데이터베이스를 이용한

식료품 제철 안내 어플리케이션 개발

2021 년 2 학기

과목명	데이터베이스
담당교수	박상원 교수님
조명	다보(DABO)
조원	박서진 문진영 이준영(조장) 이홍렬
개발기간	2021.11.15~2021.12.18

프로그램 목적 및 개요

'제철제철'은 과일, 채소, 해산물 3 종 식료품의 제철과 해당 식료품의 판매자 정보를 알려주는 모바일 어플리케이션이다. 품목과 제철 외에도 판매자 정보와 같은 다량의 복합적인 데이터를 데이터베이스를 이용해 정리하여 어플로 구현하는 것을 목표로 하였다.

어플리케이션은 크게 3 가지 기능을 제공한다. 각 기능은 하나의 메뉴로 구현되어 총 3 개의 메뉴를 지원한다. 첫째, '제철음식을 알고 싶어요'는 각 식료품의 품목명과 제철 개월을 나타낸다. 둘째, '구매하고 싶어요'는 각 식료품을 판매하는 판매자의 정보를 출력한다. 셋째, '모든 판매자 리스트'는 모든 판매자의 연락처와 지역 정보를 출력한다. 이 때, 판매자 정보는 임의로 만든 가상의 정보를 사용하였다.

어플리케이션 개발은 안드로이드 스튜디오로 진행되었으며, 해당 IDE 의 지원을 따라 프로그래밍 언어로 JAVA 를, 데이터베이스로 SQLite 를 사용하였다.

팀원 역할분담

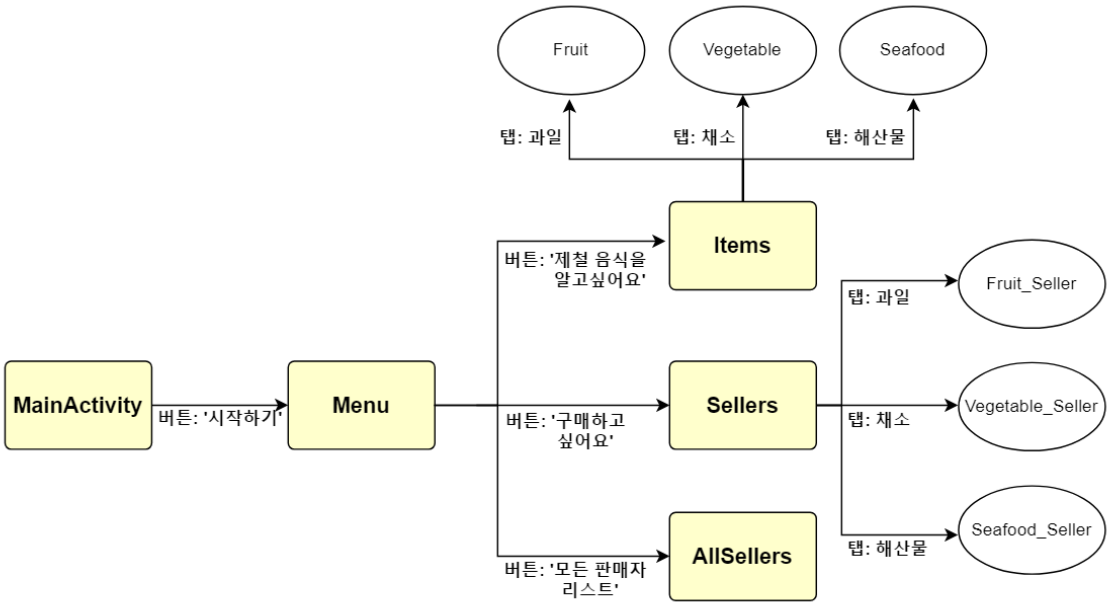
- **박서진**: 레이아웃 디자인, 발표 PPT 제작
- **문진영**: 프로그램 내 DB 구축
- **이준영**: 조장, 보고서 작성
- **이홍렬**: 어플리케이션 파일 가공

※ **공통**: 자료 조사, 안드로이드 스튜디오 및 JAVA 코드 작성

프로그램 구조 및 설명

어플리케이션 구조

'제철제철'의 어플리케이션 구조를 다이어그램으로 도식화하면 다음과 같이 표현할 수 있다. 이 때 사각형은 하나의 액티비티(페이지)를 의미하며, 원은 페이지 내의 탭을 의미한다.



