



스타벅스 온라인 주문 프로그램

바비와 내성발톱들

Mini PRJ

1.

팀원소개

2.

개발환경

3.

프로그램 소개

4.

개발 목적

5.

구현기능

6.

시연



송예린

조장



장민정

DB 관리자



신은지

이슈 관리자



장종찬

일정 관리자



주선기

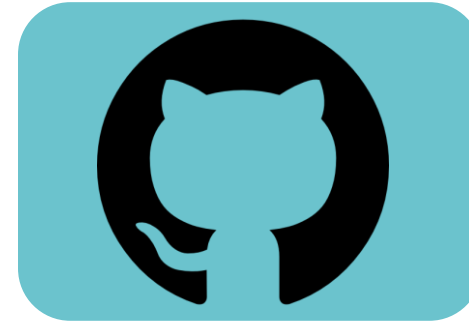
형상 관리자



Oracle DB



ERD Cloud



Git Hub



Eclipses



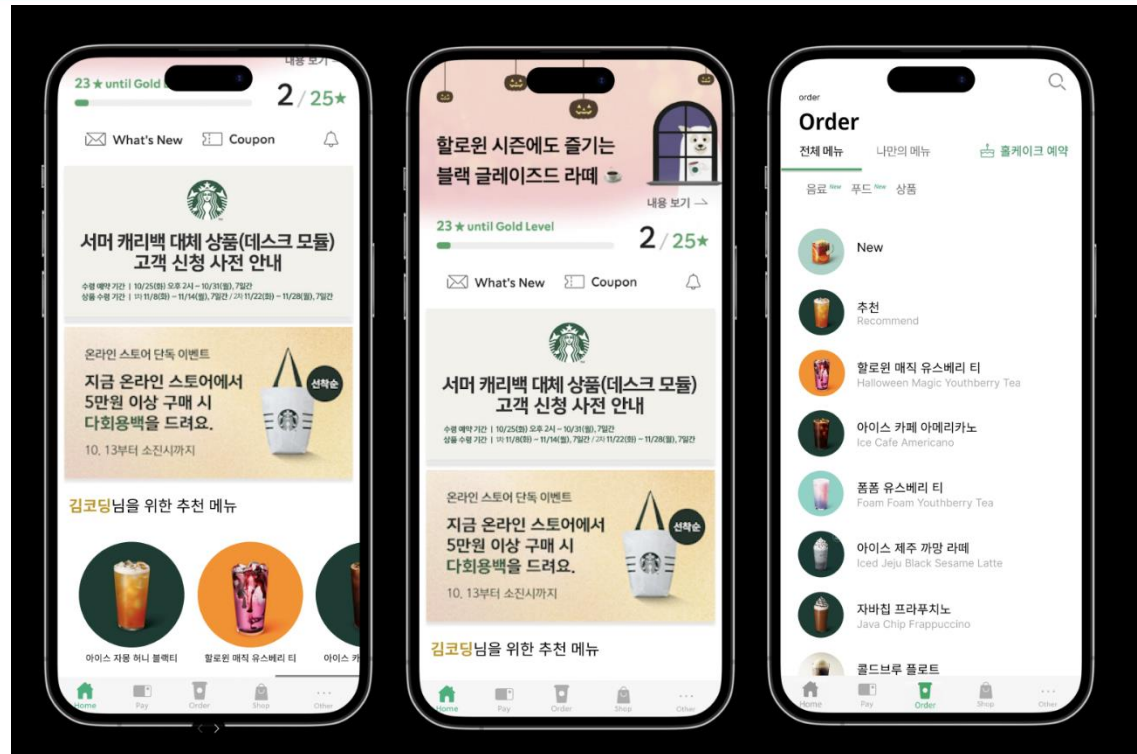
Source Tree



SQL Developer

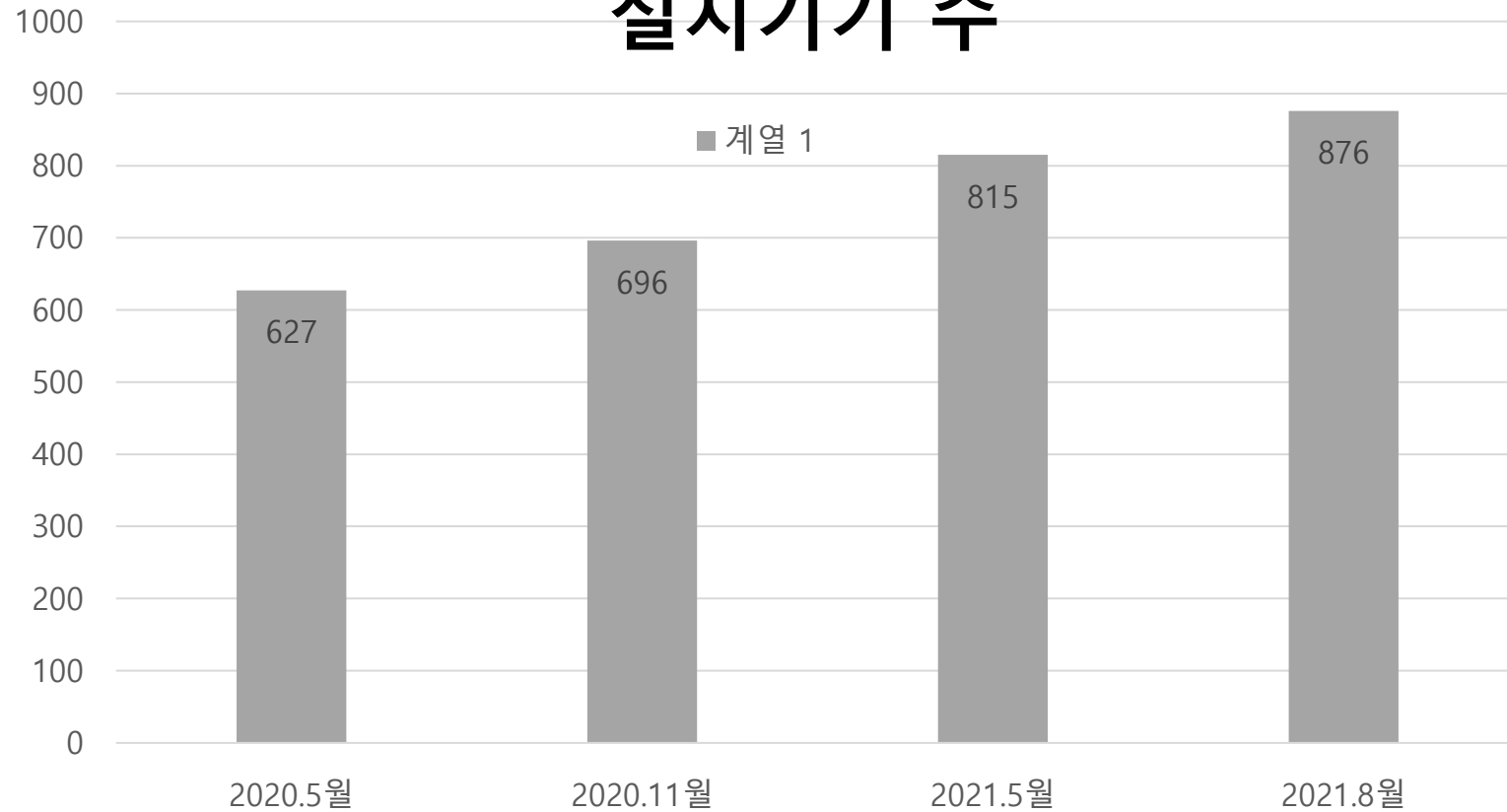


“스타벅스 온라인
주문 프로그램”

