public void showProductDetail(ChartData productData) throws ParseException {
    this.showProductDetail.strData = this.showProductDetail.getProductDetail(productDataList.get(this.nProduct));
    //U4.SeqDiagram : 2, 3, jump to C4.ShowProductDetail

    if(this.showProductDetail.strData.contains("error")) { //U4.SeqDiagram : 8
        this.showProductDetail.showProductError(this.showProductDetail.strData); //U4.SeqDiagram : 8
    }
}

C4.ShowProductDetail
ductDetail.productParser(this.showProductDetail.strData); //U4.SeqDiagram(parsedData); //U4.SeqDiagram : 10

public String getProductDetail(ChartData productData)
{
    String data = this.dbConnect.executeQuery("query");
    this.dbConnect = null; //U4.SeqDiagram : 7
    return data;
}

```

3~7

6, 7

* [7]의 DB 객체 소멸
불확실

[3]

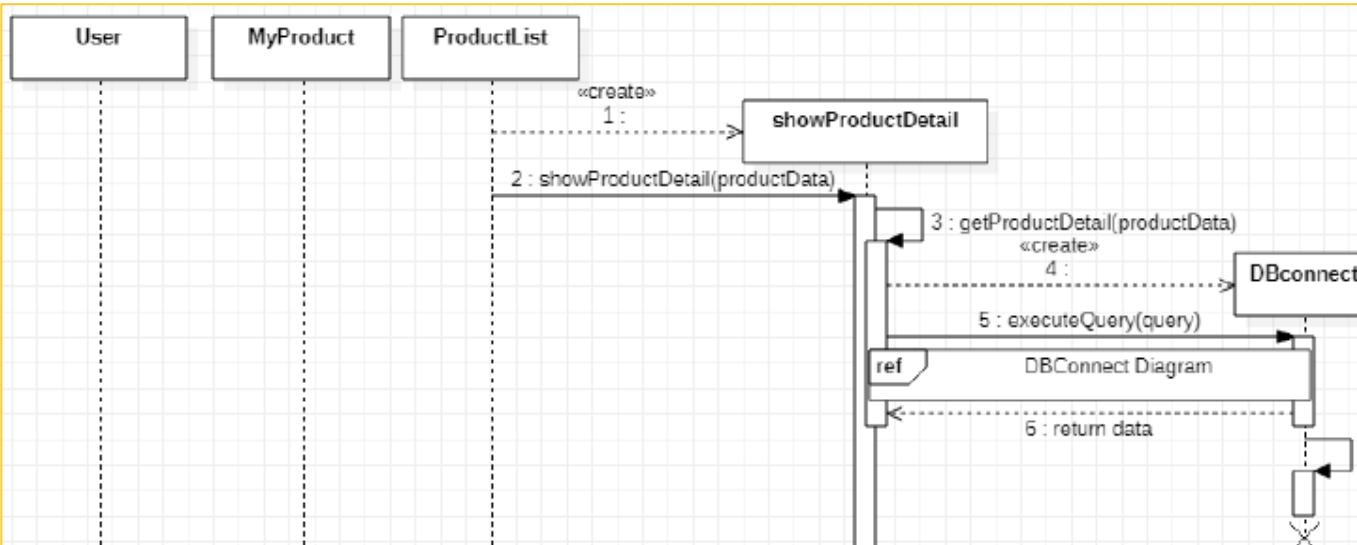
사용자가 선택한 상품의
상세보기 실행

->13슬라이드 U1.17:상품 선택

[5, 6]

DB에 쿼리를 전달하여
JSON 문자열을 저장

U4. ShowProductDetail Diagram : 3 ~ 7



[7]

C4의 getProductDetail()
내부에서 해제

02 | 코드 설명

C1.ProductList – showProductDetail(ProductData productData)

```

public void showProductDetail(ProductData productData) throws ParseException {
    this.showProductDetail.strData = this.showProductDetail.getProductDetail(productDataList.get(this.nProduct));
    //U4.SeqDiagram : 2, 3, jump to C4.ShowProductDetail

    if(this.showProductDetail.strData.contains("error")) { //U4.SeqDiagram : 8
        this.showProductDetail.showProductError(this.showProductDetail.strData);
    }
    else {
        ProductData parsedData = this.showProductDetail.productParser(this.showProductDetail.strData); //U4.SeqDiagram : 9
        this.showProductDetail.showProductDetail(parsedData); //U4.SeqDiagram : 10
    }
    Scanner scanner = new Scanner(System.in);
    String Pay = scanner.nextLine(); //U4.SeqDiagram : 11
    if(Pay == "Pay") { //U4.SeqDiagram : 11 , user requestPayment
        this.showProductDetail.requestPayment(parsedData); //U4.SeqDiagram : 13, jump to (U5)C5.Payment
    }
}
}

```

8

//U4.SeqDiagram : 8

9, 10

11

* showProductError()
추가

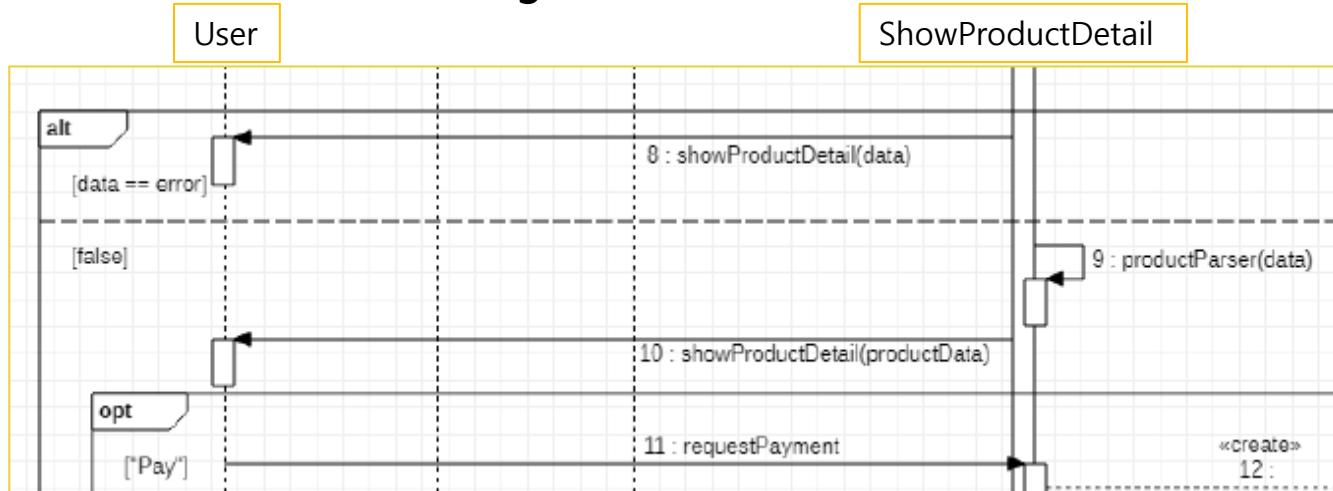
[8]

DB 반환이 에러라면
에러 결과를 보여줌

[9, 10]

JSON 문자열을 변환하
여 저장

U4. ShowProductDetail Diagram : 8 ~ 11



[11]

사용자의 결제 요청

02 | 코드 설명

- ShowProductDetail Diagram

C1.ProductList – showProductDetail(ProductoData productData)

```
else {  
    ProductoData parsedData = this.showProductDetail.productParser(this.showProductDetail.strData); //U4.S  
    this.showProductDetail.showProductDetail(parsedData); //U4.SeqDiagram : 10  
  
    Scanner scanner = new Scanner(System.in);  
    String Pay = scanner.nextLine(); //U4.SeqDiagram : 11  
    if(Pay == "Pay") { //U4.SeqDiagram : 11 , user requestPayment  
        this.showProductDetail.requestPayment(parsedData); //U4.SeqDiagram : 13, jump to (U5)C5.Payment  
    }  
}
```

11

[12]

결제 객체 생성

C4.ShowProductDetail – requestPayment(ProductoData productData)

```
public void requestPayment(ProductoData productData) throws ParseException {  
    Payment pay = new Payment(); //U4.SeqDiagram : 12  
    pay.makePayment(productData.nProductCode); //U5.PayDiagram, U5.SeqDiagram : 13  
    pay = null; //U4.SeqDiagram : 14  
}
```

12 ~ 14

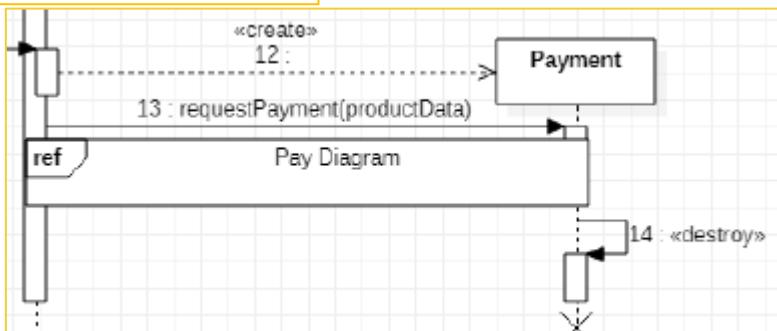
[13~14]

makePayment()로
결제 진행

객체 해제

U4. ShowProductDetail Diagram : 12 ~ 14

ShowProductDetail



02 | 코드 설명

C3. MyProduct – showMyProductDetail(MyProductInfo myProductInfo)

```
public void showMyProductDetail(MyProductInfo myProductInfo) throws ParseException {
    //U4.SeqDiagram :16, add-error : not in class_diagram
    ShowMyProductDetail showMyProductDetail = new ShowMyProductDetail(); //U4.SeqDiagram : 15
    String data1 = showMyProductDetail.getMyProductInfo(myProductInfo); //U4.SeqDiagram :17
    String data2 = showMyProductDetail.getPaymentInfo(myProductInfo); //U4.SeqDiagram :21
    15, 16, 17
    if(data1.contains("error")||data2.contains("error")) { //U4.SeqDiagram :24
        showMyProductDetail.showProductError(data1, data2); //U4.SeqDiagram :24, add : showProductError()
    }
    else {
        showMyProductDetail.myProductInfo = showMyProductDetail.productParser(data1); //U4.SeqDiagram :25
        showMyProductDetail.paymentInfo = showMyProductDetail.paymentParser(data2); //U4.SeqDiagram :26
        showMyProductDetail.showProductDetail(showMyProductDetail.myProductInfo,showMyProductDetail.paymentInfo);
        //U4.SeqDiagram :27
    }
    this.dbConnect = null; //U4.SeqDiagram :28
}
```

* 클래스 다이어그램에
미 표기 -> 추가
>showMyProductDetail()
* 인자 정보 불확실
상품 목록 or 상품 정보

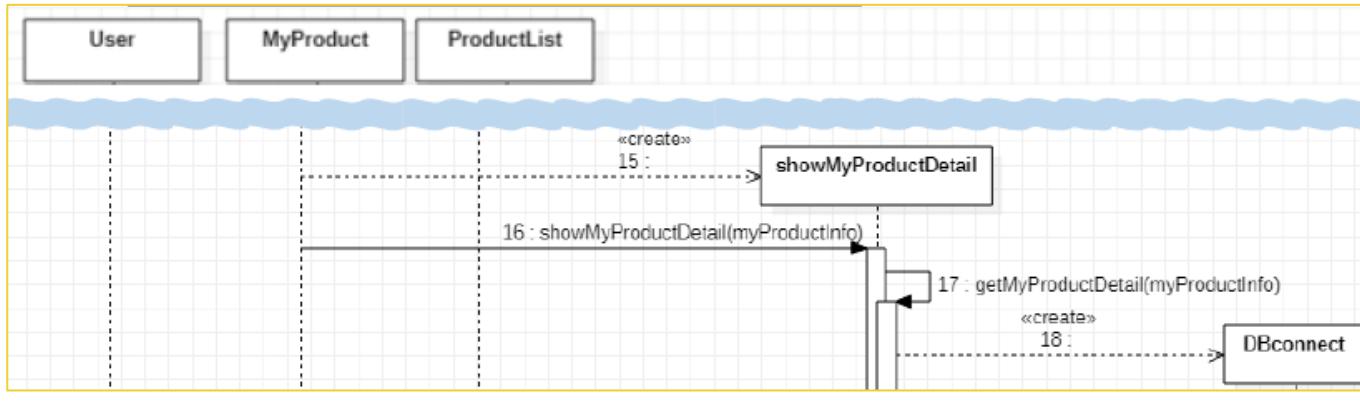
[15]

객체 생성

[17]

상품 정보를 가져오기 위해
getMyProductDetail() 실행

U4. ShowProductDetail Diagram : 15 ~ 17



C3. MyProduct – showMyProductDetail(MyProductInfo myProductInfo)

```

public void showMyProductDetail(MyProductInfo myProductInfo) throws ParseException {
    //U4.SeqDiagram :16, add-error : not in class-diagram
    ShowMyProductDetail showMyProductDetail = new ShowMyProductDetail(); //U4.SeqDiagram : 15
    String data1 = showMyProductDetail.getMyProductInfo(myProductInfo); //U4.SeqDiagram :17
    String data2 = showMyProductDetail.getPaymentInfo(myProductInfo); //U4.SeqDiagram :21

    if(data1.contains("error")||data2.contains("error")) { //U4.SeqDiagram :24
        showMyProductDetail.showProductError(data1, data2); //U4.SeqDiagram :24, add : showProductError()
    }
    else {
        showMyProductDetail.myProductInfo = showMyProductDetail.productParser(data1); //U4.SeqDiagram :25
        showMyProductDetail.paymentInfo = showMyProductDetail.paymentParser(data2); //U4.SeqDiagram :26
        showMyProductDetail.showProductDetail(showMyProductDetail.myProductInfo,showMyProductDetail.paymentInfo);
        //U4.SeqDiagram :27
    }
    this.dbConnect = null; //U4.SeqDiagram :28
}

```

[17~20]

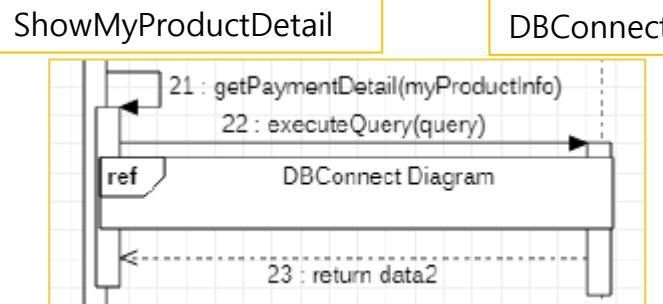
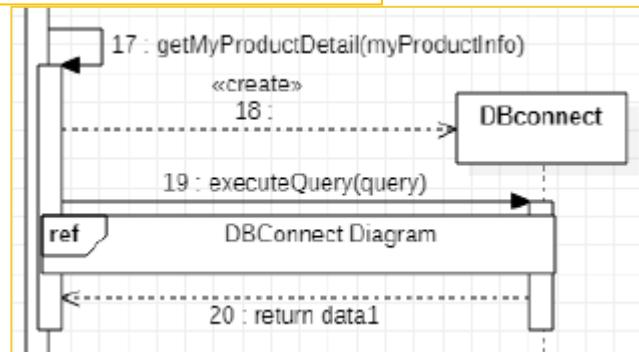
상품 정보를 가져오기 위해
DB 조회

[21~23]

결제 정보를 가져오기 위해
DB 조회

U4. ShowProductDetail Diagram : 17 ~ 23

ShowMyProductDetail



C3. MyProduct – showMyProductDetail(MyProductInfo myProductInfo)