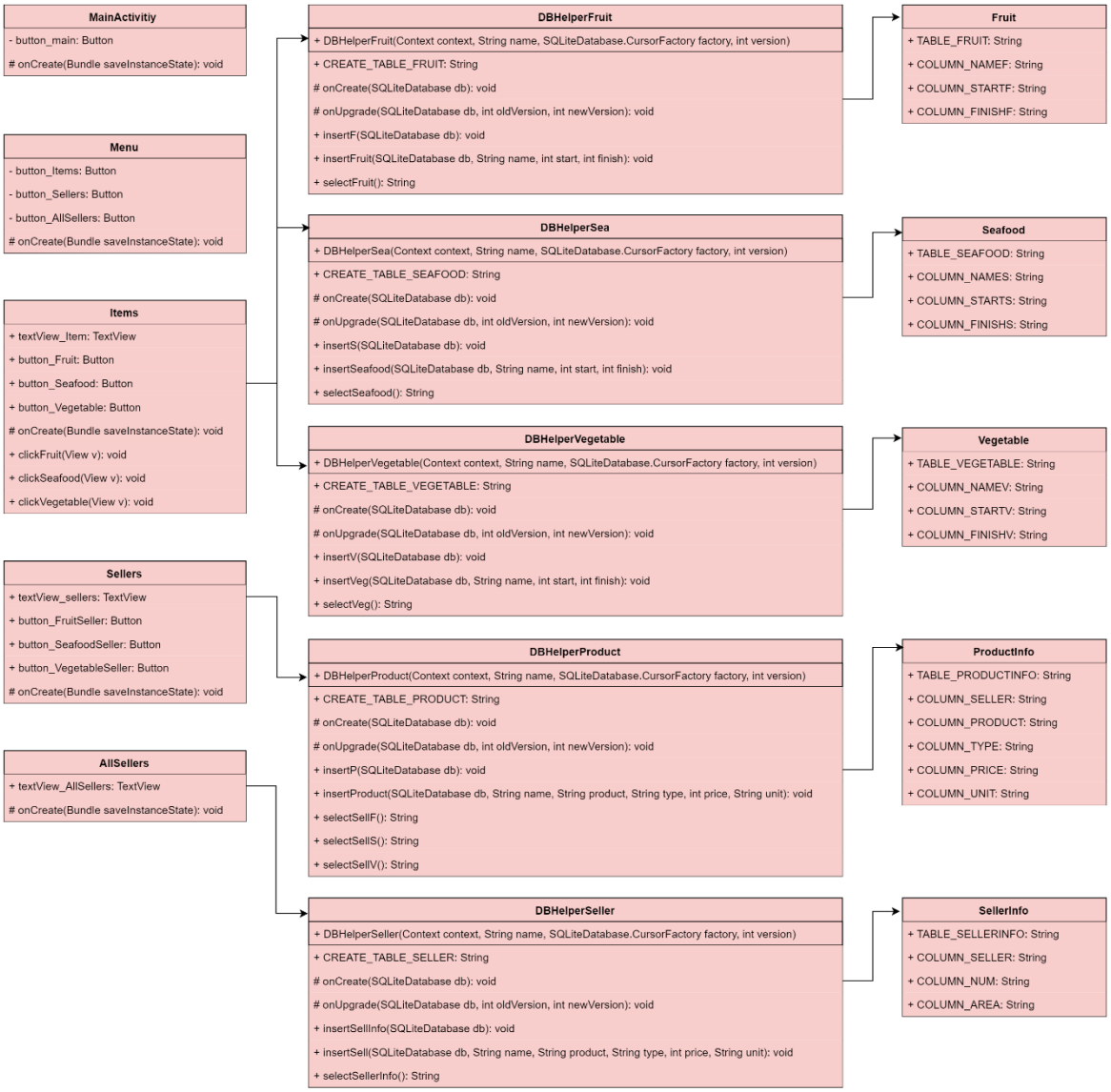
소스코드 분류

Android 스튜디오 내에서 소스코드는 다음 4 가지로 나뉜다.

1. Activity Java class: 어플의 화면(Activity) 구성 중 내부 수행에 관여하는 프로젝트 폴더 내의 자바 클래스이다. 확장자는 .java 이다.
2. Activity xml file: 1 번의 자바 클래스들에 일대일로 대응하여 화면의 구성과 디자인을 정의하는 레이아웃 파일이다. 확장자는 .xml 이다.
3. Database helper: 안드로이드 스튜디오에서 자체 지원하는 데이터베이스인 SQLite 를 지원하기 위한 클래스이다. 확장자는 .java 이다.
4. Database model: 각 테이블(데이터베이스의 Relation 에 해당)의 구조를 정의한다. 테이블 이름 및 이에 사용되는 애트리뷰트 이름의 정보를 String 자료형으로 가지며, 이를 통해 추가, 수정, 삭제를 용이하게 한다.