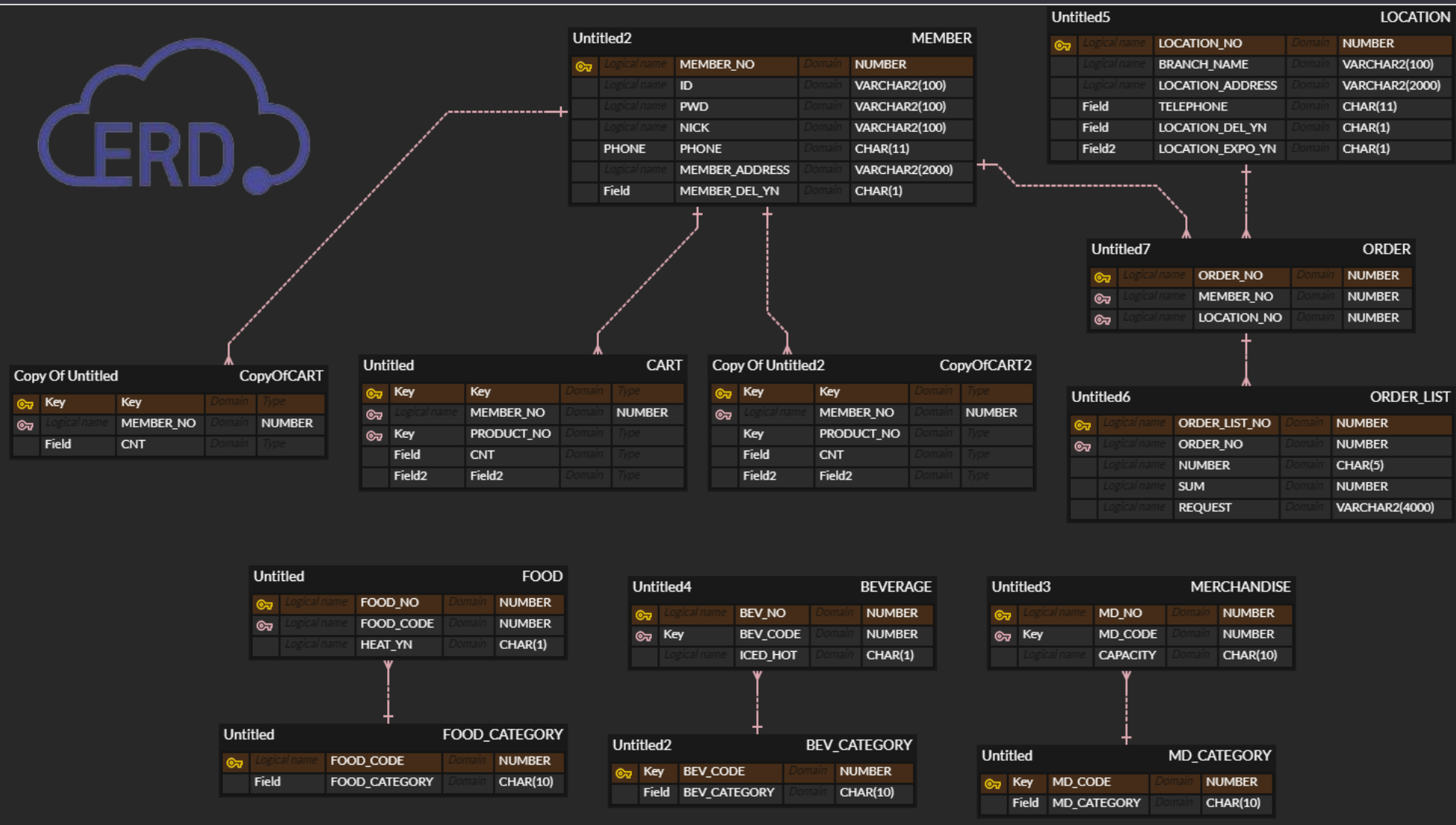


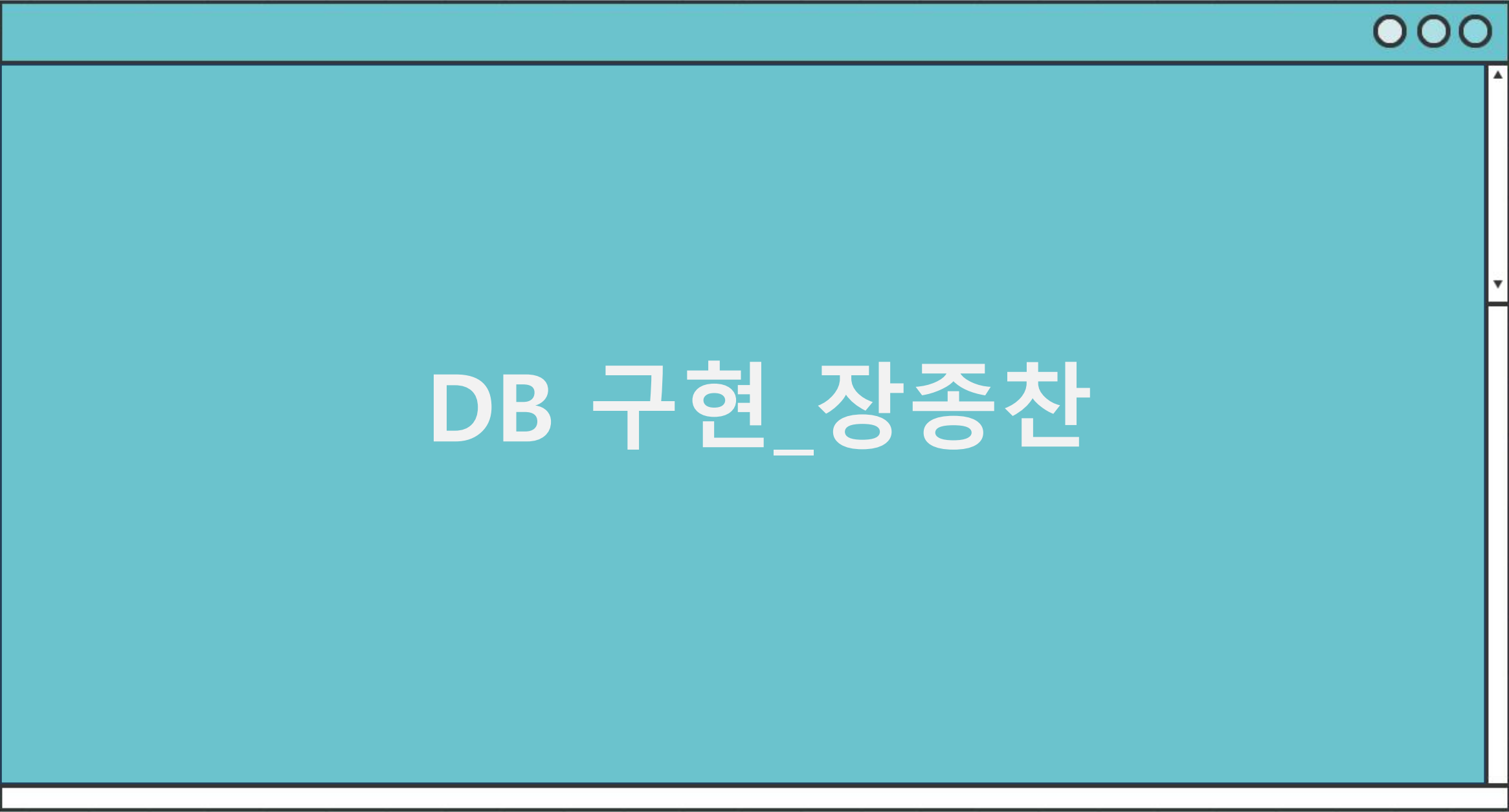
“스타벅스 온라인
주문 프로그램”