```

if(data1.contains("error")||data2.contains("error")) { //U4.SeqDiagram :24
    showMyProductDetail.showProductError(data1, data2); //U4.SeqDiagram :24, add : showProductError()
}
else {
    showMyProductDetail.myProductInfo = showMyProductDetail.productParser(data1); //U4.SeqDiagram :25
    showMyProductDetail.paymentInfo = showMyProductDetail.paymentParser(data2); //U4.SeqDiagram :26
    showMyProductDetail.showProductDetail(showMyProductDetail.myProductInfo, showMyProductDetail.paymentInfo);
    //U4.SeqDiagram :27
}
this.dbConnect = null; //U4.SeqDiagram :28

```

24
25~27
28

[24]

두 반환 값 중 하나라도 에러가 발생하면 에러 결과를 보여줌

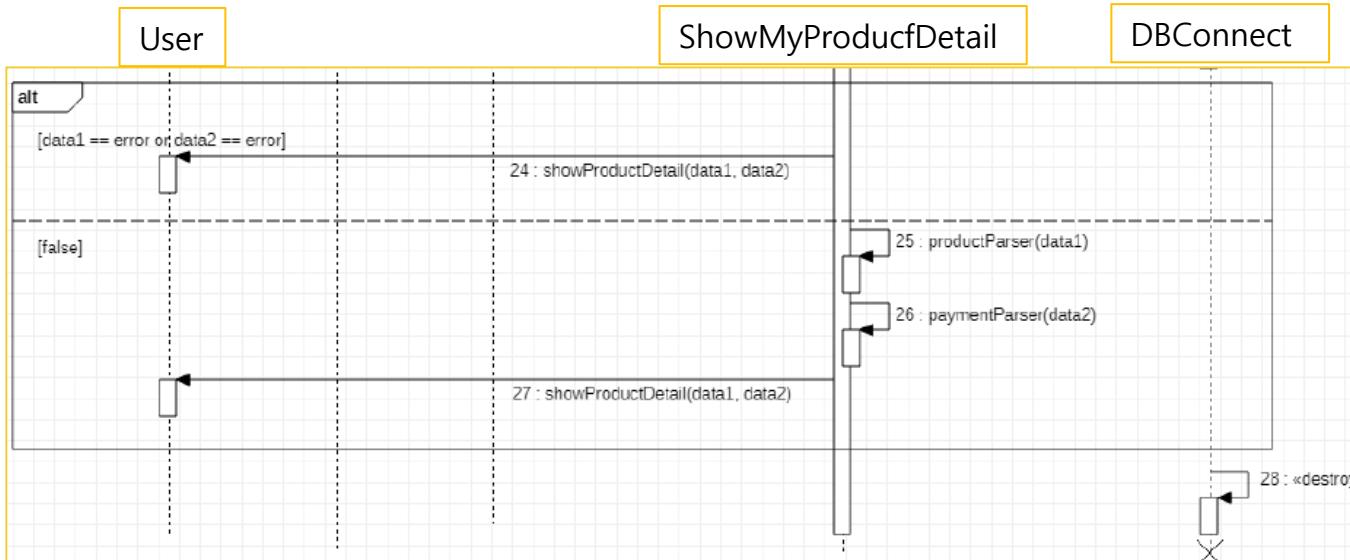
[25~27]

정상적인 경우,
DB 조회를 통해 얻은 JSON
문자열을 변환하여 저장

[28]

객체 해제

U4. ShowProductDetail Diagram : 17 ~ 23



C4. ShowMyProductInfo

```

public String getPaymentInfo(MyProductInfo myProductInfo) { //U4.SeqDiagram :21
    return dbConnect.executeQuery("query"); //U4.SeqDiagram :19, 20
}
public String getMyProductInfo(MyProductInfo myProductInfo) { //U4.SeqDiagram :17
    return dbConnect.executeQuery("query"); //U4.SeqDiagram :22, 23
}

```

[17~20]

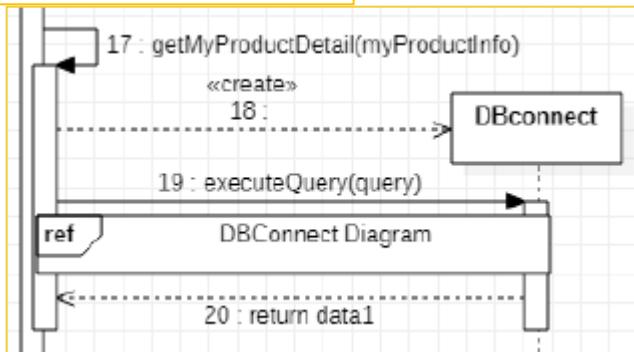
상품 정보를 가져오기 위해
DB 조회

[21~23]

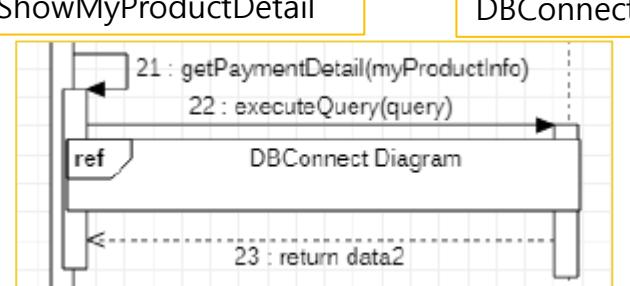
결제 정보를 가져오기 위해
DB 조회

U4. ShowProductDetail Diagram : 17 ~ 23

ShowMyProductDetail



ShowMyProductDetail



02 | 코드 설명 -Pay Diagram

C4.ShowProductDetail-requestPayment(ChartData)

```
public void requestPayment(ChartData productData) throws ParseException { //U4.SeqDiagram :13  
    Payment pay = new Payment(); //U4.SeqDiagram :12  
    2 pay.makePayment(productData.nProductCode); //U5.PayDiagram, U5.SeqDiagram : 1  
    pay = null; //U4.SeqDiagram :14  
}
```

[2]

ShowProductDetail
클래스에서 makePayment()
메서드 호출

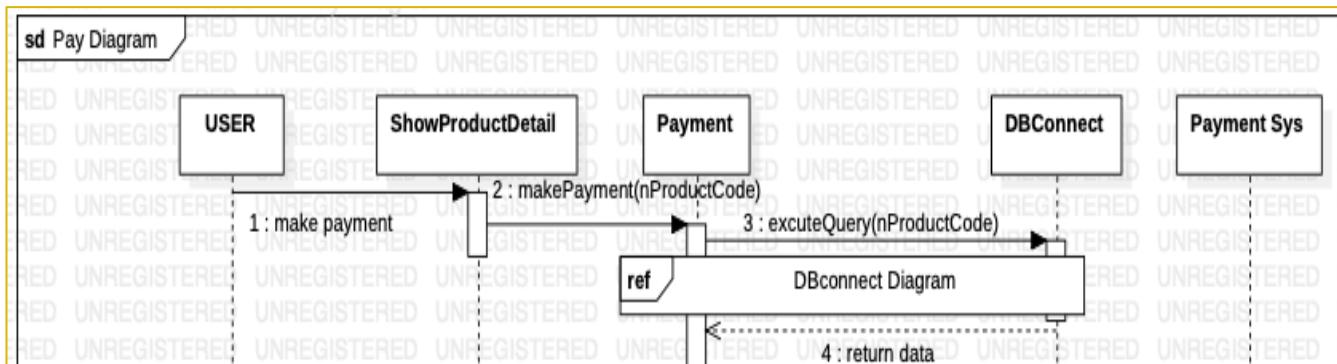
C5.Payment-makePayment(nProductCode)

```
public void makePayment(int nProductCode) throws ParseException { //U5.SeqDiagram : 2, error : no return int_void  
2 MyFriendsList myFriendList; //external system struct  
4 this.strData = dbConnect.executeQuery("nProductCode_Query"); 3 //U5.SeqDiagram : 3, 4
```

[3]

makePayment 메소드에서
executeQuery() 메소드 호출

U5. Pay Diagram : 1~4



[4]

executeQuery() 메소드 리턴
값 strData에 저장

02 | 코드 설명 -Pay Diagram

C5.Payment-makePayment(nProductCode)

```
if(!strData.contains("error")){ //U5.SeqDiagram add : error check  
5 this.paymentinfo = this.dataParser(strData); //U5.SeqDiagram : 5
```

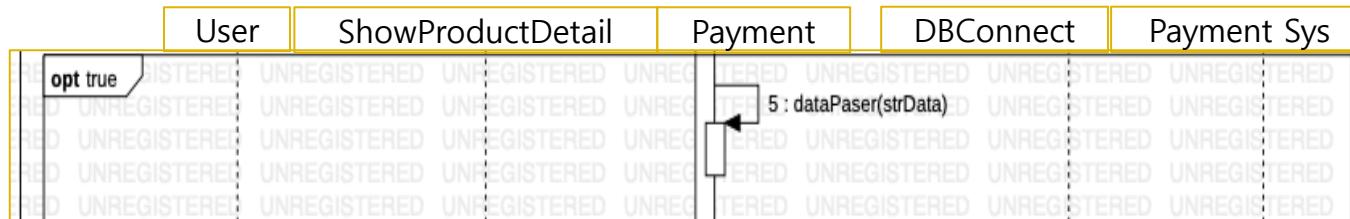
[5]

Payment 클래스의 dataParser() 메소드를 통해 DB에서 가져온 값을 paymentinfo에 맞게 가공함

C5.Payment – dataParser(data)

```
private PaymentInfo dataParser(String data) throws ParseException{ //modify : return type PaymentInfo Boolean  
5 PaymentInfo parsedData = new PaymentInfo();  
  
/*Parse from JSON String data*/  
try {  
    JSONObject jObj = (JSONObject)this.jParse.parse(data);  
  
    parsedData.strProductName = (String)jObj.get("strProductName");  
    parsedData.strProductImage = (String)jObj.get("strProductImage");  
    parsedData.nProductPrice = (Integer)jObj.get("nProductPrice");  
    parsedData.nShoppingPoint = (Integer)jObj.get("nShoppingPoint");  
} catch(ParseException e) { e.printStackTrace(); }  
  
return parsedData;  
}
```

U5. Pay Diagram : 5



02 | 코드 설명 -Pay Diagram

C5.Payment-makePayment(nProductCode)

```

6 this.nReceiver=this.selectFriend(myFriendList) //U5.SeqDiagram : 6, external system object
7 this.nQuantity = this.getQuantity(); //U5.SeqDiagram : 7
8 this.strMessage = this.getMessage(); //U5.SeqDiagram : 8

```

오류 :
 - 클래스다이어그램 내부에
 selectFriend() 메서드의 리턴
 자료형이 String이라고 표기되
 어 있음

C5.Payment-getQuantity(),getMessage(),useShoppingPoint(), calcTotalPrice(nProductPrice,nQuantity,nUsedPoint)

```

private int getQuantity(){//U5.SeqDiagram : 7
    System.out.print("Input Quantity :");
    return scanner.nextInt();
}

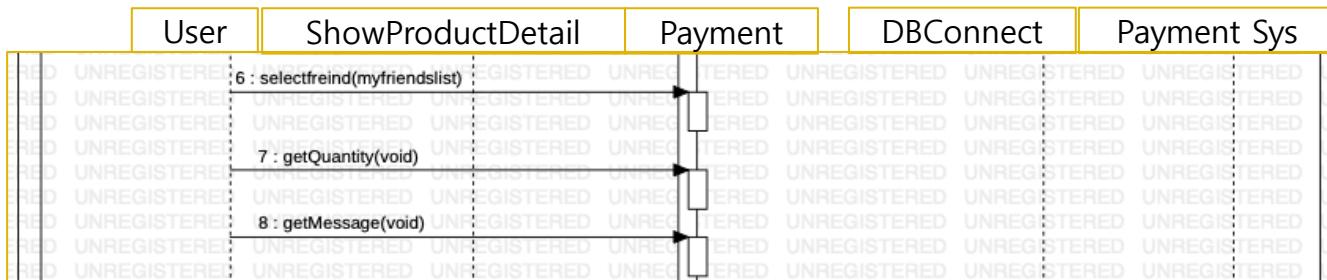
private String getMessage() { //U5.SeqDiagram : 8
    System.out.print("Input Message :");
    return scanner.nextLine();
}

```

[6]
 친구목록에서 선택된 친구의
 고유 ID를 반환하여
 nReceiver에 저장

[7]
 수량을 입력받음
 nQuantity에 저장

U5. Pay Diagram : 6~8



[8]
 메시지를 입력받음
 strMessage에 저장

02 | 코드 설명 -Pay Diagram

C5.Payment-makePayment(nProductCode)

```
9 this.nUsedPoint = this.useShoppingPoint(); //U5.SeqDiagram : 9  
10 this.nTotalPrice = this.calcTotalPrice(this.paymentinfo.nProductPrice, this.nQuantity, this.nUsedPoint); //U5.SeqDiagram : 10,
```

[9]

사용하고자하는 쇼핑 포인트
입력받음
nUsedPoint에 저장

C5.Payment-getQuantity(), getMessage(), useShoppingPoint(), calcTotalPrice(nProductPrice, nQuantity, nUsedPoint)