Java class 를 이용한 데이터베이스 구현

상기한 4 가지 분류 중 activity java class 와 일대일 대응하여 화면을 출력하는 xml 파일을 제외한 나머지 세 가지 분류(.java 확장자)의 관계를 통하여 어플리케이션의 구조를 나타내고자 한다. 다음 그림은 클래스 간의 구조를 UML 클래스 다이어그램으로 도식화한 것이다.



이 때, 좌측 클래스들은 AppCompatActivity 클래스를 상속하는 액티비티 클래스이며, 중앙의 클래스들은 SQLiteOpenHelper 클래스를 상속하는 데이터베이스 헬퍼 클래스이고, 우측 클래스들은 BaseColumns 클래스를 계승하는 데이터베이스 모델 클래스이다.

데이터베이스의 실적용 사례를 위해 하나의 탭까지 도달하여 데이터를 얻는 과정을 소스코드를 통해 설명하고자 한다. 예컨대 사용자가 채소 판매자의 정보를 얻고 싶다고 가정하자. 이 경우, 우선 사용자는 어플을 실행하고 나온 MainActivity 에서 '시작' 버튼을 눌러 Menu 액티비티로

진입한다. 다시 여기서 두 번째 버튼인 '구매하고 싶어요'를 선택하면 Sellers 액티비티로 이동하는데, Sellers 액티비티는 onCreate 메소드[1]에서 다음을 정의한다.

```
dbHelperProduct = new DBHelperProduct(Sellers.this, DATABASE_NAME, null, 1);

//정보 삽입이 가능한 상태로 만들기
SQLiteDatabase db3 = dbHelperProduct.getWritableDatabase();

String count = "SELECT count(*) FROM " +
ProductInfo.productInfo.TABLE_PRODUCTINFO;
```

만일 Sellers 액티비티가 최초로 실행되었다면 onCreate 메소드 내의 다음 if 문이 실행된다.

```
//최초 생성시에만 데이터를 삽입하고 그 이후에는 삽입하지 않도록 검사
        if(countf <= 0) {
            dbHelperProduct.insertP(db3);
        }else{}
```

이는 최초 실행에 한해 헬퍼 DBHelperProduct.java 내에서 다음과 같은 과정을 거쳐 데이터베이스를 생성한다는 것을 의미한다.

① InsertP 실행

```
//삽입할 데이터 정의
    public void insertP(SQLiteDatabase db) {
        insertProduct(db, "A", "블루베리", "과일", 29000, "1kg");
        insertProduct(db, "A", "체리", "과일", 21000, "1kg");
        insertProduct(db, "B", "감", "과일", 17000, "10kg");
        (중략)
        insertProduct(db, "M", "방어", "해산물", 90000, "6kg");
        insertProduct(db, "M", "삼치", "해산물", 44000, "3kg");
        insertProduct(db, "M", "파래", "해산물", 3000, "1kg");
        insertProduct(db, "N", "게", "해산물", 25000, "1kg");
        insertProduct(db, "N", "주꾸미", "해산물", 28000, "1kg");
        insertProduct(db, "N", "키조개", "해산물", 13000, "1kg");
    }
```

② InsertProduct 실행

```
//insertP 에서 인자받아서 테이블에 삽입
```

[1] onCreate 메소드는 해당 액티비티 진입 시 자동으로 실행되는 메소드이다.

```
    public void insertProduct(SQLiteDatabase db, String name, String product,
String type, int price, String unit) {
        ContentValues values = new ContentValues();
        //위 함수에 받은 정보를 각 애트리뷰트에 맞게 저장한다.
        values.put(ProductInfo.productInfo.COLUMN_SELLER, name);
        values.put(ProductInfo.productInfo.COLUMN_PRODUCT, product);
        values.put(ProductInfo.productInfo.COLUMN_TYPE, type);
        values.put(ProductInfo.productInfo.COLUMN_PRICE, price);
        values.put(ProductInfo.productInfo.COLUMN_UNIT, unit);
        //판매자정보 테이블에 values 값을 넣어준다.
        db.insert(ProductInfo.productInfo.TABLE_PRODUCTINFO, null, values);
    }
```