스타벅스 애플리케이션 설치기기 수



ERD Cloud





DB 구현_장종찬

시퀀스 생성

```
CREATE SEQUENCE SEQ_MEMBER_NO NOCACHE NOCYCLE;

CREATE SEQUENCE SEQ_LOCATION_NO NOCACHE NOCYCLE;

CREATE SEQUENCE SEQ_ORDER_SHOW_NO NOCACHE NOCYCLE;
CREATE SEQUENCE SEQ_ORDER_LIST_NO NOCACHE NOCYCLE;

CREATE SEQUENCE SEQ_MERCHANDISE_NO NOCACHE NOCYCLE;
CREATE SEQUENCE SEQ_MERCHANDISE_CART_NO NOCACHE NOCYCLE;

CREATE SEQUENCE SEQ_FOOD_NO NOCACHE NOCYCLE;
CREATE SEQUENCE SEQ_FOOD_CART_NO NOCACHE NOCYCLE;

CREATE SEQUENCE SEQ_BEVERAGE_NO NOCACHE NOCYCLE;
CREATE SEQUENCE SEQ_BEVERAGE_CART_NO NOCACHE NOCYCLE;
```

테이블 생성
: MEMBER, LOCATION

```
CREATE TABLE MEMBER (
    MEMBER_NO          NUMBER          CONSTRAINT PK_MEMBER_NO PRIMARY KEY
    , ID               VARCHAR2(100)    CONSTRAINT NN_MEMBER_ID NOT NULL CONSTRAINT UQ_MEMBER_ID UNIQUE
    , PWD              VARCHAR2(100)    CONSTRAINT NN_MEMBER_PWD NOT NULL
    , NICK              VARCHAR2(100)    CONSTRAINT NN_MEMBER_NICK NOT NULL
    , PHONE             CHAR(11)         CONSTRAINT NN_MEMBER_PHONE NOT NULL
    , MEMBER_ADDRESS    VARCHAR2(2000)   CONSTRAINT NN_MEMBER_MEMBER_ADDRESS NOT NULL
    , MEMBER_DEL_YN     CHAR(1)          DEFAULT 'N' CONSTRAINT CK_MEMBER_QUIT CHECK (MEMBER_DEL_YN IN ('Y','N'))
);

----- LOCATION -----
CREATE TABLE LOCATION (
    LOCATION_NO         NUMBER          CONSTRAINT PK_LOCATION_LOCATION_NO PRIMARY KEY
    , BRANCH_NAME        VARCHAR2(100)   CONSTRAINT NN_LOCATION_BRANCH_NAME NOT NULL CONSTRAINT UQ_LOCATION_BRANCH_NAME UNIQUE
    , LOCATION_ADDRESS    VARCHAR2(2000)  CONSTRAINT NN_LOCATION_LOCATION_ADDRESS NOT NULL
    , DISTANCE           CHAR(10)         CONSTRAINT NN_LOCATION_DISTANCE NOT NULL
    , TELEPHONE           CHAR(11)        CONSTRAINT NN_LOCATION_TELEPHONE NOT NULL
    , LOCATION_DEL_YN     CHAR(1)          DEFAULT 'N' CONSTRAINT CK_LOCATION_DEL_YN CHECK (LOCATION_DEL_YN IN ('Y','N'))
    , LOCATION_EXPO_YN    CHAR(1)          DEFAULT 'Y' CONSTRAINT CK_LOCATION_EXPO_YN CHECK (LOCATION_EXPO_YN IN ('Y','N'))
);
```

테이블 생성 : MERCHANDISE

```
----- MERCHANDISE -----  
INSERT INTO MERCHANDISE (  
    MD_NO  
    , MD_CODE  
    , MD_NAME  
    , MD_PRICE  
    , CAPACITY  
    , MD_DEL_YN  
    , MD_EXPO_YN  
    , MD_STOCK  
)
```

테이블 생성 : FOOD

```
----- FOOD -----  
CREATE TABLE FOOD (  
    FOOD_NO          NUMBER          CONSTRAINT PK_FOOD_NO PRIMARY KEY  
    , FOOD_NAME      VARCHAR2(100)    CONSTRAINT NN_FOOD_NAME NOT NULL CONSTRAINT UQ_FOOD_NAME UNIQUE  
    , FOOD_PRICE     NUMBER          CONSTRAINT NN_FOOD_PRICE NOT NULL  
    , HEAT_YN        CHAR(1)          DEFAULT 'N' CONSTRAINT CK_FOOD_HEAT CHECK (HEAT_YN IN('Y','N'))  
    , FOOD_DEL_YN     CHAR(1)          DEFAULT 'N' CONSTRAINT CK_FOOD_DELETE CHECK (FOOD_DEL_YN IN('Y','N'))  
    , FOOD_EXPO_YN    CHAR(1)          DEFAULT 'Y' CONSTRAINT CK_FOOD_EXPOSE CHECK (FOOD_EXPO_YN IN('Y','N'))  
    , FOOD_CODE       NUMBER          CONSTRAINT NN_FOOD_CODE NOT NULL  
    , FOOD_STOCK      NUMBER  
)  
  
CREATE TABLE FOOD_CATEGORY (  
    FOOD_CODE         NUMBER          CONSTRAINT PK_FOOD_CODE PRIMARY KEY  
    , FOOD_CATEGORY    VARCHAR2(255)   CONSTRAINT NN_FOOD_CATEGORY NOT NULL  
)  
  
CREATE TABLE FOOD_CART (  
    FOOD_CART_NO      NUMBER          CONSTRAINT PK_FOOD_CART_NO PRIMARY KEY  
    , MEMBER_NO        NUMBER          CONSTRAINT NN_MEMBER_NO_FOOD NOT NULL  
    , FOOD_NO          NUMBER          CONSTRAINT NN_FOOD_NO NOT NULL  
    , FOOD_COUNT       CHAR(5)         CONSTRAINT NN_FOOD_COUNT NOT NULL  
    , FOOD_SUM         NUMBER          CONSTRAINT NN_FOOD_SUM NOT NULL  
    , FOOD_REQUEST     VARCHAR2(4000)  NOT NULL  
)  
  
CREATE TABLE MD_CATEGORY (  
    MD_CODE           NUMBER          CONSTRAINT PK_MD_CATEGORY_MD_CODE PRIMARY KEY  
    , MD_CATEGORY      CHAR(10)        CONSTRAINT NN_MD_CATEGORY_MD_CATEGORY NOT NULL  
)
```