```
private int useShoppingPoint(){ //U5.SeqDiagram : 9  
    System.out.print("Input ShoppingPoints:");  
    return scanner.nextInt();  
}  
  
private int calcTotalPrice(int nProductPrice, int nQuantity, int nUsedPoint) { //U5.SeqDiagram : 10  
    return ((nProductPrice*nQuantity)-nUsedPoint);  
}
```

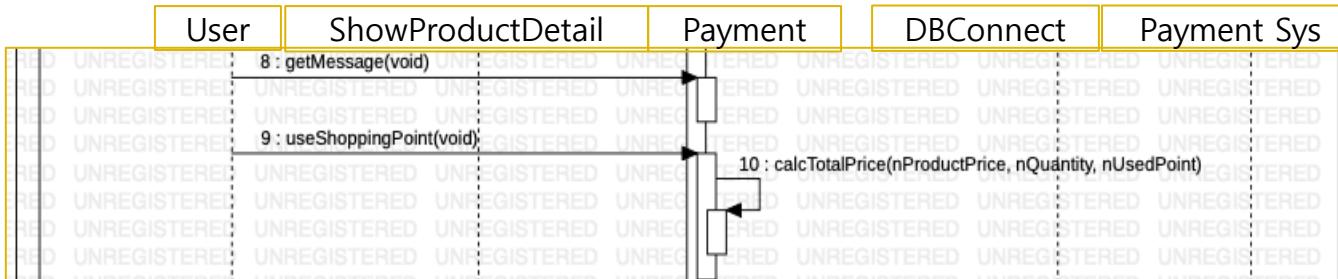
9

10

[10]

상품가격과 대수를 곱한값에
포인트 금액을 차감하여 리
턴
nTotalPrice에 저장

U5. Pay Diagram : 9, 10



02 | 코드 설명 -Pay Diagram

C5.Payment-makePayment(nProductCode)

```
boolean result = this.sendResult(this.nPaymentCode, myFriendList.strMyFriendCode, this.strMessage); //U5.SeqDiagram : 11  
13  
11
```

Payment Diagram에서
결제 수단 정하고 음

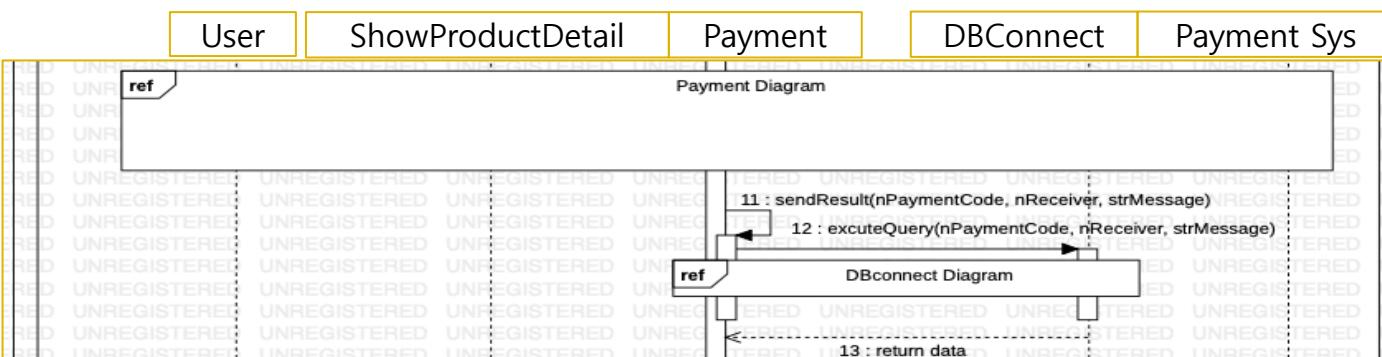
C5.Payment-sendResult(nPaymentCode, strReceiver,strMessage)

```
private boolean sendResult(int nPaymentCode, String strReceiver, String strMessage) throws ParseException { //U5.SeqDiagram : 11  
String queryPaymentCodeReceiverMessage = "query" + Integer.toString(nPaymentCode) + strReceiver + strMessage;  
JSONObject jObj = (JSONObject)this.jParse.parse(this.dbConnect.executeQuery(queryPaymentCodeReceiverMessage)); //U5.SeqDiagram : 12, 13  
12  
return (Boolean)jObj.get("sendResult");  
}  
11
```

[11] DB를 통해
받는 사람에게
메시지와 상품을 전송

[12] sendResult() 메소드에서
executeQuery() 메소드 호출

U5. Pay Diagram : 11~13



[13] executeQuery() 메소드 리턴
값 result에 저장

02 | 코드 설명 -Pay Diagram

C5.Payment-makePayment(nProductCode)

```

if(result){14
    System.out.println("success"); //U5.SeqDiagram : 14
}
else{
    15 this.cancelPay(this.nPaymentCode); //U5.SeqDiagram : 15
    System.out.println("Fail"); //U5.SeqDiagram : 16
}

```

[14]

result의 값이 true인 경우 사용자에게 결제 성공의 메시지를 출력

C5.Payment-cancelPay(nPaymentCode)

```

private void cancelPay(int nPaymentCode) { //U5.SeqDiagram : 15
    // external system
}

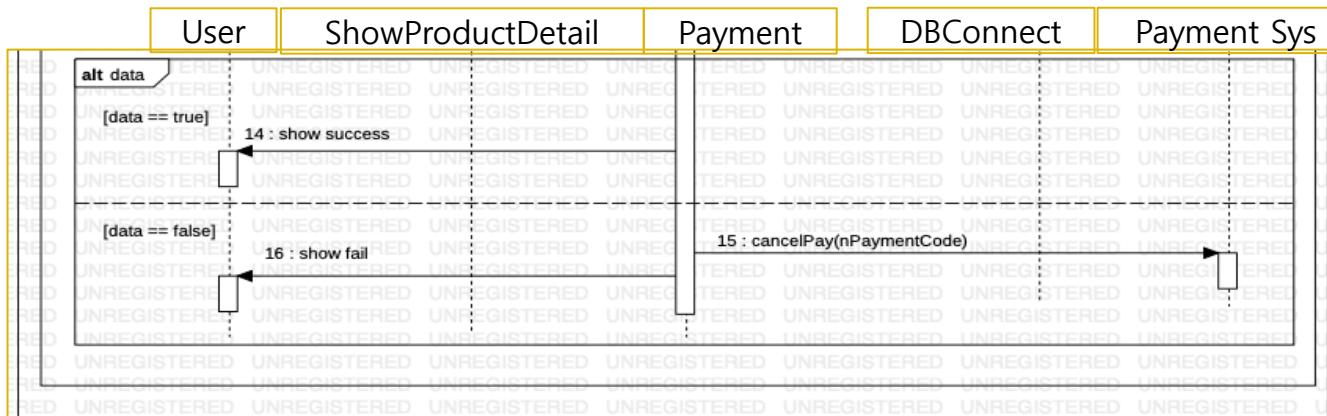
```

15

[15]

결제를 취소하기 위해 cancelPay() 메소드를 실행함

U5. Pay Diagram : 14~16



[16]

result의 값이 false인 경우 사용자에게 결제 실패의 메시지를 출력

02 | 코드 설명 -Payment Diagram

C5.Payment-makePayment(nProductCode)

```
while(this.nPaymentCode == 0){//jump to U6.PaymentDiagram  
1 this.nPaymentMethod = this.getPaymentMethod(); //U6.SeqDiagram : 1
```

어떤 메소드에서 실행
되는지 적혀 있지 않음
=> Payment 객체의
makePayment() 메소드
에서 구현함

C5.Payment-getPaymentMethod()

```
public int getPaymentMethod() {  
    System.out.print("Choose PaymentMethod :");  
    return scanner.nextInt();  
}
```

[1]
결제 방식을 입력 받아
nPaymentMethod에 저장

U6. Payment Diagram : 1



02 | 코드 설명 -Payment Diagram

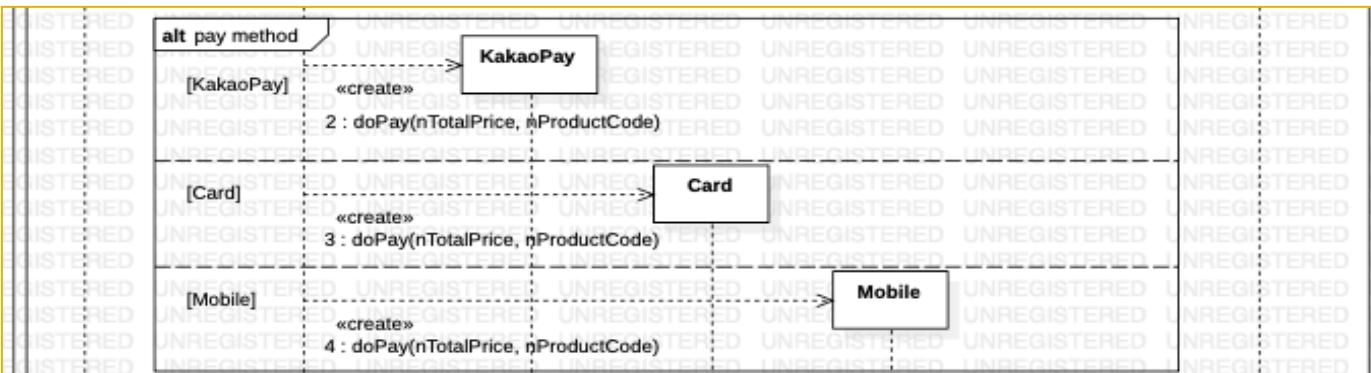
C6.DoPay

```
while(this.nPaymentCode == 0){//jump to U6.PaymentDiagram  
    this.nPaymentMethod = this.getPaymentMethod(); //U6.SeqDiagram : 1  
  
    if(this.nPaymentMethod==1){  
        KakaoPay kakaoPay= new KakaoPay(this.nTotalPrice,this.nProductCode); //U6.SeqDiagram : 2  
        this.nPaymentCode = kakaoPay.requestPay(kakaoPay.nTotalPrice, kakaoPay.nProductCode); //U6.SeqDiagram : 5  
        kakaoPay = null; //U6.SeqDiagram : 17  
    }  
    else if(this.nPaymentMethod==2){  
        CreditCardPay card = new CreditCardPay(this.nTotalPrice, this.nProductCode); //U6.SeqDiagram : 3  
        this.nPaymentCode = card.requestPay(card.nTotalPrice, card.nProductCode); //U6.SeqDiagram : 7  
        card = null; //U6.SeqDiagram : 18  
    }  
    else if(this.nPaymentMethod==3){  
        MobilePay mobilePay = new MobilePay(this.nTotalPrice, this.nProductCode); //U6.SeqDiagram : 4  
        this.nPaymentCode = mobilePay.requestPay(mobilePay.nTotalPrice, mobilePay.nProductCode); //U6.SeqDiagram : 9  
        mobilePay = null; //U6.SeqDiagram : 19  
    }  
}
```

[2, 3, 4]

입력 방식에 따라
결제 객체 생성

U6. Payment Diagram : 2, 3, 4



02 | 코드 설명 -Payment Diagram

C6. DoPay

```
class DoPay{ //class 6
    protected int nTotalPrice;
    protected int nProductCode;

    protected int requestPay(int nTotalPrice, int nProductCode) {
        if(Success) {
            return nPaymentCode_FromExternalSystem;
        }
        else {
            return 0;
        }
    }
    public DoPay(int nTotalPrice, int nProductCode) {
        this.nTotalPrice = nTotalPrice;
        this.nProductCode = nProductCode;
    }
}
```

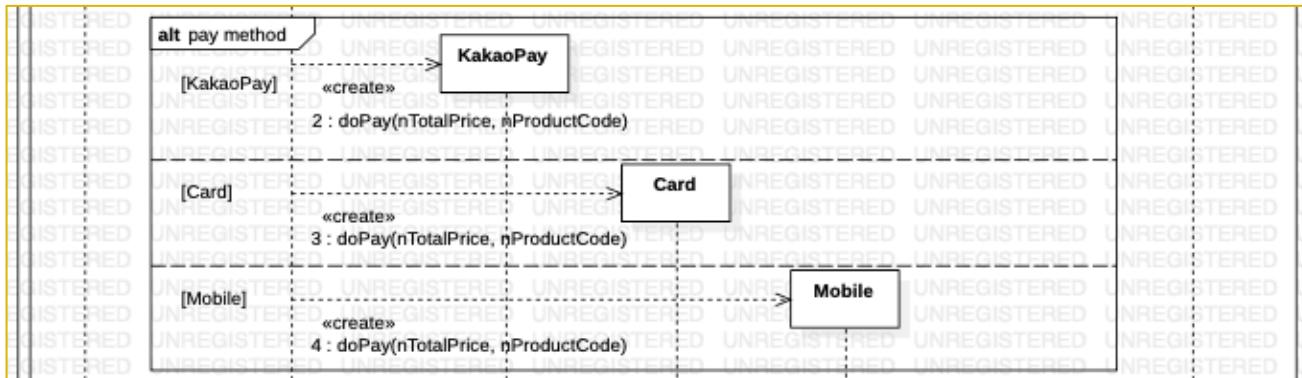
KakaoPay, MobilePay, CreditCardPay의 부모
DoPay

*클래스 다이어그램
인터페이스
*관계선
일반화 관계

*DoPay의 생성자인데
doPay라고 표기함

[2, 3, 4]
입력 방식에 따라
결제 객체 생성

U6. Payment Diagram : 2, 3, 4



02 | 코드 설명 -Payment Diagram

C5.Payment-makePayment(nProductCode)

```

if(this.nPaymentMethod==1){
    KakaoPay kakaoPay= new KakaoPay(this.nTotalPrice,this.nProductCode); //U6.SeqDiagram : 2
    2
    6 this.nPaymentCode = kakaoPay.requestPay(kakaoPay.nTotalPrice, kakaoPay.nProductCode); //U6.SeqDiagram : 5
    kakaoPay = null; //U6.SeqDiagram : 17
    5
}
  
```

*어떤 메소드에서 실행되는지 적혀 있지 않음
=> Payment 객체의 makePayment() 메소드에서 구현

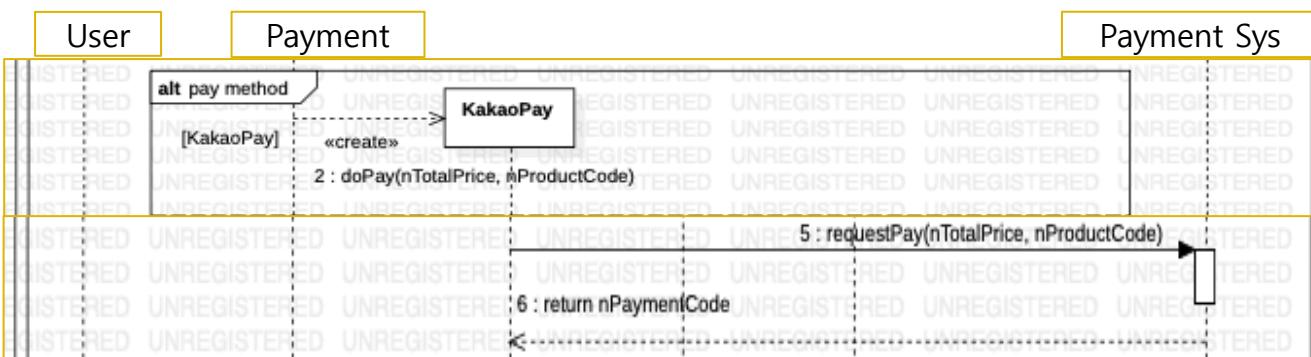
C6.KakaoPay