③ 인자를 거쳐 모델 ProductInfo.java 의 해당 구조를 통해 데이터베이스에 저장

```
public static class productInfo implements BaseColumns {
        public static final String TABLE_PRODUCTINFO = "productInfo"; //판매품목
테이블 이름 정의
        public static final String COLUMN_SELLER = "seller"; //판매자 이름
attribute
        public static final String COLUMN_PRODUCT = "product"; //품목 이름
attribute
        public static final String COLUMN_TYPE = "type"; //품목에 대한 유형(과일,
해산물, 채소) attribute
        public static final String COLUMN_PRICE = "price"; //가격 attribute
        public static final String COLUMN_UNIT = "unit"; //판매 단위 attribute
```

따라서 해당 데이터는 Sellers.java 에서 정의한 DATABASE_NAME = "season.db"에 저장된다. 사용자가 채소 판매자의 정보를 얻고자 '채소' 탭을 선택하면, onCreate 메소드의 다음 코드가 실행된다. 이 때 onClick(View v)는 항목의 터치에 반응하는 메소드이다.

```
button_VegetableSeller.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView_sellers.setText( dbHelperProduct.selectSellV());
            }
        });
```

이는 다시 DBHelperProduct.java 의 selectSellV() 메소드를 실행시키는데, 그 내용은 다음과 같다.

```
//판매품목에서 채소만 파는 판매자를 보고 싶을 때
    public String selectSellV(){
        SQLiteDatabase db = getReadableDatabase();
        String result = "";
        Cursor cursor = db.rawQuery("SELECT * FROM productInfo WHERE type =
'채소'", null); // 테이블의 전체 정보 삽입
```

```
        while(cursor.moveToNext()) {
            result += " 판매자: " + cursor.getString(1) //column(0) 은 id
정보이므로 1 부터 얻어온다.
                    +", " + cursor.getString(2)
                    + ", 가격: " + cursor.getString(4)
                    + ", 단위: " + cursor.getString(5)
                    +"\n";
        }
        return result;
    }
```

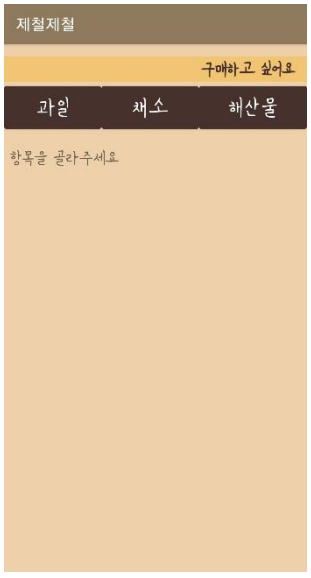
여기서 Cursor 는 안드로이드 스튜디오에서 제공하는 데이터베이스 참조 인터페이스로, Table 구조의 행과 열을 개별적으로 참조하는 기능을 수행한다. Cursor 타입의 cursor 는 season.db 파일 내의 productInfo 릴레이션에서 type 이 '채소'인 뷰를 생성하여 가리킨다. 이는 해당 메소드에 제시된 SQL 질의문 "SELECT * FROM productInfo WHERE type = '채소'"에 의해 이루어진다.

하단 반복문에 의해 읽어온 정보는 하나의 String 으로 저장되고, selectSellV() 메소드는 이를 반환하여 사용자에게 '채소 판매자' 정보를 제공한다.

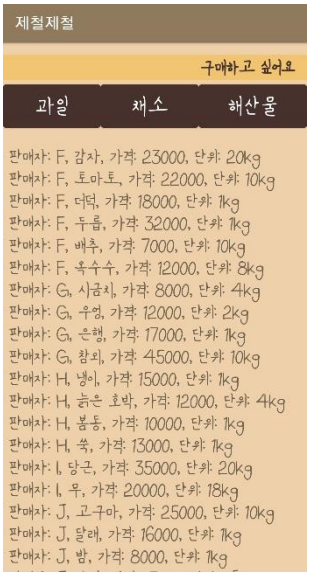
상기한 내용을 실제 어플리케이션 화면으로 나타내면 다음과 같다.