테이블 생성 : BEVERAGE

```
----- BEVERAGE -----
CREATE TABLE BEVERAGE (
    BEV_NO          NUMBER          CONSTRAINT PK_BEV_NO PRIMARY KEY
    , BEV_NAME      VARCHAR2(100)   CONSTRAINT NN_BEV_NAME NOT NULL UNIQUE
    , BEV_PRICE     NUMBER          CONSTRAINT NN_BEV_PRICE NOT NULL
    , ICED_HOT      CHAR(1)         DEFAULT 'I' CONSTRAINT CK_BEV_ICED CHECK (ICED_HOT IN('I','H'))
    , BEV_DEL_YN    CHAR(1)         DEFAULT 'N' CONSTRAINT CK_BEV_DELETE CHECK (BEV_DEL_YN IN('Y','N'))
    , BEV_EXPO_YN   CHAR(1)         DEFAULT 'Y' CONSTRAINT CK_BEV_EXPOSE CHECK (BEV_EXPO_YN IN('Y','N'))
    , BEV_CODE      NUMBER          CONSTRAINT NN_BEV_CODE NOT NULL
    , BEV_STOCK     NUMBER          NOT NULL
);

CREATE TABLE BEV_CATEGORY (
    BEV_CODE        NUMBER          CONSTRAINT PK_BEV_CODE PRIMARY KEY
    , BEV_CATEGORY   VARCHAR2(255)   CONSTRAINT NN_BEV_CATEGORY NOT NULL
);

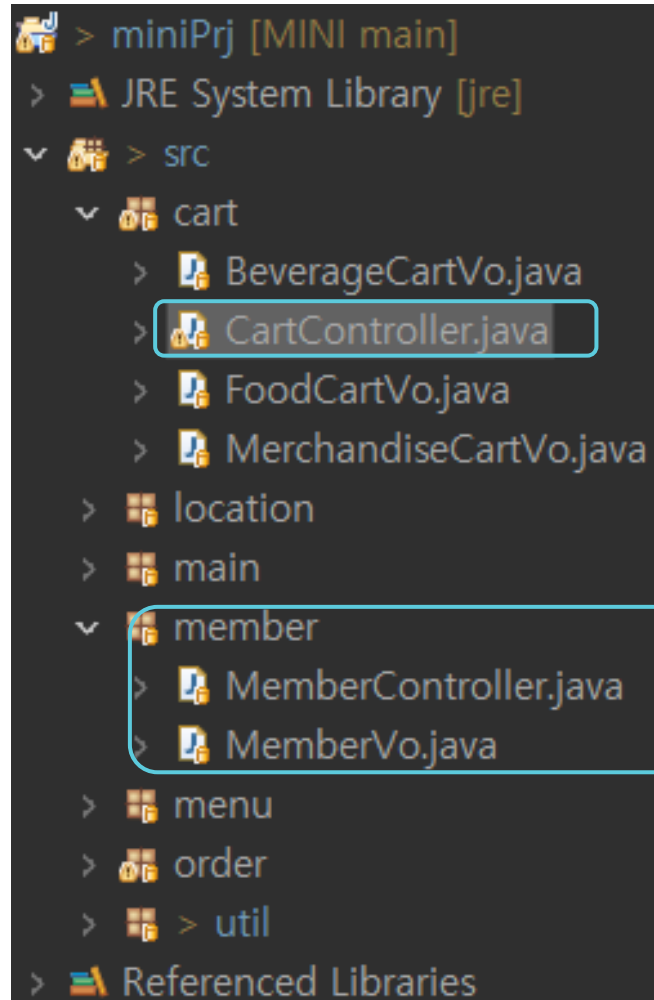
CREATE TABLE BEVERAGE_CART (
    BEVERAGE_CART_NO NUMBER  CONSTRAINT PK_BEVERAGE_CART_NO PRIMARY KEY
    , MEMBER_NO      NUMBER  CONSTRAINT NN_BEV_CART_MEMBER_NO NOT NULL
    , BEV_NO         NUMBER  CONSTRAINT NN_BEV_NO NOT NULL
    , BEV_COUNT      CHAR(5)  CONSTRAINT NN_BEV_COUNT NOT NULL
    , BEV_SUM        NUMBER  CONSTRAINT NN_BEV_SUM NOT NULL
    , BEV_REQUEST    VARCHAR2(4000)
);
```

테이블 생성 : ORDER

```
----- ORDER -----  
CREATE TABLE ORDER_SHOW (  
    ORDER_NO      NUMBER          CONSTRAINT PK_ORDER_SHOW_ORDER_NO PRIMARY KEY  
    , MEMBER_NO   NUMBER          CONSTRAINT NN_ORDER_SHOW_MEMBER_NO NOT NULL  
    , LOCATION_NO  NUMBER          CONSTRAINT NN_ORDER_SHOW_LOCATION_NO NOT NULL  
);  
  
CREATE TABLE ORDER_LIST (  
    ORDER_LIST_NO  NUMBER          CONSTRAINT PK_ORDER_LIST_ORDER_LIST_NO PRIMARY KEY  
    , ORDER_NO     NUMBER          CONSTRAINT NN_ORDER_LIST_ORDER_NO NOT NULL  
);
```



JAVA구현_장민정



```
> miniPrj [MINI main]
> JRE System Library [jre]
v > src
  v > cart
    > BeverageCartVo.java
    > CartController.java
    > FoodCartVo.java
    > MerchandiseCartVo.java
  > location
  > main
  v > member
    > MemberController.java
    > MemberVo.java
  > menu
  > order
  > util
> Referenced Libraries
```

Member

- printMenu

1. Join
2. Login

- printMenuAfterLogin

1. changePwd
2. changeMemberAddress

3. changePhone

4. Logout

5. Quit

Cart

CartController


```
public void printMenu() throws Exception {
    System.out.println("===== MEMBER =====");
    System.out.println("0. 이전 메뉴로 돌아가기");
    System.out.println("1. 로그인");
    System.out.println("2. 회원가입");
    System.out.print("메뉴 번호 입력: ");
    String num = JdbcTemplate.SC.nextLine();
    switch (num) {
        case "1":
            login();
            break;
        case "2":
            join();
            break;
        case "0":
            System.out.println("이전 메뉴로 돌아갑니다.");
            return;
        default:
            System.out.println("잘못 입력하셨습니다.");
    }
} // printMenu method
```

Code

printMenu
회원 관련 메뉴 출력
메서드

```
===== MINI PROJECT =====
0. 종료
1. 회원
2. 매장
3. 주문
메뉴 번호 입력: 1
===== MEMBER =====
0. 이전 메뉴로 돌아가기
1. 로그인
2. 회원가입
메뉴 번호 입력: |
```

```
public void printMenuAfterLogin() throws Exception {  
    System.out.println("0. 뒤로가기");  
    System.out.println("1. 비밀번호 변경");  
    System.out.println("2. 주소 변경");  
    System.out.println("3. 전화번호 변경");  
    System.out.println("4. 장바구니");  
    System.out.println("5. 로그아웃");  
    System.out.println("6. 회원탈퇴");  
  
    System.out.print("메뉴 번호 선택: ");  
    String num = JDBCTemplate.SC.nextLine();  
    switch (num) {  
        case "0":  
            return ;  
        case "1":  
            changePwd();  
            break;  
        case "2":  
            changeMemberAddress();  
            break;  
        case "3":  
            changePhone();  
            break;  
        case "4":  
            showCart();  
            break;  
        case "5":  
            logout();  
            break;  
        case "6":  
            quit();  
    }  
}
```

Code

printMenuAfterLogin 로그인 이후 메뉴 출력 메서드

```
메뉴 번호 입력: 1  
아이디: system  
비밀번호: 1234  
관리자님 환영합니다.  
===== MINI PROJECT =====  
0. 종료  
1. 회원  
2. 매장  
3. 주문  
메뉴 번호 입력: 1  
0. 뒤로가기  
1. 비밀번호 변경  
2. 주소 변경  
3. 전화번호 변경  
4. 장바구니  
5. 로그아웃  
6. 회원탈퇴  
메뉴 번호 선택:
```

```

public void emptyCart() throws Exception {
    Connection conn = JdbcTemplate.getConnection();

    // Bev 카트 비우기
    String sql1 = "DELETE FROM BEVERAGE_CART";
    PreparedStatement pstmt1 = conn.prepareStatement(sql1);
    int result1 = pstmt1.executeUpdate();
    if(result1 == 0) {
        System.out.println("음료 장바구니 비우기 실패 ...");
        return ;
    }

    // 시퀀스 삭제
    String sql2 = "DROP SEQUENCE SEQ_BEVERAGE_CART_NO";
    PreparedStatement pstmt2 = conn.prepareStatement(sql2);
    boolean result2 = pstmt2.execute();
    if(result2) {
        System.out.println("음료 카트 시퀀스 지우기 실패 ...");
        return ;
    }

    // 시퀀스 생성
    String sql3 = "CREATE SEQUENCE SEQ_BEVERAGE_CART_NO NOCACHE NOCYCLE";
    PreparedStatement pstmt3 = conn.prepareStatement(sql3);
    boolean result3 = pstmt3.execute();
    if(result3) {
        System.out.println("음료 카트 시퀀스 생성 실패 ...");
        return ;
    }
}