```

class KakaoPay extends DoPay{ //class 6
    2 public KakaoPay(int nTotalPrice, int nProductCode) {
        super(nTotalPrice, nProductCode);
        // TODO Auto-generated constructor stub
    }
    5 protected int requestPay(int nTotalPrice, int nProductCode) { //U6.SeqDiagram : 5
        boolean Success;
        if(Success) {
            return nPaymentCode_FromExternalSystem; //U6.SeqDiagram : 6, 11
        }
        else {
            return 0; //U6.SeqDiagram : 6, 14
        }
    }
}
  
```

[2] 부모 클래스인 DoPay에 있는 생성자를 통해 KakaoPay 객체를 생성

U6. Payment Diagram : 2, 5, 6



[5] 결제를 외부시스템으로 진행 후 성공 결과 리턴

[6] requestPay() 메소드의 반환 값을 nPaymentCode에 저장

02 | 코드 설명 -Payment Diagram

C5.Payment-makePayment(nProductCode)

```
else if(this.nPaymentMethod==2){  
    CreditCardPay card = new CreditCardPay(this.nTotalPrice, this.nProductCode); //U6.SeqDiagram : 3  
    8 this.nPaymentCode = card.requestPay(card.nTotalPrice, card.nProductCode); //U6.SeqDiagram : 7,  
    card = null; //U6.SeqDiagram : 18  
}
```

3

7

C6.CreditCardPay

```
class CreditCardPay extends DoPay{ //class 6  
    3 public CreditCardPay(int nTotalPrice, int nProductCode) {  
        super(nTotalPrice, nProductCode);  
        // TODO Auto-generated constructor stub  
    }  
    7 protected int requestPay(int nTotalPrice, int nProductCode) { //U6.SeqDiagram : 7.  
        if(Success) {  
            return nPaymentCode_FromExternalSystem; //U6.SeqDiagram : 8, 12  
        }  
        else {  
            return 0; //U6.SeqDiagram : 8, 15  
        }  
    }  
}
```

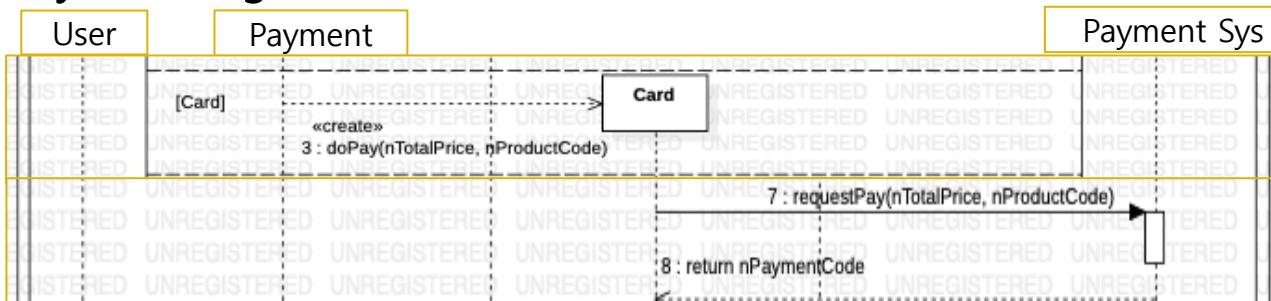
[3]

부모 클래스인 DoPay에 있는 생성자를 통해 CreditCardPay 객체를 생성

[7]

결제를 외부시스템으로 진행 후 성공 결과 리턴

U6. Payment Diagram : 3, 7, 8



[8]

requestPay() 메소드의 반환 값을 nPaymentCode에 저장

02 | 코드 설명 -Payment Diagram

C5.Payment-makePayment(nProductCode)

```

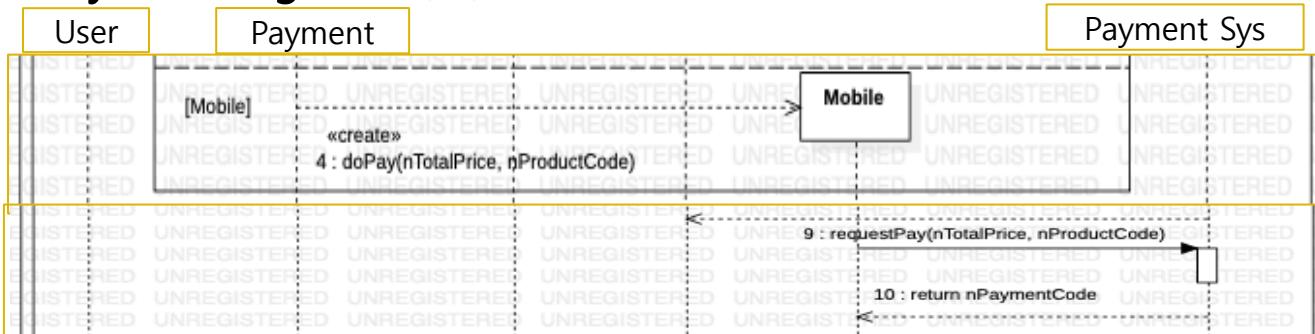
else if(this.nPaymentMethod==3){
    MobilePay mobilePay = new MobilePay(this.nTotalPrice, this.nProductCode); //U6.SeqDiagram : 4
    10 this.nPaymentCode = mobilePay.requestPay(mobilePay.nTotalPrice, mobilePay.nProductCode); //U6.SeqDiagram : 9
    mobilePay = null; //U6.SeqDiagram : 19
}
  
```

C6.MobilePay

```

class MobilePay extends DoPay{ //class 6
    public MobilePay(int nTotalPrice, int nProductCode) {
        4 super(nTotalPrice, nProductCode);
        // TODO Auto-generated constructor stub
    }
    9 protected int requestPay(int nTotalPrice, int nProductCode) { //U6.SeqDiagram : 9
        if(Success) {
            return nPaymentCode_FromExternalSystem; //U6.SeqDiagram : 10, 13
        }
        else {
            return 0; //U6.SeqDiagram : 10, 16
        }
    }
}
  
```

U6. Payment Diagram : 4, 9, 10



[4] 부모 클래스인 DoPay에 있는 생성자를 통해 MobilePay 객체를 생성

[9] 결제를 외부시스템으로 진행 후 성공 결과 리턴

[10] requestPay() 메소드의 반환 값을 nPaymentCode에 저장

02 | 코드 설명 -Payment Diagram

C5.Payment-makePayment(nProductCode)

```
while(this.nPaymentCode == 0){//jump to U6.PaymentDiagram 11, 12, 13
    this.nPaymentMethod = this.getPaymentMethod(); //U6.SeqDiagram : 1

    if(this.nPaymentMethod==1){
        KakaoPay kakaoPay= new KakaoPay(this.nTotalPrice,this.nProductCode); //U6.SeqDiagram : 2
        this.nPaymentCode = kakaoPay.requestPay(kakaoPay.nTotalPrice, kakaoPay.nProductCode); //U6.SeqDiagram : 5, else
        kakaoPay = null;//U6.SeqDiagram : 17
    }
    else if(this.nPaymentMethod==2){
        CreditCardPay card = new CreditCardPay(this.nTotalPrice,this.nProductCode); //U6.SeqDiagram : 3
        this.nPaymentCode = card.requestPay(card.nTotalPrice, card.nProductCode); //U6.SeqDiagram : 7, : Diagram 6 and
        card = null;//U6.SeqDiagram : 18
    }
    else if(this.nPaymentMethod==3){
        MobilePay mobilePay = new MobilePay(this.nTotalPrice, this.nProductCode); //U6.SeqDiagram : 4
        this.nPaymentCode = mobilePay.requestPay(mobilePay.nTotalPrice, mobilePay.nProductCode); //U6.SeqDiagram : 9
        mobilePay = null;//U6.SeqDiagram : 19
    }
}
```

[11, 12, 13]

nPaymentCode의 값이 0이면
계속 반복문을 진행함

U6. Payment Diagram : 11~13



02 | 코드 설명 -Payment Diagram

C5.Payment-makePayment(nProductCode)

```

while(this.nPaymentCode == 0){//jump to U6.PaymentDiagram 14, 15, 16
    this.nPaymentMethod = this.getPaymentMethod(); //U6.SeqDiagram : 1

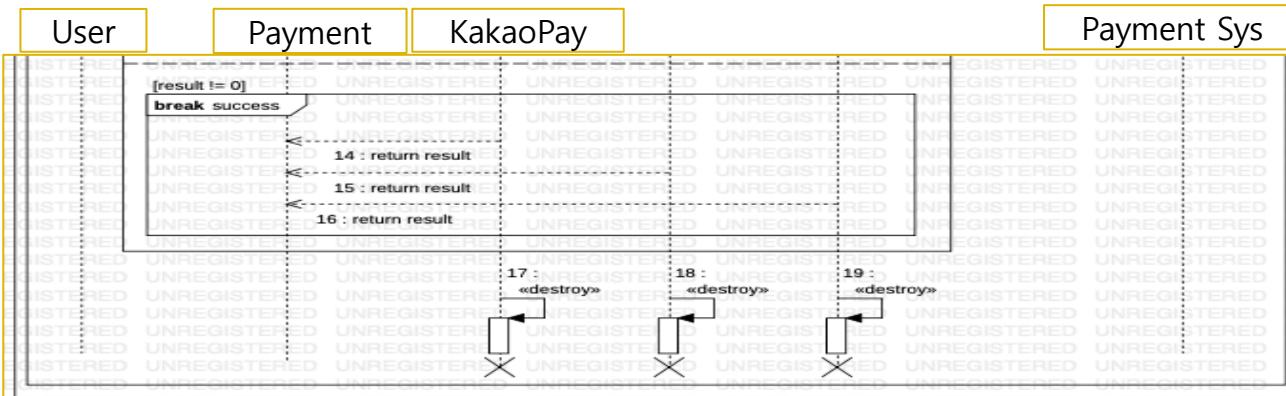
    if(this.nPaymentMethod==1){
        KakaoPay kakaoPay= new KakaoPay(this.nTotalPrice,this.nProductCode); //U6.SeqDiagram : 2
        this.nPaymentCode = kakaoPay.requestPay(kakaoPay.nTotalPrice, kakaoPay.nProductCode); //U6.SeqDiagram : 5, end
    17 kakaoPay = null; //U6.SeqDiagram : 17
    }
    else if(this.nPaymentMethod==2){
        CreditCardPay card = new CreditCardPay(this.nTotalPrice,this.nProductCode); //U6.SeqDiagram : 3
        this.nPaymentCode = card.requestPay(card.nTotalPrice, card.nProductCode); //U6.SeqDiagram : 7, : Diagram 6 and
    18 card = null; //U6.SeqDiagram : 18
    }
    else if(this.nPaymentMethod==3){
        MobilePay mobilePay = new MobilePay(this.nTotalPrice, this.nProductCode); //U6.SeqDiagram : 4
        this.nPaymentCode = mobilePay.requestPay(mobilePay.nTotalPrice, mobilePay.nProductCode); //U6.SeqDiagram : 9,
    19 mobilePay = null; //U6.SeqDiagram : 19
    }
}

```

[14, 15, 16]
nPaymentCode의 값이 0이 아니면 반복문을 탈출함

[17, 18, 19]
반복문이 한번 돌때마다 생성된 객체를 제거

U6. Payment Diagram : 14~19



02 | 코드 설명 – Exchange

C0. Main – MyProduct()

```
else if(click == "EXCHANGE") { 1 //U3.SeqDiagram : 16, U7.SeqDiagram : 1  
    int nPaymentCode = menu.nextInt();  
    myProduct.exchange.requestExchange(nPaymentCode); 2 //U3.SeqDiagram : 16, U7.SeqDiagram : 2  
}
```

[1]

사용자가 교환을 요청하면
MyProduct()에서 교환을 실
행

C7. Exchange – requestExchange(int nPaymentCode)

```
public void requestExchange(int nPaymentCode) { //U7.SeqDiagram : 2  
    4 this.strData = dbConnect.executeQuery(this.strCheckExcValid); //U7.SeqDiagram : 3, 4  
                                         3  
    if(this.strData == "false") { //false  
        5 this.sendMessage(strCannotExchangeMessage); //U7.SeqDiagram : 5  
    }  
}
```

[2]

교환 실행
requestExchange()
*인자 nPaymentCode가
존재하지 않아 입력 대체

[3]

유효성 쿼리 실행

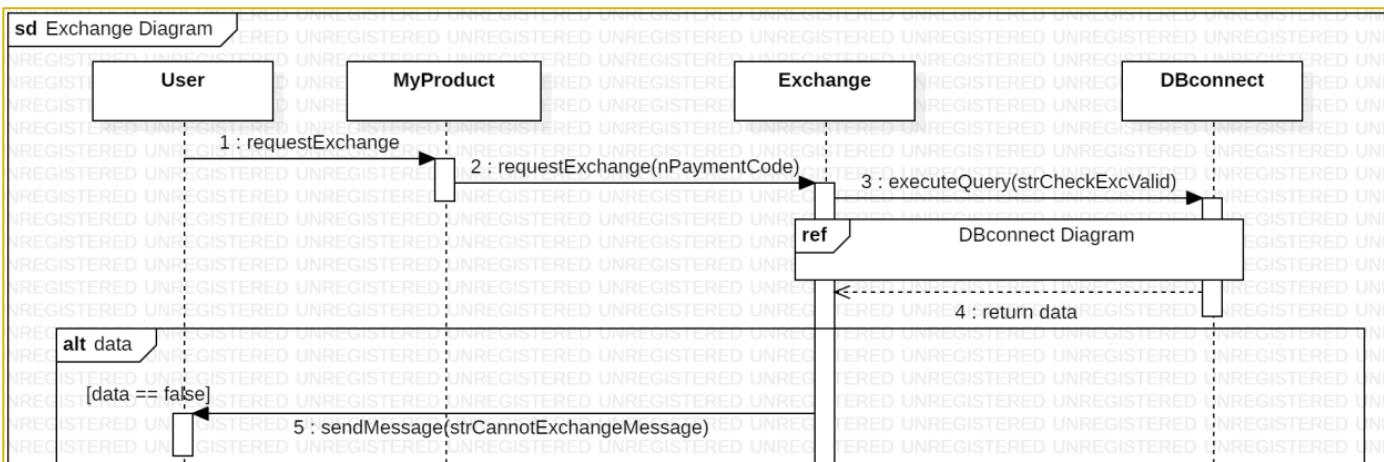
[4]

쿼리 결과를 문자열로 반환

[5]

Data가 false면
교환불가 메시지 전송

U7. Exchange Diagram : 1 ~ 5



02 | 코드 설명 -Exchange

C7. Exchange – requestExchange(int nPaymentCode)