 # 1 MainActivity

 # 2 Menu



 # 3 Seller



 # 4 Seller: 채소 선택

소스 코드

Activity Java Class

① MainActivity.java

안드로이드 스튜디오에서 MainActivity.java 의 onCreate() 함수는 자바의 main 함수와 동일한 역할을 수행한다. 즉 어플리케이션 실행 시 가장 처음 실행되는 액티비티이다.

```
package com.example.test;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;



public class MainActivity extends AppCompatActivity {

    //메인화면 버튼
    private Button button_main;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        //'시작하기' 버튼 누르면 메인화면에서 메뉴화면으로 이동
        button_main = findViewById(R.id.button_main);
        button_main.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, Menu.class);
                startActivity(intent);


            }
        });

    }
}
```

② Menu.java

각각의 기능을 제공하는 3 가지 버튼을 출력하는 액티비티이다. MainActivity 에서 '시작하기' 버튼을 선택하는 것으로 해당 액티비티에 진입한다.

```
package com.example.test;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Menu extends AppCompatActivity {

    //'제철음식을 알고싶어요' 버튼
    private Button button_Items;
    //'구매하고 싶어요' 버튼
    private Button button_Sellers;
    //'모든 판매자 리스트' 버튼
    private Button button_AllSellers;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);

        //'제철음식을 알고싶어요' 버튼 클릭하면 Items 액티비티로 이동
        button_Items = findViewById(R.id.button_Items);
        button_Items.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Menu.this, Items.class);
                startActivity(intent);

            }
        });

        //'구매하고 싶어요' 버튼 클릭하면 Sellers 액티비티로 이동
        button_Sellers = findViewById(R.id.button_Sellers);
        button_Sellers.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Menu.this, Sellers.class);
                startActivity(intent);

            }
        });

        //'모든 판매자 리스트' 버튼을 클릭하면 AllSellers 액티비티로 이동
```

```
        button_AllSellers = findViewById(R.id.button_AllSellers);
        button_AllSellers.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Menu.this, AllSellers.class);
                startActivity(intent);

            }
        });

    }
}
```

③ Items.java

첫 번째 기능인 '제철음식을 알고 싶어요' 기능을 제공하는 클래스이다. Menu 액티비티에서 해당 버튼을 선택 시 이 액티비티로 진입하며, 과일, 채소, 해산물 3 개의 탭을 통해 데이터베이스에서 읽은 해당 분류의 품목명과 제철 정보를 제공한다.

```
package com.example.test;

import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import info.androidhive.sqlite.helper.DBHelperFruit;
import info.androidhive.sqlite.helper.DBHelperSea;
import info.androidhive.sqlite.helper.DBHelperVegetable;

public class Items extends AppCompatActivity {

    DBHelperFruit dbHelper; //과일 이용 클래스
    DBHelperVegetable dbHelperV; //채소 이용 클래스
    DBHelperSea dbHelperSea; //해산물 이용 클래스

    public TextView textView_Item;
    public Button button_Fruit;
    public Button button_Vegetable;
    public Button button_Seafood;
    public static final String DATABASE_NAME = "season.db";



    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_items);
        //Layout 의 요소들과 일대일 대응한다.
        textView_Item = findViewById(R.id.textView_Items);
        button_Fruit = (Button) findViewById(R.id.button_Fruit);
        button_Vegetable= (Button) findViewById(R.id.button_Vegetable);
        button_Seafood = (Button) findViewById(R.id.button_Seafood);

        textView_Item.setText(" 항목을 골라주세요");


        dbHelper = new DBHelperFruit(Items.this, DATABASE_NAME, null,1); //과일
테이블 선언
        dbHelperV = new DBHelperVegetable(Items.this, DATABASE_NAME, null, 1);
//채소 테이블 선언
        dbHelperSea = new DBHelperSea(Items.this, DATABASE_NAME, null, 1);
//해산물 테이블 선언

        SQLiteDatabase db = dbHelper.getWritableDatabase(); //정보 삽입이 가능한
상태로 만들기
        SQLiteDatabase db1 = dbHelperV.getWritableDatabase();
        SQLiteDatabase db2 = dbHelperSea.getWritableDatabase();

        dbHelper.onCreate(db); //과일 테이블 생성

        dbHelperV.onCreate(db1); //채소 테이블 생성

        dbHelperSea.onCreate(db2); //해산물 테이블 생성

        String count = "SELECT count(*) FROM fruit";
        Cursor cursorc = db.rawQuery(count, null);
        //커서: 데이터베이스 테이블을 이용할 때 가리켜서 읽어오는 역할
        cursorc.moveToFirst();
        int countf = cursorc.getInt(0);

        //최초 생성시에만 데이터를 삽입하고 그 이후에는 삽입하지 않도록 검사
        //3 개 유형 모두 같은 페이지이기 때문에 기준을 과일 테이블 1 개로 하였다.
        if(countf <= 0) {
            dbHelper.insertF(db);
            dbHelperV.insertV(db1);
            dbHelperSea.insertS(db2);
        }
        else{}


    }
    //각 버튼을 눌렀을 때 화면에 보여지는 텍스트 뷰
    //각 데이터베이스 헬퍼의 함수를 불러온다
    public void clickFruit(View v) {  //버튼을 눌렀을 때 출력용
        textView_Item.setText(dbHelper.selectFruit());
    }
```

```
    public void clickVegetable(View v) {
        textView_Item.setText(dbHelperV.selectVeg());
    }
    public void clickSeafood(View v) {
        textView_Item.setText(dbHelperSea.selectSeafood());
    }


}
```