```

Code

emptyCart 장바구니 비우기 메서드

```

5. 로그인아웃
6. 회원탈퇴
메뉴 번호 선택: 4
관리자님의 장바구니
-----
상품명 | 수량 | 가격 | 요청사항
-----
아메리카노 | 1 | 4500 | 얼음많이
생크림카스텔라 | 1 | 4500 | 포크두개주세요
제주 조랑말 머그 | 1 | 44000 | null
-----
===== 장바구니 메뉴 =====
0. 뒤로가기
1. 주문하기
2. 비우기
메뉴 번호 입력: 2
장바구니 비우기 성공 !!!
===== MINI PROJECT =====

```

```
public void emptyCart() throws Exception {
    Connection conn = JdbcTemplate.getConnection();

    // Bev 카트 비우기
    String sql1 = "DELETE FROM BEVERAGE_CART";
    PreparedStatement pstmt1 = conn.prepareStatement(sql1);
    int result1 = pstmt1.executeUpdate();
    if(result1 == 0) {
        System.out.println("음료 장바구니 비우기 실패 ...");
        return ;
    }

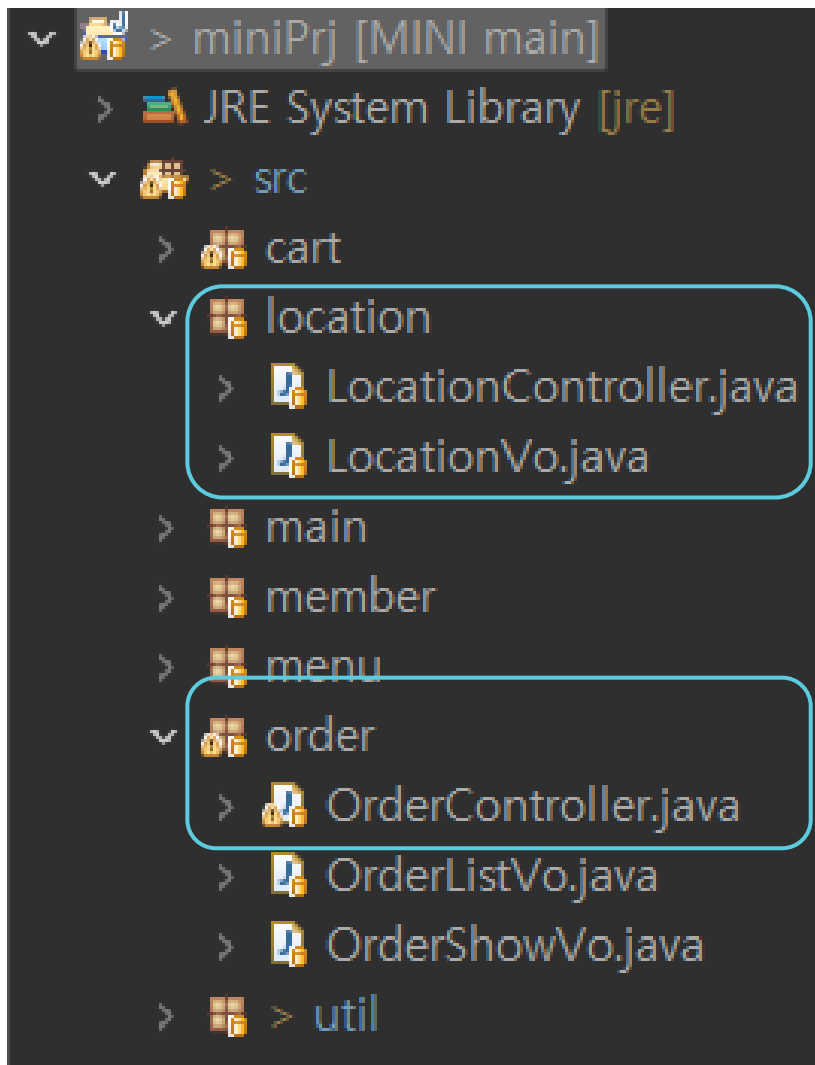
    // 시퀀스 삭제
    String sql2 = "DROP SEQUENCE SEQ_BEVERAGE_CART_NO";
    PreparedStatement pstmt2 = conn.prepareStatement(sql2);
    boolean result2 = pstmt2.execute();
    if(result2) {
        System.out.println("음료 카트 시퀀스 지우기 실패 ...");
        return ;
    }

    // 시퀀스 생성
    String sql3 = "CREATE SEQUENCE SEQ_BEVERAGE_CART_NO NOCACHE NOCYCLE";
    PreparedStatement pstmt3 = conn.prepareStatement(sql3);
    boolean result3 = pstmt3.execute();
    if(result3) {
        System.out.println("음료 카트 시퀀스 생성 실패 ...");
        return ;
    }
}
```

1.
DELET FROM (테이블명)으로
카트에 담긴 모든 데이터를 삭제
2.
카트에 담긴 순서를 나타내는
CART_NO 시퀀스를 삭제, 다시 생성
하면서 초기화



JAVA구현_신은지



location

- locationController
 1. showLocation
 2. showinitLocation
 3. printMenu

Order

- OrderController
 1. orderCart

```
private void showLocation() throws Exception {
    // conn
    Connection conn = JdbcTemplate.getConnection();

    String sql = "SELECT * FROM LOCATION WHERE LOCATION_DEL_YN = 'N' AND LOCATION_EXPO_YN = 'Y'";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rs = pstmt.executeQuery();

    locationList = new ArrayList<>();

    while (rs.next()) {
        LocationVo lv = new LocationVo();

        String locationNo = rs.getString("LOCATION_NO");
        String branchName = rs.getString("BRANCH_NAME");
        String distance = rs.getString("DISTANCE");
        String locationAddress = rs.getString("LOCATION_ADDRESS");
        String telephone = rs.getString("TELEPHONE");
        String locationDelYn = rs.getString("LOCATION_DEL_YN");
        String locationExpoYn = rs.getString("LOCATION_EXPO_YN");

        lv.setLocationNo(locationNo);
        lv.setBranchName(branchName);
        lv.setDistance(distance);
        lv.setLocationAddress(locationAddress);
        lv.setTelephone(telephone);
        lv.setLocationDelYn(locationDelYn);
        lv.setLocationExpoYn(locationExpoYn);

        locationList.add(lv); // 리스트에 추가합니다
    }

    for (LocationVo lv : locationList) {
        System.out.println("=====");
        System.out.println("지점번호: " + lv.getLocationNo());
        System.out.println("지점명: " + lv.getBranchName());
        System.out.println("지점주소: " + lv.getLocationAddress());
        System.out.println("거리(단위: m): " + lv.getDistance());
        System.out.println("전화번호: " + lv.getTelephone());
    }
    System.out.println("=====");
}
```

전체 매장 조회 기능

1. 폐업하지 않았으며 노출시킨 매장 정보를 모두 조회한 sql 구문 실행
2. While문 안에서 ArraaayList에 차례로 데이터 담기.
3. 향상된 for문을 통해, ArrayList에 담은 값을 꺼내서 출력하기