```
else { //true          6  
    if (this.oneonOneInquiry() == false) {    //U7.SeqDiagram : 6  
        10 this.sendMessage(this.strCannotExchangeMessage); //U7.SeqDiagram : 10  
    }  
}
```

[6]
1대1 문의 진행

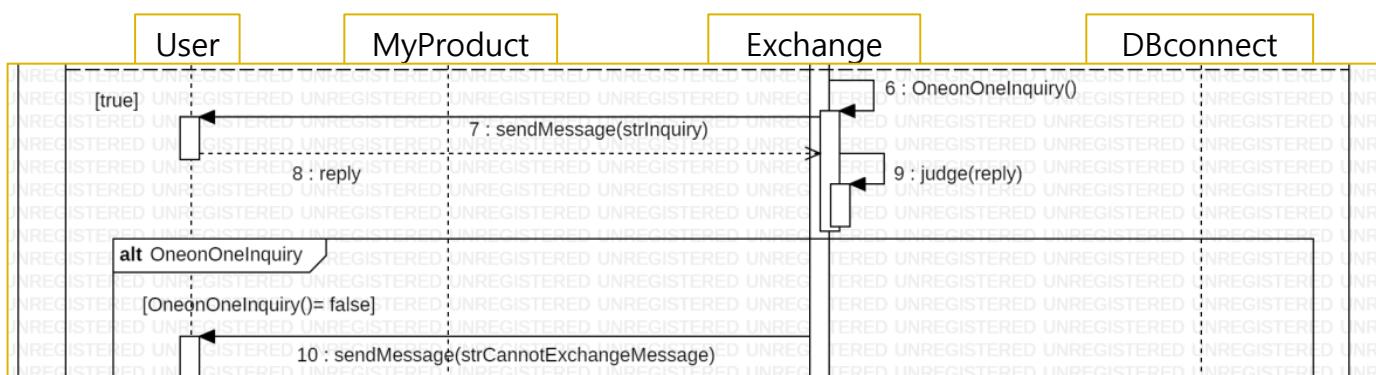
C7. Exchange – oneonOneInquiry()

```
public boolean oneonOneInquiry() { //U7.SeqDiagram : 6  
    7 this.sendMessage(this.strInquiry); //U7.SeqDiagram : 7  
    try (Scanner input = new Scanner(System.in)) {  
        8 String reply = input.nextLine(); //U7.SeqDiagram : 8  
        9 return this.judge(reply); //U7.SeqDiagram : 9  
    }  
}
```

[7]
1대1 문의 입력 요청

[8]
1대1 문의 입력
*예시 : scanner 입력

U7. Exchange Diagram : 6 ~ 10



[9]
입력된 문의를 검증
*judge()의 반환 : boolean

[10]
Data가 false면
교환불가 메시지 전송

02 | 코드 설명 -Exchange

C7. Exchange – requestExchange(int nPaymentCode)

```
else { // oneonOneInquiry() == true  
11 this.sendMessage(strRetrieveRequestMessage); //U7.SeqDiagram : 11  
12 this.sendProduct(nPaymentCode); //U7.SeqDiagram : 12      13  
14 this.bRetrieveProduct = this.retrieveProduct(nPaymentCode); //U7.SeqDiagram : 13, 14  
15 this.sendMessage(strCompleteMessage); //U7.SeqDiagram : 15  
}  
}
```

[11] 상품 반환 메시지 전송

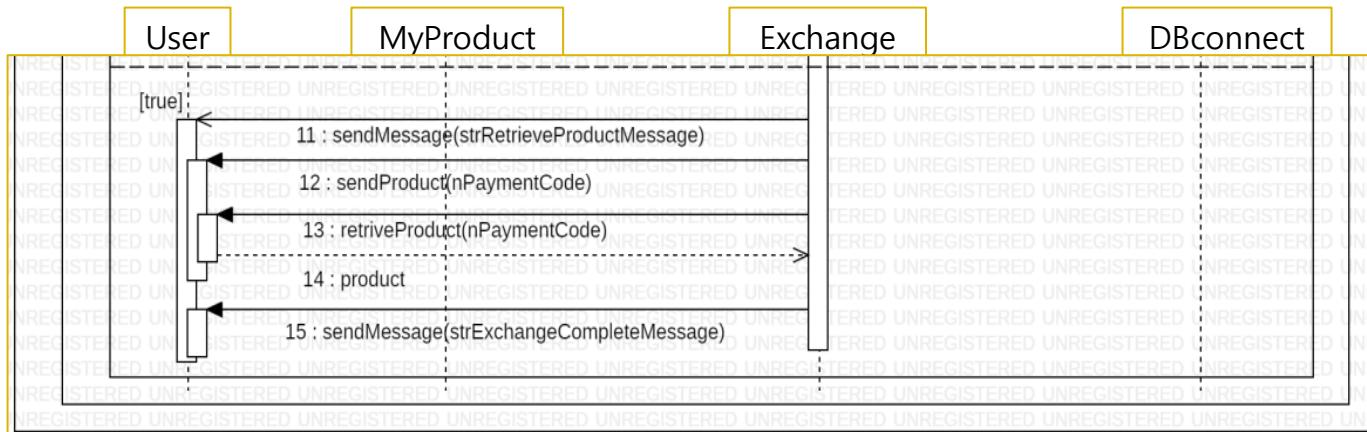
[12] 새 제품 발송

[13] 제품 회수

[14] 회수 여부 저장
회수 : boolean

[15] 교환 절차 종료 메시지 전송

U7. Exchange Diagram : 11 ~ 15



02 | 코드 설명 -Refund

C0. Main – MyProduct()

```
else if(click == "REFUND") { 1 //U3.SeqDiagram : 15 , U8.SeqDiagram : 1
    int nPaymentCode = menu.nextInt();
    myProduct.refund.requestRefund(nPaymentCode); 2//U3.SeqDiagram : 15, U8.SeqDiagram : 2
}
```

[1]

사용자가 환불을 요청하면
MyProduct()에서 환불을 실
행

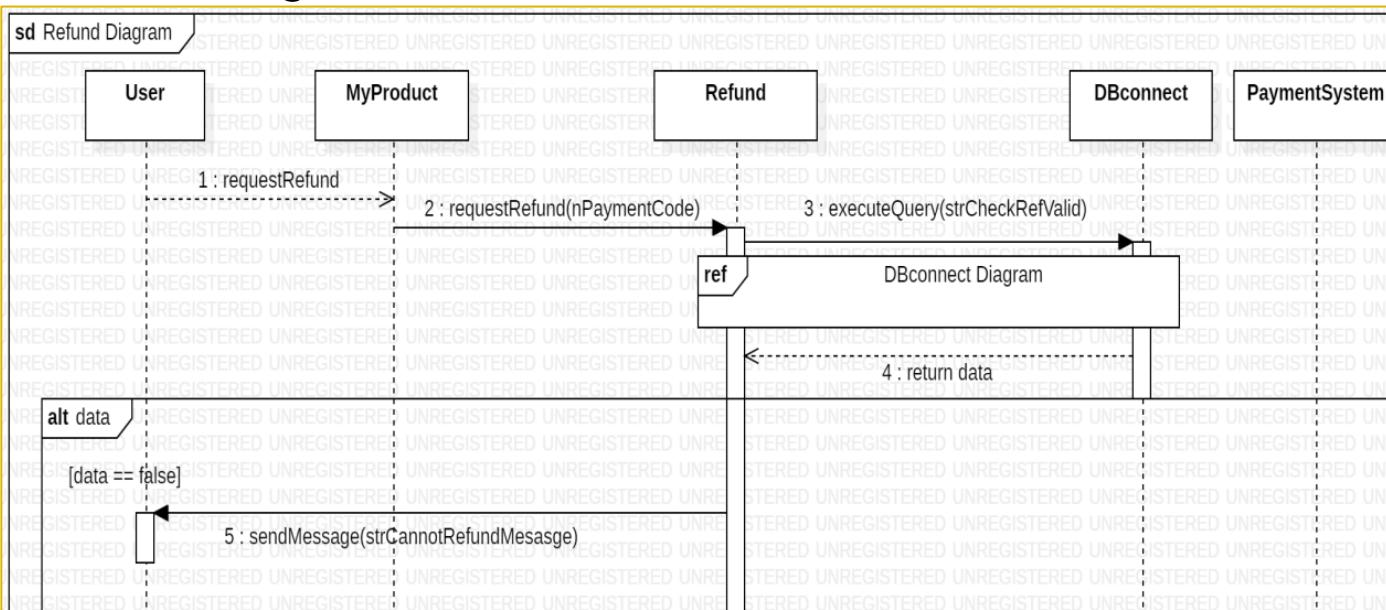
C8. Refund – requestRefund(int nPaymentCode)

```
public void requestRefund(int nPaymentCode) { //U8.SeqDiagram : 2
    4 this.strData = dbConnect.executeQuery(this.strCheckRefValid); //U8.SeqDiagram : 3, 4
    3
    if(this.strData.contains("false")) { //U8.SeqDiagram : 5
        5 this.sendMessage(strCannotRefundMessage); //U8.SeqDiagram : 5
    }
}
```

[2]

교환 실행
requestRefund()
*인자 nPaymentCode가
존재하지 않아 입력 대체

U8. Refund Diagram : 1 ~ 5



[3]

유효성 쿼리 실행

[4]

쿼리 결과 반환

[5]

data가 false면
환불 불가 메시지 전송

02 | 코드 설명 -Refund

C8. Refund – requestRefund(int nPaymentCode)

```

else {
    this.requestRefundToPaymentSystem(nPaymentCode); 6 //U8.SeqDiagram : 6, 7,
    // error : 7 is void but it has return
    8 this.sendMessage(strRetrieveRequestMessage); //U8.SeqDiagram : 8
    10 this.bRetrieveProduct = this.retrieveProduct(nPaymentCode); //U8.SeqDiagram : 9, 10
    11 this.sendMessage(strCompleteMessage); //U8.SeqDiagram : 11 9
}

```

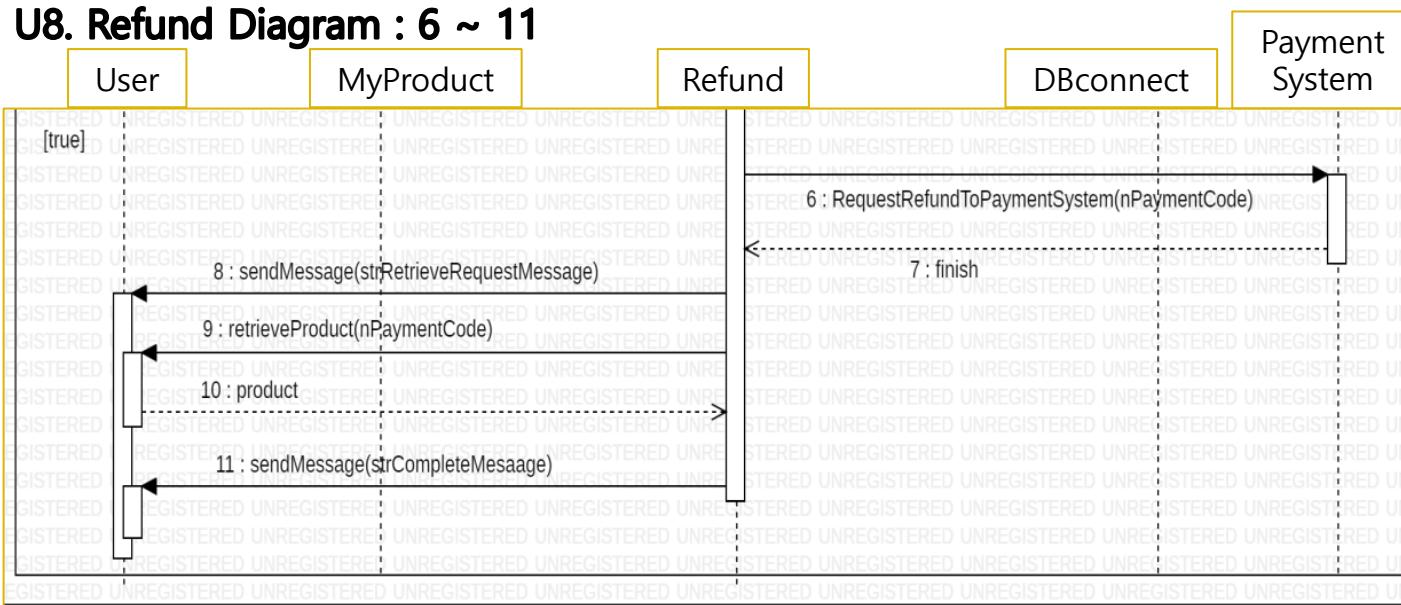
[7]
void형 메서드의 반환

[6]
외부 시스템에
환불 요청 - void

[8]
상품 반환 요청 메시지

[9]
제품 회수

U8. Refund Diagram : 6 ~ 11



[10]
제품 회수 여부 저장
회수 : boolean

[11]
환불 절차 종료 메시지 전송

03 | 오류수정 및 추가

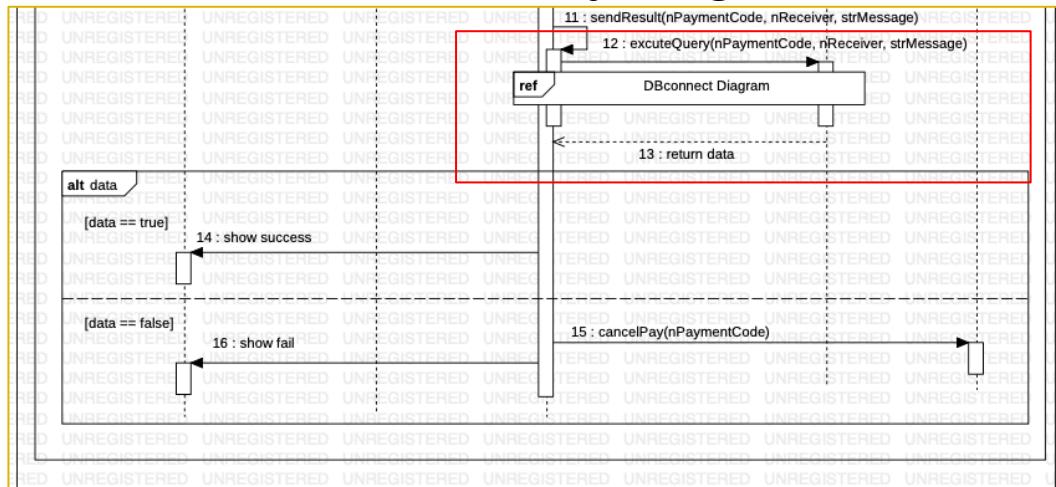
유스케이스 명세서 U5. 결제

▣ 이벤트 흐름

○ 정상 흐름

1. 상품 코드를 상세보기에서 받아온다.
2. DB로부터 받아온 정보를 표시한다.
3. 받는 사람을 선택한다.
4. 수량을 입력한다.
5. 메시지카드를 입력한다.
6. 쇼핑 포인트를 입력한다.
7. 최종 결제 금액을 표시한다.
8. 결제 방식을 요청한다.
9. 최종 결제가 승인 되었는지 확인한다.
10. 받는 사람에게 메세지를 전송한다.
11. "상품이 정상적으로 결제 되었습니다." 메시지를 보여준다.

시퀀스 다이어그램 C5. Pay Diagram