Code

```
private void showLocation() throws Exception {
    // conn
    Connection conn = JdbcTemplate.getConnection();

    String sql = "SELECT * FROM LOCATION WHERE LOCATION_DEL_YN = 'N' AND LOCATION_EXPO_YN = 'Y'";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rs = pstmt.executeQuery();

    locationList = new ArrayList<>();

    while (rs.next()) {
        LocationVo lv = new LocationVo();

        String locationNo = rs.getString("LOCATION_NO");
        String branchName = rs.getString("BRANCH_NAME");
        String distance = rs.getString("DISTANCE");
        String locationAddress = rs.getString("LOCATION_ADDRESS");
        String telephone = rs.getString("TELEPHONE");
        String locationDelYn = rs.getString("LOCATION_DEL_YN");
        String locationExpoYn = rs.getString("LOCATION_EXPO_YN");

        lv.setLocationNo(locationNo);
        lv.setBranchName(branchName);
        lv.setDistance(distance);
        lv.setLocationAddress(locationAddress);
        lv.setTelephone(telephone);
        lv.setLocationDelYn(locationDelYn);
        lv.setLocationExpoYn(locationExpoYn);

        locationList.add(lv); // 리스트에 추가합니다
    }
    for (LocationVo lv : locationList) {
        System.out.println("=====");
        System.out.println("지점번호: " + lv.getLocationNo());
        System.out.println("지점명: " + lv.getBranchName());
        System.out.println("지점주소: " + lv.getLocationAddress());
        System.out.println("거리(단위: m): " + lv.getDistance());
        System.out.println("전화번호: " + lv.getTelephone());
    }
    System.out.println("=====");
}
```



```
===== MINI PROJECT =====
0. 종료
1. 회원
2. 매장
3. 주문
메뉴 번호 입력: 2
----- 매장조회 -----
0. 뒤로가기
1. 전체 매장 조회
2. 근처 매장 조회
메뉴 번호 입력: 1
=====
지점번호: 1
지점명: 역삼포스코점
지점주소: 서울특별시 강남구 테헤란로134
거리(단위: m): 70
전화번호: 01045681687
=====
지점번호: 2
지점명: 국기원사거리점
지점주소: 서울특별시 강남구 테헤란로 125
거리(단위: m): 133
전화번호: 01059591234
=====
지점번호: 3
지점명: 청담점
지점주소: 서울특별시 강남구 도산대로532, 인희빌딩 1층
거리(단위: m): 3300
전화번호: 01087556121
=====
지점번호: 4
지점명: 석촌호수점
지점주소: 서울특별시 송파구 석촌호수로262
거리(단위: m): 6300
전화번호: 01078315465
=====
지점번호: 5
지점명: 왕십리역9번출구점
지점주소: 서울특별시 성동구 고산지로 234
거리(단위: m): 6900
전화번호: 01054565445
```



```
private void showInitLocation() throws Exception {
    // conn
    Connection conn = JdbcTemplate.getConn();

    String sql = "SELECT * FROM location WHERE LOCATION_DEL_YN = 'N' "
        + "AND LOCATION_EXPO_YN = 'Y' "
        + "AND TO_NUMBER(DISTANCE) <= 500 ORDER BY TO_NUMBER(DISTANCE)ASC";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rs = pstmt.executeQuery();

    ArrayList<LocationVo> lvList = new ArrayList<LocationVo>();

    while (rs.next()) {
        LocationVo lv = new LocationVo();

        String locationNo = rs.getString("LOCATION_NO");
        String branchName = rs.getString("BRANCH_NAME");
        String distance = rs.getString("DISTANCE");
        String telephone = rs.getString("TELEPHONE");
        String locationAddress = rs.getString("LOCATION_ADDRESS");

        lv.setLocationNo(locationNo);
        lv.setBranchName(branchName);
        lv.setDistance(distance);
        lv.setTelephone(telephone);
        lv.setLocationAddress(locationAddress);

        lvList.add(lv); // 리스트에 추가합니다
    }

    for (LocationVo lv: lvList) {
        System.out.println("=====");
        System.out.println("지점번호: " + lv.getLocationNo());
        System.out.println("지점명: " + lv.getBranchName());
        System.out.println("주소: " + lv.getLocationAddress());
        System.out.println("거리(단위: m): " + lv.getDistance());
        System.out.println("전화번호: " + lv.getTelephone());
    }
    System.out.println("=====");
}
```

매장 조회 기능

1. 폐업하지 않았고 노출시킨 매장 중에서 500m 이내의 매장정보를 거리순으로 조회
2. While문 안에서 ArrayList에 차례로 데이터 담기.
3. 향상된 for문을 통해, ArrayList에 담은 값을 꺼내서 출력하기

Code

```
private void showInitLocation() throws Exception {
    // conn
    Connection conn = JdbcTemplate.getConn();

    String sql = "SELECT * FROM location WHERE LOCATION_DEL_YN = 'N' "
        + "AND LOCATION_EXPO_YN = 'Y' "
        + "| AND TO_NUMBER(DISTANCE) <= 500 ORDER BY TO_NUMBER(DISTANCE)ASC";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rs = pstmt.executeQuery();

    ArrayList<LocationVo> lvList = new ArrayList<LocationVo>();

    while (rs.next()) {
        LocationVo lv = new LocationVo();

        String locationNo = rs.getString("LOCATION_NO");
        String branchName = rs.getString("BRANCH_NAME");
        String distance = rs.getString("DISTANCE");
        String telephone = rs.getString("TELEPHONE");
        String locationAddress = rs.getString("LOCATION_ADDRESS");

        lv.setLocationNo(locationNo);
        lv.setBranchName(branchName);
        lv.setDistance(distance);
        lv.setTelephone(telephone);
        lv.setLocationAddress(locationAddress);

        lvList.add(lv); // 리스트에 추가합니다
    }
    for (LocationVo lv: lvList) {
        System.out.println("=====");
        System.out.println("지점번호: " + lv.getLocationNo());
        System.out.println("지점명: " + lv.getBranchName());
        System.out.println("주소: " + lv.getLocationAddress());
        System.out.println("거리(단위: m): " + lv.getDistance());
        System.out.println("전화번호: " + lv.getTelephone());
    }
    System.out.println("=====");
}
```



```
----- 매장조회 -----
0. 뒤로가기
1. 전체 매장 조회
2. 근처 매장 조회
메뉴 번호 입력: 2
=====
지점번호: 1
지점명: 역삼포스코점
주소: 서울특별시 강남구 테헤란로134
거리(단위: m): 70
전화번호: 01045681687
=====
지점번호: 2
지점명: 국기원사거리점
주소: 서울특별시 강남구 테헤란로 125
거리(단위: m): 133
전화번호: 01059591234
=====
```

```
public class OrderController {
    public void orderCart() throws Exception {
        // conn
        Connection conn = JdbcTemplate.getConnection();

        // sql
        String sql = "INSERT INTO ORDER_SHOW (ORDER_NO, MEMBER_NO, LOCATION_NO)"
            + " VALUES (SEQ_ORDER_SHOW_NO.NEXTVAL, ?, ?)";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, Main.LoginMember.getMemberNo());
        pstmt.setString(2, Main.selectLocation.getLocationNo());
        int rowsAffected = pstmt.executeUpdate();

        if (rowsAffected > 0) {
            try {
            } catch (Exception e) {
                System.out.println("ORDER_SHOW 예외 발생");
                e.printStackTrace();
                return ;
            }
        }

        // 주문 트랜잭션 커밋
        conn.commit();
    } // orderCart
}
```

1. 회원번호, 매장번호를 사용자에게 입력 받아 DB에 삽입
2. 예외 발생시 출력문 출력
3. 삽입 내용을 반영하기 위해 주문 트랜잭션 커밋



JAVA구현_주선기

```

v [Icon] > miniPrj [MINI main]
  > [Icon] JRE System Library [jre]
  v [Icon] > src
    > [Icon] cart
    v [Icon] location
      > [Icon] LocationController.java
      > [Icon] LocationVo.java
    > [Icon] main
    > [Icon] member
    v [Icon] menu
      > [Icon] BeverageVo.java
      > [Icon] FoodVo.java
      > [Icon] MenuController.java
      > [Icon] MerchandiseVo.java
    v [Icon] order
      > [Icon] OrderController.java
      > [Icon] OrderListVo.java
      > [Icon] OrderShowVo.java
    > [Icon] > util