```

private boolean sendResult(int nPaymentCode, int nReceiver, String strMessage) throws ParseException { //U5.SeqDiagram : 11
    String queryPaymentCodeReceiverMessage = "query" + Integer.toString(nPaymentCode) + nReceiver + strMessage;
    JSONObject j0bj = (JSONObject)this.jParse.parse(this.dbConnect.executeQuery(queryPaymentCodeReceiverMessage)); //U5.SeqDiagram : 12, 13

    return (Boolean)j0bj.get("sendResult");
}

```

- 정상흐름 10번 이후 DB연결을 하여 결제 여부를 받아오는 흐름이 없음
- 시퀀스 다이어그램에는 있음

=> DB연결하여 결과값 받아오는 과정 추가함

03 | 오류수정 및 추가

C5. Payment

```
class Payment{ //class 5
    private Scanner scanner = new Scanner(System.in);
    private JSONParser jParse = new JSONParser();
    private DBConnect dbConnect = new DBConnect();
    private PaymentInfo paymentinfo = new PaymentInfo();
    private String strMessage;
    private String strData;
    private int nProductCode;
    private int nQuantity;
    private int nUsedPoint;
    private int nTotalPrice;
    private int nPaymentMethod;
    private int nPaymentCode = 0;
    private int nReceiver;
```

```
private JSONParser jParse = new JSONParser();  
->속성 추가(dataParser() 메서드에서 사용하기 위함)
```

03 | 오류수정 및 추가

C5. Payment

```
class DoPay{ //class 6
    protected int nTotalPrice;
    protected int nProductCode;

    protected int requestPay(int nTotalPrice, int nProductCode) {
        if(Success) {
            return nPaymentCode_FromExternalSystem;
        }
        else {
            return 0;
        }
    }
    public DoPay(int nTotalPrice, int nProductCode) {
        this.nTotalPrice = nTotalPrice;
        this.nProductCode = nProductCode;
    }
}
```

[JAVA 이클립스 기준]

Protected 혹은 Public

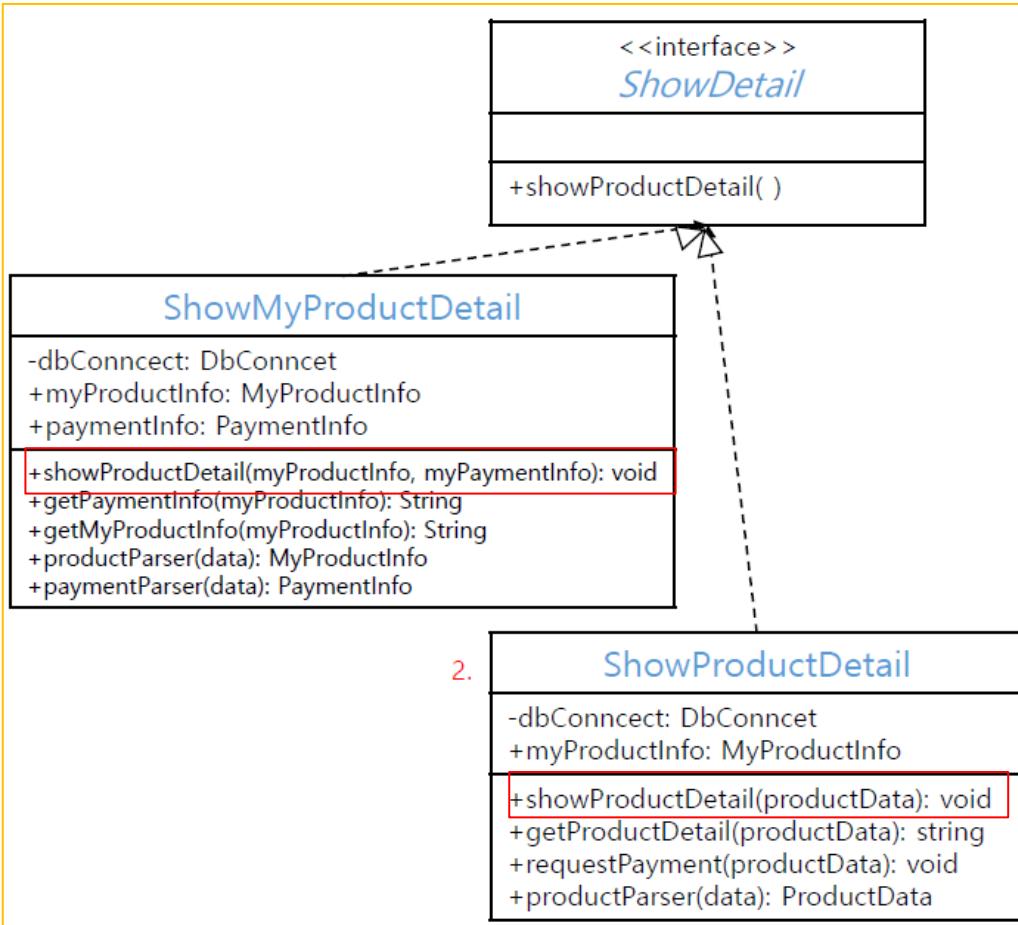
requestPay()의 접근 지정자가 같지 않아 오류 발생

자식 클래스의 requestPay()의 접근 지정자를 부모와 통일

```
class KakaoPay extends DoPay{ //class 6
    public KakaoPay(int nTotalPrice, int nProductCode) {
        super(nTotalPrice, nProductCode);
        // TODO Auto-generated constructor stub
    }
    protected int requestPay(int nTotalPrice, int nProductCode) { //U6.SeqDiagram : 5,
        boolean Success;
        if(Success) {
            return nPaymentCode_FromExternalSystem; //U6.SeqDiagram : 6, 11
        }
        else {
            return 0; //U6.SeqDiagram : 6, 14
        }
    }
}
```

03 | 오류수정 및 추가

C4. ShowMyProductDetail



인터페이스 제거

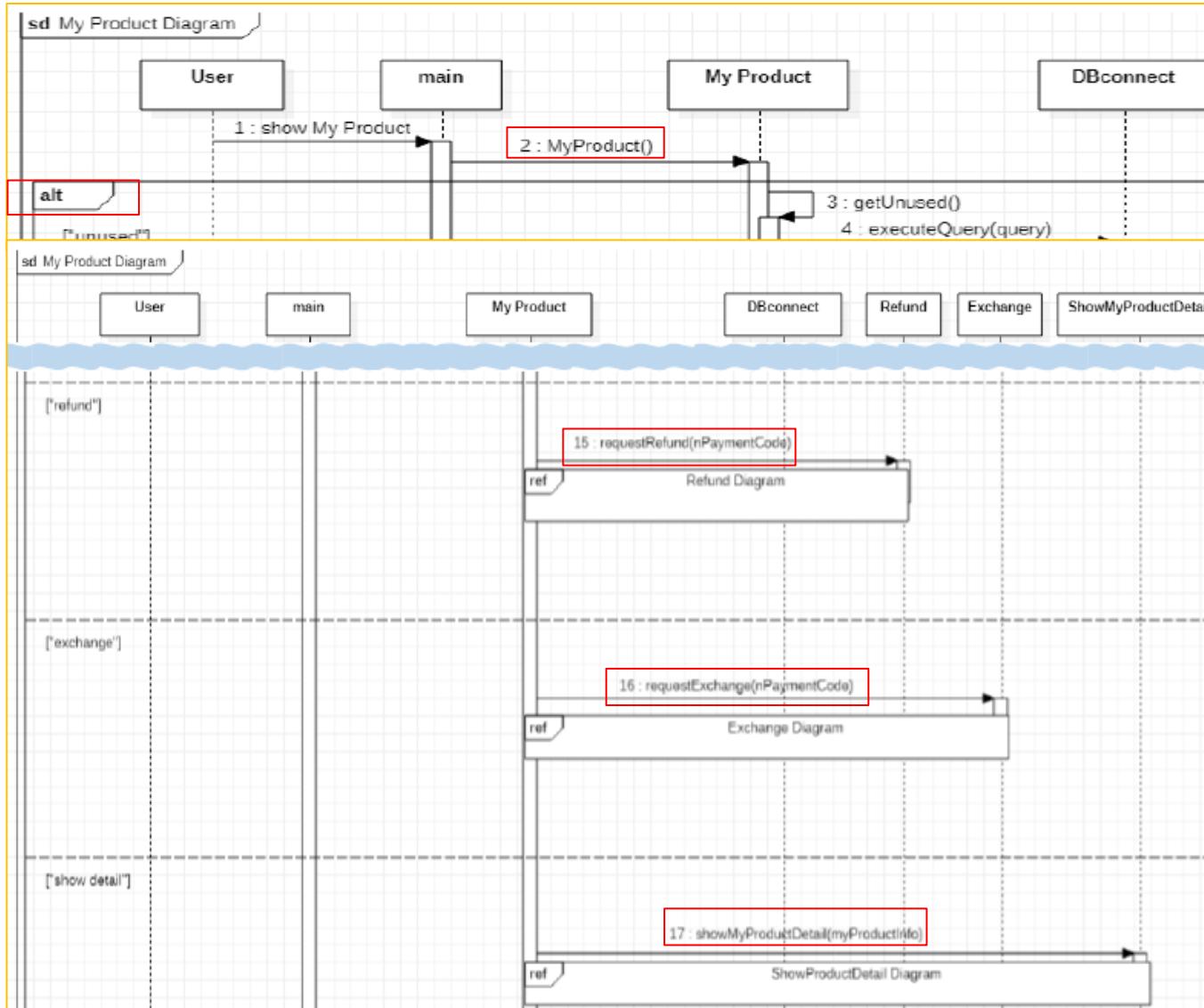
이클립스 기준

인터페이스 메서드를 구현할 때
인자의 차이로 오류 발생

```
/*
interface ShowDetail{ //class 4 interface
    public void showProductDetail();
}
*/
```

03 | 오류수정 및 추가

C3. MyProduct



다음 페이지..

03 | 오류수정 및 추가

C0. Main – MyProduct()

```
else if(click == "REFUND") { //U3.SeqDiagram : 15 , U8.SeqDiagram : 1
    int nPaymentCode = menu.nextInt();
    myProduct.refund.requestRefund(nPaymentCode); //U3.SeqDiagram : 15, U8.SeqDiagram : 2
}

else if(click == "EXCHANGE") { //U3.SeqDiagram : 16, U7.SeqDiagram : 1
    int nPaymentCode = menu.nextInt();
    myProduct.exchange.requestExchange(nPaymentCode); //U3.SeqDiagram : 16, U7.SeqDiagram : 2
}

else if(click == "SHOW_DETAILS") { //U3.SeqDiagram : 17
    int nPaymentCode = menu.nextInt();

    myProduct.strData = myProduct.dbConnect.executeQuery("__Query_ShowDetail_include_myProductInfo__");
    if(myProduct.strData.contains("ERROR")) { //add : U3.SeqDiagram :17_1
        myProduct.showErrorResult(myProduct.strData); //add : U3.SeqDiagram :17_1
    }
    else { //add : U3.SeqDiagram :15_1
        myProduct.dataIndexing((ArrayList<MyProductInfo>)myProduct.dataParser(myProduct.strData));

        for(int nIndex = 0 ; nIndex < myProduct.myProductInfo.size();nIndex++) { //add : U3.SeqDiagram :15_2
            if(nPaymentCode == myProduct.myProductInfo.get(nIndex).nPaymentCode) { //add : U3.SeqDiagram :15_3
                myProduct.showMyProductDetail(myProduct.myProductInfo.get(nPaymentCode)); //U3.SeqDiagram :15_4
            }
        }
    }
}
```

전달 인자의 값이 없음

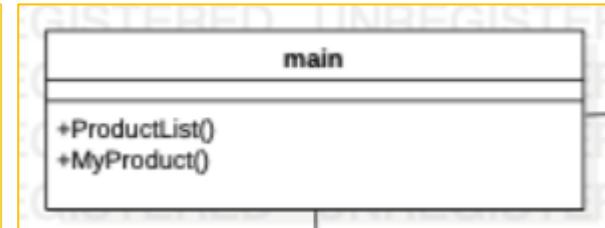
->DB 조회를 통해 목록을 받아오는 절차 추가

03 | 오류수정 및 추가

C0.Main - main()