```

location

- locationController
- 1. selectLocation

Order

- orderController
- 1.orderList

Menu

- Menucontroller
- 1. printMenu

```
package util;

import java.sql.Connection;

public class JDBCTemplate {
    public static final Scanner SC = new Scanner(System.in);

    public static Connection getConn() throws Exception {
        String driver = "oracle.jdbc.OracleDriver";
        String url = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
        // 계정에 따라 id 값 바뀌서 하기
        String id = "C##MINI";
        String pwd = "1234";
        Class.forName(driver);
        Connection conn = DriverManager.getConnection(url, id, pwd);

        conn.setAutoCommit(false);

        return conn;
    } // getConn
} // class
```

1. util package에 Scanner 생성
2. getConn 메소드 만들어서 Java와 Oracle을 연결할 Connector 생성
3. Auto Commit 기능 끄고, 직접 커밋해서 사용할 수 있게함

```
public class Main {  
    public static MemberVo loginMember = null;  
    public static LocationVo selectLocation = null;  
    public static BeverageVo selectBeverage = null;  
    public static FoodVo selectFood = null;  
    public static MerchandiseVo selectMerchandise = null;  
}
```

4. 사용자 선택 정보 저장

```
while(rs.next()) {  
    int orderNo = rs.getInt("주문번호");  
    String name = rs.getString("상품명");  
    int count = rs.getInt("수량");  
    int sum = rs.getInt("가격");  
    String request = rs.getString("요청사항");  
    allSum += sum;  
  
    System.out.print(orderNo);  
    System.out.print(" | ");  
    System.out.print(name);  
    System.out.print(" | ");  
    System.out.print(count);  
    System.out.print(" | ");  
    System.out.print(sum);  
    System.out.print(" | ");  
    System.out.println(request);  
}  
System.out.println("-----");  
System.out.println("합계: " + allSum);  
System.out.println("주문자 전화번호: " + Main.LoginMember.getPhone());  
System.out.println("배송지: " + Main.LoginMember.getMemberAddress());  
System.out.println("주문한 지점번호: " + Main.selectLocation.getLocationNo());  
System.out.println("주문한 지점명: " + Main.selectLocation.getBranchName());  
System.out.println("주문한 지점주소: " + Main.selectLocation.getLocationAddress());  
System.out.println("=====");  
  
// 주문 지점 초기화  
Main.selectLocation = null;  
// orderList
```

1.
주문정보
(주문번호, 상품명, 수량, 가격, 요청사항) 가져와서 출력
2.
배송지 정보(배송지 주소, 연락처)
가져와서 출력
3.
선택했던 지점 정보
(지점번호, 지점명, 지점주소)
가져와서 출력

Code

```

while(rs.next()) {
    int orderNo = rs.getInt("주문번호");
    String name = rs.getString("상품명");
    int count = rs.getInt("수량");
    int sum = rs.getInt("가격");
    String request = rs.getString("요청사항");
    allSum += sum;

    System.out.print(orderNo);
    System.out.print(" | ");
    System.out.print(name);
    System.out.print(" | ");
    System.out.print(count);
    System.out.print(" | ");
    System.out.print(sum);
    System.out.print(" | ");
    System.out.println(request);
}
System.out.println("-----");
System.out.println("합계: " + allSum);
System.out.println("주문자 전화번호: " + Main.LoginMember.getPhone());
System.out.println("배송지: " + Main.LoginMember.getMemberAddress());
System.out.println("주문한 지점번호: " + Main.selectLocation.getLocationNo());
System.out.println("주문한 지점명: " + Main.selectLocation.getBranchName());
System.out.println("주문한 지점주소: " + Main.selectLocation.getLocationAddress());
System.out.println("=====");

// 주문 지점 초기화
Main.selectLocation = null;
// orderList

```



```

-----
===== 장바구니 메뉴 =====
0. 뒤로가기
1. 주문하기
2. 비우기
메뉴 번호 입력: 1
|=====|
황히찬님의 영수증
|=====|
주문번호 | 상품명 | 수량 | 가격 | 요청사항
|-----|
1 | 아메리카노 | 2 | 9000 | 얼음많이
2 | 생크림카스텔라 | 1 | 4500 | 포크두개
3 | 제주 조랑말 머그 | 1 | 44000 | 포장해주세요
4 |-----|
5 합계: 57500
6 주문자 전화번호: 01010108080
7 배송지: 서울시 마포구 공덕동
8 주문한 지점번호: 1
9 주문한 지점명: 역삼포스코점
10 주문한 지점주소: 서울특별시 강남구 테헤란로134
11 |=====|
12 주문 완료 !!!
13

```



JAVA구현_송예린

```

v [IDE] > miniPrj [MINI main]
  > [IDE] JRE System Library [jre]
  v [IDE] > src
    v [IDE] cart
      > [IDE] BeverageCartVo.java
      > [IDE] CartController.java
      > [IDE] FoodCartVo.java
      > [IDE] MerchandiseCartVo.java
    > [IDE] location
    > [IDE] main
    > [IDE] member
    v [IDE] menu
      > [IDE] BeverageVo.java
      > [IDE] FoodVo.java
      > [IDE] MenuController.java
      > [IDE] MerchandiseVo.java
    > [IDE] order
    > [IDE] > util
  
```

cart

- BeverageCart
- CartController
- MerchandiseCartVo

Menu

- Menucontroller
 1. merchandiseShow
 2. foodShow
 3. beverageShow

```
public void beverageCart(String num) throws Exception {
    Connection conn = JdbcTemplate.getConnection();

    String sql = "INSERT INTO BEVERAGE_CART ( BEVERAGE_CART_NO
    , MEMBER_NO
    , BEV_NO
    , BEV_COUNT
    , BEV_SUM
    , BEV_REQUEST
    )
    VALUES (
        SEQ_BEVERAGE_CART_NO.NEXTVAL
        , ?
        , ?
        , ?
        , ?
        , ?
    )";

    System.out.print("메뉴 수량: ");
    String count = JdbcTemplate.SC.nextLine();
    System.out.print("메뉴 요청사항: ");
    String request = JdbcTemplate.SC.nextLine();
}
```

```
// 회원번호
String no = Main.LoginMember.getMemberNo();
// 상품 한 개 가격
String price = Main.selectBeverage.getBevPrice();
// 상품 총 가격: price * count
int sumInt = Integer.parseInt(count) * Integer.parseInt(price);
String sum = Integer.toString(sumInt);

PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, no); // 유저 No 넘겨야함
pstmt.setString(2, num);
pstmt.setString(3, count);
pstmt.setString(4, sum); // Sum 넘겨야함 Num * Count
pstmt.setString(5, request);
int result = pstmt.executeUpdate();
```

사용자가 입력한 음료, 음식, 상품을
각각의 카트에 담아주는 foodCart

```
-- INSERT
INSERT INTO BEVERAGE_CART (
    BEVERAGE_CART_NO
    , MEMBER_NO
    , BEV_NO
    , BEV_COUNT
    , BEV_SUM
    , BEV_REQUEST
)
VALUES (
    SEQ_BEVERAGE_CART_NO.NEXTVAL
    , ?
    , ?
    , ?
    , ?
    , ?
)
```

1. INSERT를 사용하여 장바구니에
사용자로부터 입력받은 데이터 넣어주기

```
public void beverageCart(String num) throws Exception {
    Connection conn = JdbcTemplate.getConnection();

    String sql = "INSERT INTO BEVERAGE_CART ( BEVERAGE_CART_N

    System.out.print("메뉴 수량: ");
    String count = JdbcTemplate.SC.nextLine();
    System.out.print("메뉴 요청사항: ");
    String request = JdbcTemplate.SC.nextLine();
```

```
// 회원번호
String no = Main.LoginMember.getMemberNo();
// 상품 한 개 가격
String price = Main.selectBeverage.getBevPrice();
// 상품 총 가격: price * count
int sumInt = Integer.parseInt(count) * Integer.parseInt(price);
String sum = Integer.toString(sumInt);

PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, no); // 유저 No 넘겨야함
pstmt.setString(2, num);
pstmt.setString(3, count);
pstmt.setString(4, sum); // Sum 넘겨야함 Num * Count
pstmt.setString(5, request);
int result = pstmt.executeUpdate();
```

2.

사용자로부터 메뉴 수량과 요청사항
입력받아 데이터에 저장

3.

주문을 처리하는 과정에서 회원번호,
상품 가격, 상품 총 가격을 계산하는
과정. 각 물음표 자리에 값을 설정하여
SQL 문을 완성하고 실행

4.

장바구니 추가 됐는지 출력

```
// 회원번호
String no = Main.loginMember.getMemberNo();
// 상품 한 개 가격
String price = Main.selectBeverage.getBevPrice();
// 상품 총 가격: price * count
int sumInt = Integer.parseInt(count) * Integer.parseInt(price);
String sum = Integer.toString(sumInt);
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, no); // 유저 No 넘겨야함
pstmt.setString(2, num);
pstmt.setString(3, count);
pstmt.setString(4, sum); // Sum 넘겨야함 Num * Count
pstmt.setString(5, request);
int result = pstmt.executeUpdate();
```

```
if (result != 1) {
    System.out.println("장바구니 추가 실패 ...");
}
System.out.println("상품이 장바구니에 담겼습니다 !!!");
Main.selectBeverage = null;
conn.commit();
} // beverage
```

2.

사용자로부터 메뉴 수량과 요청사항
입력받아 데이터에 저장

3.

주문을 처리하는 과정에서 회원번호,
상품 가격, 상품 총 가격을 계산하는
과정. 각 물음표 자리에 값을 설정하여
SQL 문을 완성하고 실행

4.

장바구니 추가 됐는지 출력


```
private void beverageShow() throws Exception {
    // conn
    Connection conn = JdbcTemplate.getConnection();

    // SQL
    String sql = "SELECT B.BEV_NO , B.BEV_NAME , B.BEV_PRICE , C.BEV_CATEGORY , B.BEV_STOCK
    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rs = pstmt.executeQuery();
```

```
ArrayList<BeverageVo> mList = new ArrayList<BeverageVo>();
BeverageVo mc = null;

while(rs.next()) {
    String bevNo = rs.getString("BEV_NO");
    String bevName = rs.getString("BEV_NAME");
    String bevPrice = rs.getString("BEV_PRICE");
    String bevCategory = rs.getString("BEV_CATEGORY");
    String bevStock = rs.getString("BEV_STOCK");

    mc = new BeverageVo();

    mc.setBevNo(bevNo);
    mc.setBevName(bevName);
    mc.setBevPrice(bevPrice);
    // Category 값을 Code 멤버에 할당해줌 -> 임시로.....
    mc.setBevCode(bevCategory);
    mc.setBevStock(bevStock);

    mList.add(mc);
}
```

음료, 음식, 상품 품목을 조회할 수 있는
merchandiseShow, foodShow, beverageShow
메소드 생성

```
SELECT B.BEV_NO , B.BEV_NAME , B.BEV_PRICE , C.BEV_CATEGORY , B.BEV_STOCK
FROM BEVERAGE B
JOIN BEV_CATEGORY C ON B.BEV_CODE = C.BEV_CODE
;
```

1.
BEVERAGE테이블과 BEVERAGE_CART
테이블을 조인하여 SELECT문으로
저장된 데이터 꺼내오기
2.
쿼리를 실행하면 결과 집합이 생성되고,
해당 결과 집합은 ResultSet 객체인 rs에 저장

```
private void beverageShow() throws Exception {  
    // conn  
    Connection conn = JdbcTemplate.getConnection();  
  
    // SQL  
    String sql = "SELECT B.BEV_NO , B.BEV_NAME , B.BEV_PRICE , C.BEV_CATE  
    PreparedStatement pstmt = conn.prepareStatement(sql);  
    ResultSet rs = pstmt.executeQuery();
```

```
    ArrayList<BeverageVo> mList = new ArrayList<BeverageVo>();  
    BeverageVo mc = null;  
  
    while(rs.next()) {  
        String bevNo = rs.getString("BEV_NO");  
        String bevName = rs.getString("BEV_NAME");  
        String bevPrice = rs.getString("BEV_PRICE");  
        String bevCategory = rs.getString("BEV_CATEGORY");  
        String bevStock = rs.getString("BEV_STOCK");  
  
        mc = new BeverageVo();  
  
        mc.setBevNo(bevNo);  
        mc.setBevName(bevName);  
        mc.setBevPrice(bevPrice);  
        // Category 값을 Code 멤버에 할당해줌 -> 임시로.....  
        mc.setBevCode(bevCategory);  
        mc.setBevStock(bevStock);  
  
        mList.add(mc);  
    }  
}
```

3.

음료 정보를 데이터베이스에서 조회하여 BeverageVo 객체에 저장하고, 이 객체들을 ArrayList에 추가하는 작업을 수행한다.



시연



소감



Q&A

```
-- 영수증 출력
SELECT
    O.ORDER_NO "주문번호"
    , B.BEV_NAME "상품명"
    , C.BEV_COUNT "수량"
    , C.BEV_SUM "가격"
    , C.BEV_REQUEST "요청사항"
FROM BEVERAGE_CART C
JOIN BEVERAGE B ON (C.BEV_NO = B.BEV_NO)
JOIN ORDER_SHOW O ON (C.MEMBER_NO = O.MEMBER_NO)

UNION

SELECT
    O.ORDER_NO "주문번호"
    , F.FOOD_NAME "상품명"
    , C.FOOD_COUNT "수량"
    , C.FOOD_SUM "가격"
    , C.FOOD_REQUEST "요청사항"
FROM FOOD_CART C
JOIN FOOD F ON (C.FOOD_NO = F.FOOD_NO)
JOIN ORDER_SHOW O ON (C.MEMBER_NO = O.MEMBER_NO)

UNION

SELECT
    O.ORDER_NO "주문번호"
    , M.MD_NAME "상품명"
    , C.MD_COUNT "수량"
    , C.MD_SUM "가격"
    , C.MD_REQUEST "요청사항"
FROM MERCHANDISE_CART C
JOIN MERCHANDISE M ON (C.MD_NO = M.MD_NO)
JOIN ORDER_SHOW O ON (C.MEMBER_NO = O.MEMBER_NO)
;
```

```
----- 장바구니 출력 -----
SELECT
    B.BEV_NAME "메뉴명"
    , C.BEV_NO "상품번호"
    , C.BEV_COUNT "개수"
    , C.BEV_SUM "합계"
    , C.BEV_REQUEST "요청사항"
FROM BEVERAGE_CART C
JOIN BEVERAGE B ON (C.BEV_NO = B.BEV_NO)

UNION

SELECT
    F.FOOD_NAME "메뉴명"
    , C.FOOD_NO "상품번호"
    , C.FOOD_COUNT "개수"
    , C.FOOD_SUM "합계"
    , C.FOOD_REQUEST "요청사항"
FROM FOOD_CART C
JOIN FOOD F ON (C.FOOD_NO = F.FOOD_NO)

UNION

SELECT
    M.MD_NAME "메뉴명"
    , C.MD_NO "상품번호"
    , C.MD_COUNT "개수"
    , C.MD_SUM "합계"
    , C.MD_REQUEST "요청사항"
FROM MERCHANDISE_CART C
JOIN MERCHANDISE M ON (C.MD_NO = M.MD_NO)
;
```