```
public class Main { //class main
    public static void main() throws ParseException{ // add : main
        try (Scanner menu = new Scanner(System.in)) {
            int click = menu.nextInt();

            if(click == 1) { //U1.SeqDiagram : 1
                ProductList(); //U1.SeqDiagram : 1
            }
        }
    }
}
```

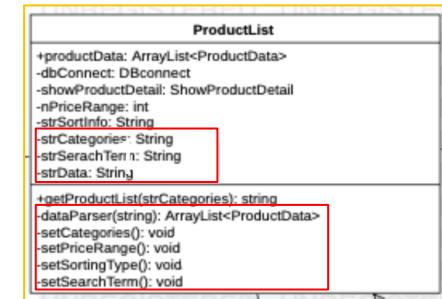


main() 메서드 추가

C1.ProductList

```
535⊕ public void setCategories() { //U1.SeqDiagram : 4, private -> public
536     try (Scanner scanner = new Scanner(System.in)) {
537         System.out.print("input category ");
538         this.Categories = scanner.nextLine();
539     }
540 }

553⊕ public void setSearchTerm() { //U1.SeqDiagram : 6, private -> public
554     try (Scanner scanner = new Scanner(System.in)) {
555         System.out.print("input searchTerm");
556         this.SearchTerm = scanner.nextLine();
557     }
558 }
```



접근 지정자 변경
private -> public

다수의 접근 지정자 표기 오류 [p10,p12,p14,p23]

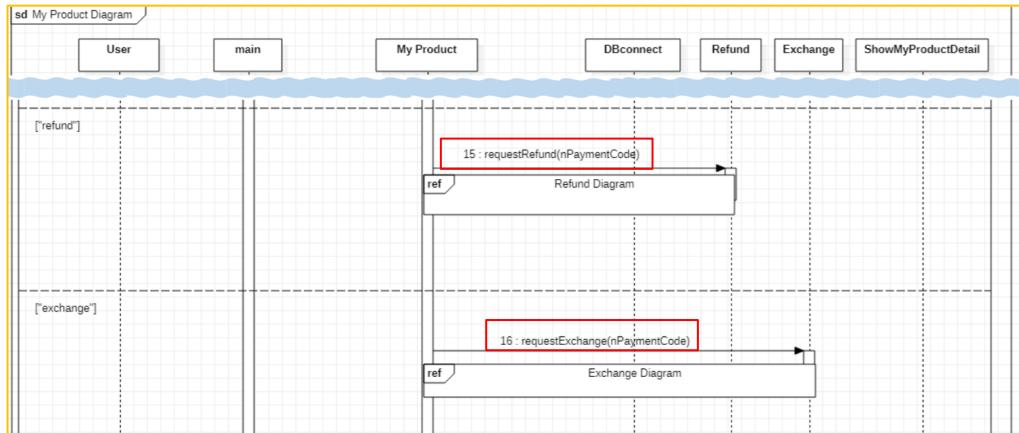
03 | 오류수정 및 추가

C0.Main – MyProduct()

```
else if(click == "REFUND") {    //U3.SeqDiagram : 15 , U8.SeqDiagram : 1
    int nPaymentCode = menu.nextInt();
    myProduct.refund.requestRefund(nPaymentCode);    //U3.SeqDiagram : 15, U8.SeqDiagram : 2
}

else if(click == "EXCHANGE") {  //U3.SeqDiagram : 16, U7.SeqDiagram : 1
    int nPaymentCode = menu.nextInt();
    myProduct.exchange.requestExchange(nPaymentCode);    //U3.SeqDiagram : 16, U7.SeqDiagram : 2
}
```

* 인자의 출처 불분명
-> 입력으로 대체



15. 만약 환불 메뉴를 선택 한다면, requestRefund 메서드를 호출하여 환불 페이지로 이동한다.
16. 만약 교환 메뉴를 선택 한다면, requestExchange 메서드를 호출하여 교환 페이지로 이동한다.

03 | 오류수정 및 추가

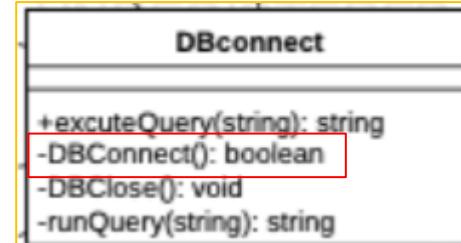
C2.DBConnect – executeQuery(String query)

```
class DBConnect{ //class 2
    public String executeQuery(String query) { //U2.SeqDiagram : 1
        boolean bConnectResult = this.DBconnect(); //U2.SeqDiagram : 2

        if(bConnectResult == false) { //U2.SeqDiagram : 3
            return "ERROR"; //U2.SeqDiagram : 4
        }
    }
}
```

*클래스와 메서드 명칭이 동일

- > 생성자 경고
- > 이름 수정



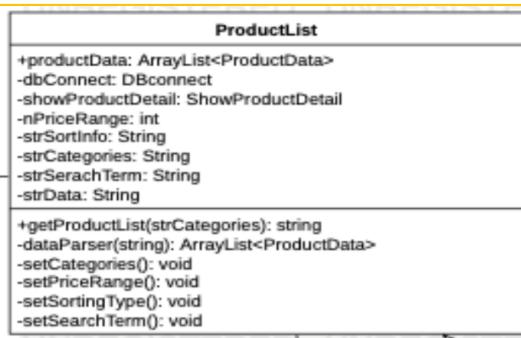
C0.Main – ProductList()

```
productList.showProductList(productList.productDataList);
//U1.SeqDiagram : 16, jump to C1.ProductList
if(productList.nProduct != 0) { //U1.SeqDiagram : 18
    productList.showProductDetail(productList.productDataList.get(productList.nProduct));
    //U1.SeqDiagram : 18,
} //jump to (U4) C4.ShowMyProductDetail
}
}
```

*showProductList()
클래스 디어그램에는 없으나
시퀀스 디어그램에는 있음

시퀀스 디어그램 U1 설명부분

12. 만약 사용자가 가격대를 입력하면
13. setPriceRange()함수를 호출하여 변수를 초기화하고, 리스트를 초기화한다.
14. 만약 사용자가 정렬기준을 선택하면
15. setSortingType()함수를 호출하여 변수를 초기화하고, 리스트를 초기화한다.
16. showProductList()함수를 호출하여 사용자에게 리스트를 출력한다.
17. 만약 사용자가 한 개의 상품을 선택한다면
18. showProductDetail 함수를 실행한다.



03 | 오류수정 및 추가

C0.Main – ProductList()

```
productList.strData = productList.getProductList(); //U1.SeqDiagram : 7~9 , jump to C1.ProductList  
  
if(productList.strData.contains("error")) { //U1.SeqDiagram : 10  
    System.out.println("error page"); //U1.SeqDiagram : 10  
}  
else {  
    productList.dataIndexing(productList.dataParser(productList.strData)); //U1.SeqDiagram : 11
```

* dataIndexing() 추가

C0.Main – ProductList()

```
else {  
    productList.dataIndexing(productList.dataParser(productList.strData)); //U1.SeqDiagram : 11  
    click = menu.nextLine();  
    if(click == "PriceRange") { //U1.SeqDiagram : 12  
        productList.setPriceRange(); //U1.SeqDiagram : 13  
    }  
    else if(click == "setSortingType") { //U1.SeqDiagram : 14  
        productList.setSortingType(); //U1.SeqDiagram : 15  
    }
```

12. 만약 사용자가 가격대를 입력하면
13. setPriceRange()함수를 호출하여 변수를 초기화하고, 리스트를 초기화한다.
14. 만약 사용자가 정렬기준을 선택하면
15. setSortingType()함수를 호출하여 변수를 초기화하고, 리스트를 초기화한다.

유스케이스 명세서 U1.ProductList

- 선택 흐름
 - 원하는 정렬 기준을 선택하면(추천순, 가격낮은순, 가격 높은순, 할인율높은순, 할인율낮은순, 최근등록순) 그에 따라 리스트를 재정렬 후 정상흐름 3을 실행한다.

클래스 다이어그램에서 자료구조 객체상에서 표현된게 가격정보밖에 없다

시퀀스 다이어그램 U1 설명부분

03 | 오류수정 및 추가

C0.Main – MyProduct()

```
if(click == "UNUSED") {    //U3.SeqDiagram : 3  
    myProduct.strData = myProduct.getUnused(); //U3.SeqDiagram : 3, jump to C3.MyProduct  
    if(myProduct.strData.contains("ERROR")) { //U3.SeqDiagram : 6  
        6 myProduct.showErrorResult(myProduct.strData); //U3.SeqDiagram : 6  
    }  
    else { //U3.SeqDiagram : 7  
        myProduct.dataIndexing((ArrayList<MyProductInfo>)myProduct.dataParser(myProduct.strData));  
        //U3.SeqDiagram : 7  
        8 myProduct.showResult(myProduct.myProductInfo); //U3.SeqDiagram : 8  
    }  
}
```

시퀀스 다이어그램 U.3 설명부분

6. 만약 DB의 리턴값이 error 일 경우, showResult 메서드로 사용자에게 결과를 보여준다.
7. 아닐경우, DB의 리턴값을 dataParser 메서드로 분석한다.
8. showResult 메서드로 사용자에게 결과를 보여준다.

*showErrorResult()추가
6,8번이 같은 메서드 사용

C1.ProductList – showProductDetail(ProductoData productData)

```
public void showProductDetail(ProductoData productData) throws ParseException {  
    this.showProductDetail.strData = this.showProductDetail.getProductDetail(productDataList.get(this.nProduct));  
    //U4.SeqDiagram : 2, 3, jump to C4.ShowProductDetail  
  
    if(this.showProductDetail.strData.contains("error")) { //U4.SeqDiagram : 8  
        this.showProductDetail.showError(this.showProductDetail.strData); //U4.SeqDiagram : 8  
    }  
}
```

ProductList
+productData: ArrayList<ProductoData>
-dbConnect: DBconnect
-showProductDetail: ShowProductDetail
-nPriceRange: int
-strSortInfo: String
-strCategories: String
-strSearchTerm: String
-strData: String
+getProductList(strCategories): string
-dataParser(string): ArrayList<ProductoData>
-setCategories(): void
-setPriceRange(): void
-setSortingType(): void
-setSearchTerm(): void

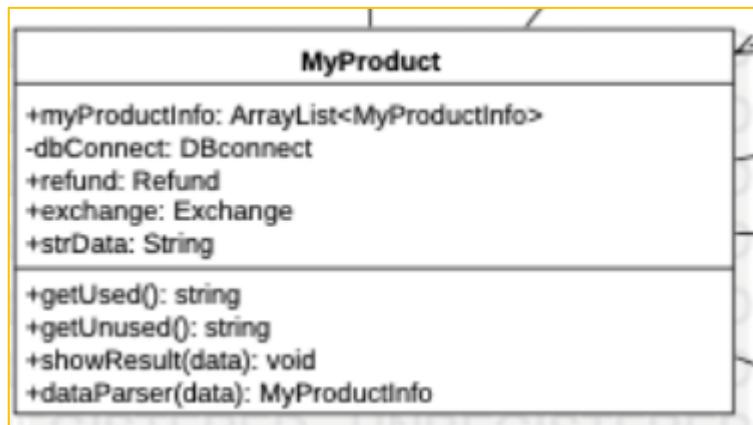
* showProductError() 추가

03 | 오류수정 및 추가

C3.MyProduct – showMyProductDetail(MyProductInfo myProductInfo)

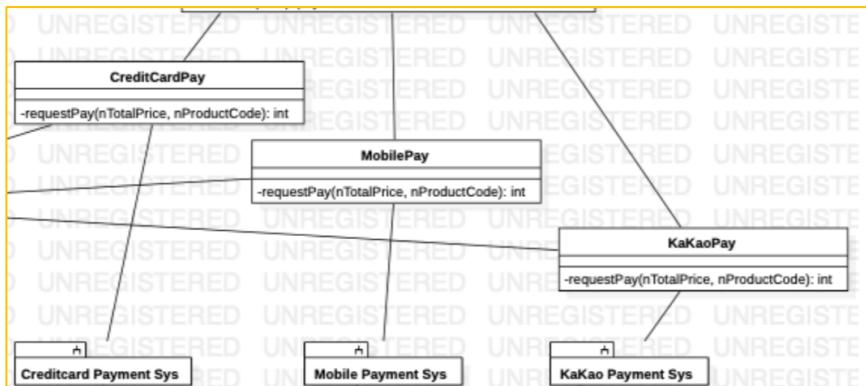
```
public void showMyProductDetail(MyProductInfo myProductInfo) throws ParseException {  
    //U4.Sequigram :16, add-error : not in class-diagram
```

* 클래스 다이어그램에 미 표기
>showMyProductDetail() 추가



시퀀스 다이어그램 U4 설명부분

15. MyProduct 객체에서 **showMyProductDetail** 객체를 생성한다.
16. **showMyProductDetail** 메서드를 호출하여 페이지를 이동한다.
17. 상품의 정보를 가져오기 위해



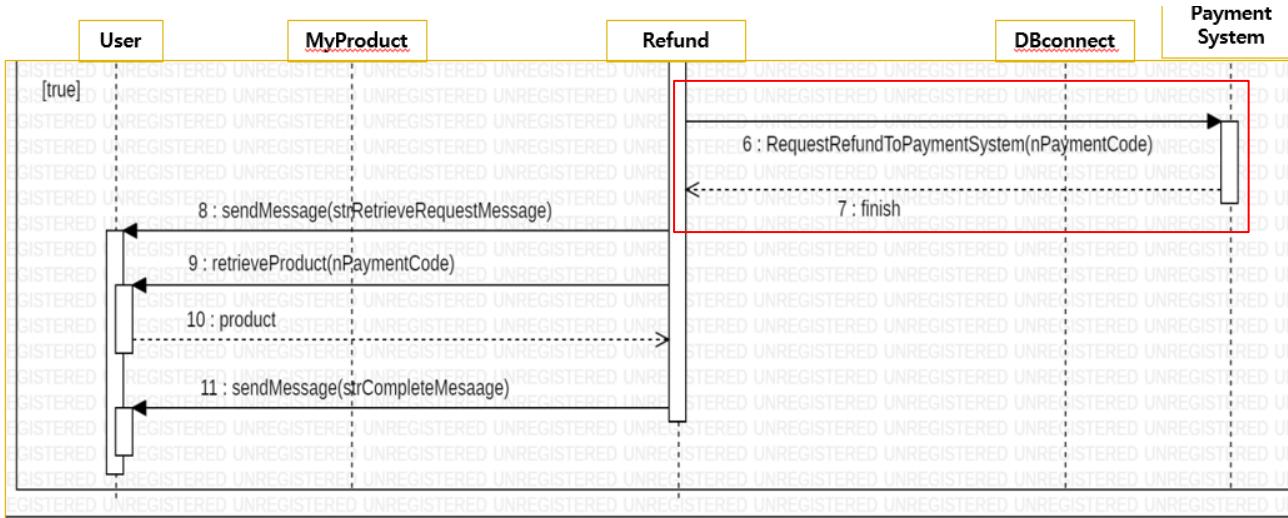
KakaoPay, MobilePay, CreditCardPay의 부모
DoPay

*클래스 다이어그램 관계선

일반화 관계

03 | 오류수정 및 추가

U8.Refund Diagram 6, 7



Refund

```

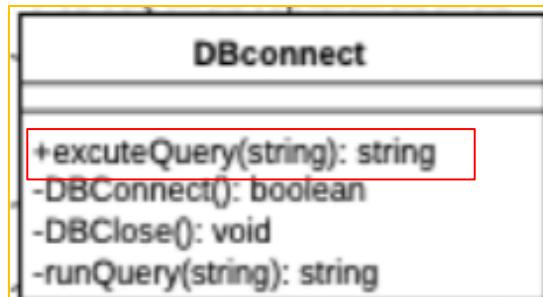
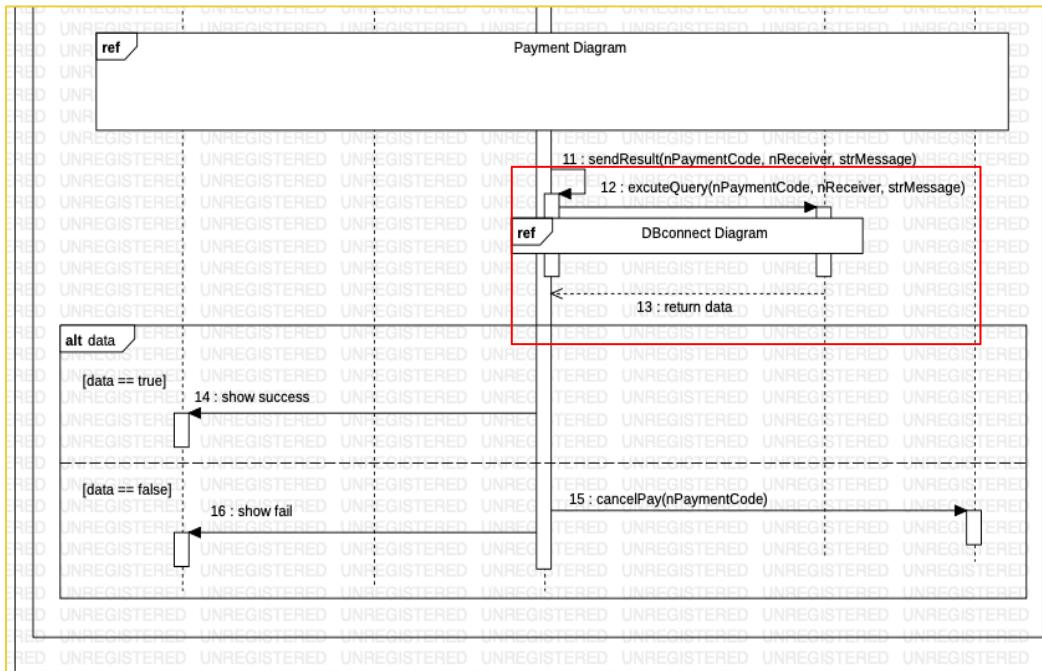
-strCompleteMessage: String
-strCannotRefundMessage: String
-strRetrieveRequestMessage: String
-strCheckRefVaild: String
-strData: String
-dbConnect: DBconnect

+requestRefund(nPaymentCode): void
+requestRefundToPaymentSystem(nPaymentCode): void
+retrieveProduct(nPaymentCode): boolean
+sendMessage(String): void
  
```

void형 메서드의 반환

03 | 오류수정 및 추가

U5.Pay Diagram 12, 13



`excuteQuery()` 메서드의 반환인자의 자료형이 String
-> 13. `return data`의 리턴 화살표의 출발지점이
12.`excuteQuery()`에 의한 활성화 바에 있지 않음

12.`excuteQuery()` 메서드의 전달인자는 1개(String)
-> U5의 시퀀스 다이어그램 12번에서는 3개의 전달인자를 포함함

03 | 오류수정 및 추가

U1.ProductList()

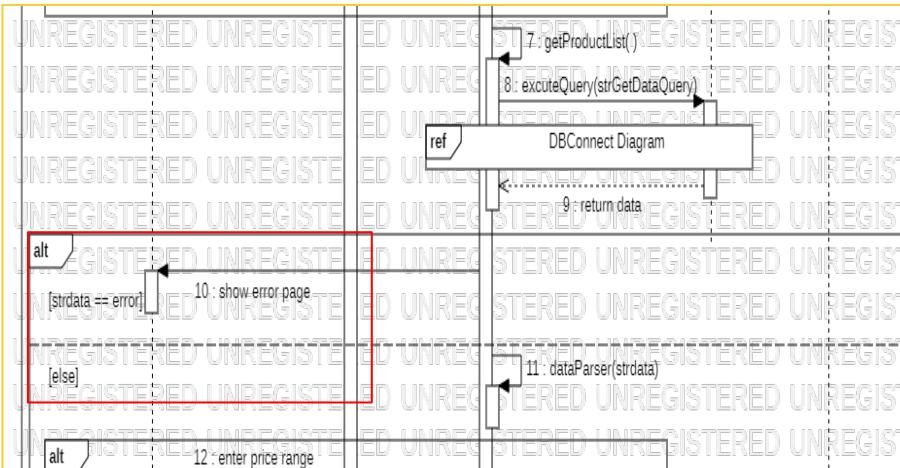
▣ 이벤트 흐름

○ 정상 흐름

1. 첫 화면에 접속한다.
2. DB연결 유스케이스로 원하는 정보(첫화면, 베스트, 종류, 검색어 등)를 쿼리문 형태로 보내고, 결과를 리스트 형태로 저장한다.
3. 결과를 화면에 보여준다.

○ 선택 흐름

- 상황(생일, 스몰럭셔리, 가벼운선물), 추천, 베스트, 브랜드, 검색 중 하나를 선택하면 정상흐름 2,3을 실행한다.
- 원하는 가격대를 지정하면 포함되지 않는 상품을 리스트에서 제외한 뒤 정상흐름 3을 실행한다.
- 원하는 정렬 기준을 선택하면(추천순, 가격낮은순, 가격높은순, 할인율높은순, 할인율낮은순, 최근등록순) 그에 따라 리스트를 재정렬 후 정상흐름 3을 실행한다.
- 조회할 한가지의 상품을 선택하면 상세정보 유스케이스로 이동한다.



정상흐름 2번의 DB연결에서 error가 반환되었을때의 선택흐름이 없음

시퀀스 다이어그램에는 포함되어 있음

시퀀스 다이어그램 U1. DB 연결 이후 부분

03 | 오류수정 및 추가

U2.DBConnect

▣ 이벤트 흐름

○ 정상 흐름

1. 데이터베이스에 연결한다.
2. 상품 목록과 선물함에서 보낸 쿼리문을 실행한다.
3. 쿼리의 결과를 리스트로 저장한다.
4. 데이터베이스 연결을 닫는다.
5. 호출한 유스케이스로 복귀하여 결과를 반환한다.

유스케이스 명세서 U2 정상흐름

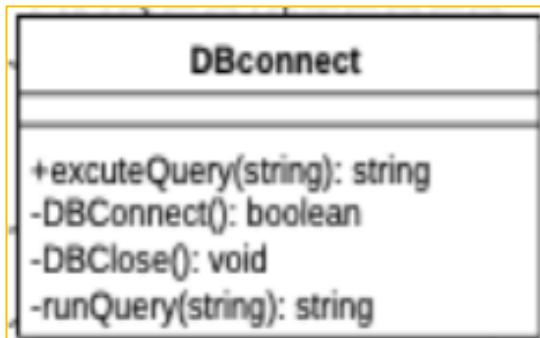
1. `DBconnect` 클래스에 접근하여 `executeQuery()` 메소드를 실행하면 다음과 같은 순서로 실행된다.
2. `DBConnect()` 함수를 호출하여 DB시스템에 연결한다.
3. 만약 DB시스템 연결에 실패할 경우
4. 예러를 반환한다.
5. 성공한다면, `runQuery()` 함수를 호출하여 DB에 쿼리문을 전송한다.
6. 쿼리문의 결과를 받아온다.
7. `DBClose()` 함수를 호출하여 DB시스템의 연결을 닫는다.
8. 결과를 `string` 형태로 반환한다.

시퀀스 다이어그램 U2 설명부분

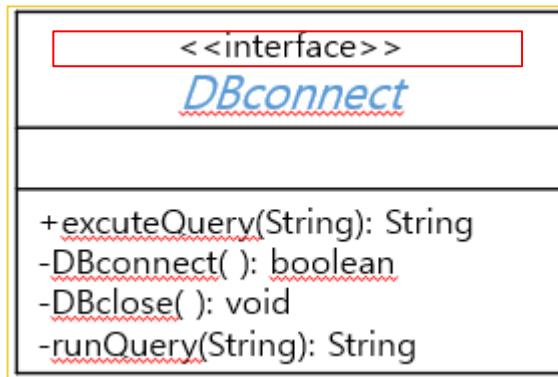
쿼리의 결과를 **리스트**로 저장한다고 되어있음
-> 시퀀스에서는 **String** 형태로 반환함

03 | 오류수정 및 추가

C2.DBConnect



전체 클래스 다이어그램 C2



클래스 다이어그램 C2 상세설명부분

클래스 다이어그램 상세설명 부분의 DBConnect 클래스에
<<interface>>부분이 잘못 표기 되어있음

03 | 오류수정 및 추가

C5.Payment

Payment

```
-creditCardPay: CreditCardPay  
-mobilePay: MobilePay  
-kakaoPay: KakaoPay  
-dbConnect: dbconnect  
-paymentinfo: PaymentInfo  
-myfriendslist: MyfriendsList  
-strMessage: string  
-strData: string  
-nProductCode: int  
-nQuantity: int  
-nUsedPoint: int  
-nTotalPrice: int  
-nPaymentMethod: int  
-nPaymentCode: int  
-nReceiver: int  
  
+makePayment(nProductCode): int  
-getQuantity(): int  
-getMessage(): string  
-getPaymentMethod(): int  
-useShoppingPoint(): int  
-calcTotalPrice(nProductPrice, nQuantity, nUsedPoint): int  
-sendResult(nPaymentCode, strReceiver, strMessage): boolean  
-cancelPay(nPaymentCode)  
-selectfriend(myfriendslist): int  
-dataPaser(data): paymentInfo
```

cancelPay() 메서드의 리턴 타입이
표기되어 있지 않음

03 | 오류수정 및 추가

U8.Refund

유스케이스 명세서 U8 정상흐름

▣ 이벤트 흐름

- 정상 흐름
 - 1. 회원으로부터 환불 요청을 받는다.
 - 2. 회원의 결제 정보를 확인, 유효성을 검사한다.
 - 3. 결제 정보를 토대로 해당 통장으로 금액을 환불 한다.
 - 4. 환불완료 메시지를 출력한다.
 - 5. 카카오톡으로 환불완료 메시지를 보낸다.

U8 유스케이스 명세서에는 **고객이 상품을 반환하는 흐름이 없음**

-> U8 시퀀스 다이어그램에는 포함되어있음

시퀀스 다이어그램 U8 설명부분

1. 유저가 환불을 요청한다.
2. 선물함에서 **환불**을 실행한다.
3. **DBconnect**의 메소드, 유효성 검사한다.
4. **string** 데이터를 **리턴받고**, 이는 "true" 또는 "false"이다.
5. **data == false** 일 경우 유저에게 잘못된 요청으로 환불이 불가능하다고 메세지를 보낸다.
6. **true** 일 경우 결제 시스템에 환불을 요청한다.
7. 외부 시스템에서 환불이 끝나면
8. 고객에게 상품 반환 요청 메세지를 보내고
9. 상품 반환 절차를 진행한다.
10. 고객이 상품을 돌려 보내면(택배 등으로 물건을 받을 경우) **Boolean** 변수를 **true**로 바꿔 주는 방식
11. 환불 절차가 끝났다는 메세지를 유저에게 보낸다.

03 | 오류수정 및 추가

C1.ProductList – dataParser(String data)

```
public ArrayList<ProductData> dataParser(String data){ //U1.SeqDiagram :11 , private->public
    ArrayList<ProductData> parsedData = new ArrayList<>();
    ProductData tmpData = new ProductData();

    /*Parse from JSON String*/

    try {
        JSONParser jParse = new JSONParser();
        JSONObject jObj = (JSONObject)jParse.parse(data);
        JSONArray jArray = (JSONArray)jObj.get("ProductDataList");

        for(int jArrIndex = 0 ; jArrIndex < jArray.size();jArrIndex++) {
            JSONObject productData = (JSONObject)jArray.get(jArrIndex);

            tmpData.strProductName = (String)productData.get("strProductName");
            tmpData.strProductImage = (String)productData.get("strProductImage");
            tmpData.strProductBrand = (String)productData.get("strProductBrand");
            tmpData.strProductDescription = (String)productData.get("strProductDescription");
            tmpData.nProductCode = (Integer)productData.get("nProductCode");
            tmpData.nProductPrice = (Integer)productData.get("nProductPrice");

            parsedData.add(tmpData);
        }
    } catch(ParseException e) { e.printStackTrace(); }
    tmpData = null;
    return parsedData;
}
```

JSON을 사용하기 위해 멤버변수
로 JSONParser를 추가로 선언

04 | 구현시 어려웠던 점

- 동기화가 잘 이루어지지 않아 자료형 자체가 다른 경우에서 구현의 어려움이 있었습니다.
- 시퀀스 다이어그램의 메시지의 송신 위치가 통일되어있지 않아 많은 고민을 하게 되었습니다.
- 이름이 동일하거나 유사한 메서드명, 오해할 수 있는 멤버 변수 명이 다수 존재해서 구현할 때 많은 수정이 있었습니다.
- 객체의 생성 지점이 실제 구현에서 불가능한 경우나 중복 생성하는 경우가 많아 임의의 조정이 필요했습니다

05 | Q & A