④ Sellers.java

두 번째 기능인 '구매하고 싶어요' 기능을 제공하는 클래스이다. Menu 액티비티에서 해당 버튼을 선택 시 이 액티비티로 진입하며, 과일, 채소, 해산물 3 개의 탭을 통해 데이터베이스에서 읽은 해당 품목 판매자의 정보를 제공한다.

```
package com.example.test;

import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import info.androidhive.sqlite.helper.DBHelperProduct;
import info.androidhive.sqlite.model.ProductInfo;

public class Sellers extends AppCompatActivity {

    DBHelperProduct dbHelperProduct;

    public Button button_FruitSeller; //과일 보여주는 버튼
    public Button button_VegetableSeller;//채소 보여주는 버튼
    public Button button_SeafoodSeller;//해산물 보여주는 버튼
    public TextView textView_sellers;
    //사용할 데이터베이스 이름
    public static final String DATABASE_NAME = "season.db";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sellers);
        //텍스트뷰, 버튼 선언,사용할 데이터베이스 이름
        button_FruitSeller = (Button) findViewById(R.id.button_FruitSeller);
        button_VegetableSeller = (Button)
findViewById(R.id.button_VegetableSeller);
```

```
        button_SeafoodSeller = (Button)
findViewById(R.id.button_SeafoodSeller);

        textView_sellers = findViewById(R.id.textView_sellers);
        dbHelperProduct = new DBHelperProduct(Sellers.this, DATABASE_NAME,
null, 1);

        textView_sellers.setText(" 항목을 골라주세요");

        //정보 삽입이 가능한 상태로 만들기
        SQLiteDatabase db3 = dbHelperProduct.getWritableDatabase();

        dbHelperProduct.onCreate(db3);

        String count = "SELECT count(*) FROM " +
ProductInfo.productInfo.TABLE_PRODUCTINFO;

        Cursor cursorc = db3.rawQuery(count, null);
        cursorc.moveToFirst();
        int countf = cursorc.getInt(0);

        //최초 생성시에만 데이터를 삽입하고 그 이후에는 삽입하지 않도록 검사
        if(countf <= 0) {


            dbHelperProduct.insertP(db3);

        }else{}

        //버튼 과일,채소,해산물을 사용할 수 있게 해주는 onClick 기능 선언
        //버튼 누르고 View v 로 데이터 표현
        button_FruitSeller.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView_sellers.setText( dbHelperProduct.selectSellF());
            }
        });
        button_VegetableSeller.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView_sellers.setText( dbHelperProduct.selectSellV());
            }
        });
        button_SeafoodSeller.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView_sellers.setText( dbHelperProduct.selectSellS());
            }
        });

    }
}
```

⑤ AllSellers.java

세 번째 기능인 '모든 판매자 리스트' 기능을 제공하는 클래스이다. Menu 액티비티에서 해당 버튼을 선택 시 이 액티비티로 진입하며, 모든 판매자의 연락처와 지역 정보를 제공한다.

```
package com.example.test;

import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import info.androidhive.sqlite.helper.DBHelperSeller;

public class AllSellers extends AppCompatActivity {

    DBHelperSeller dbHelperSeller;

    public TextView textView_AllSellers;

    public static final String DATABASE_NAME = "season.db";



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_all_sellers);

        textView_AllSellers = findViewById(R.id.textView_AllSellers);
        dbHelperSeller = new DBHelperSeller(AllSellers.this, DATABASE_NAME,
null, 1); //판매자 정보 테이블 선언

        SQLiteDatabase db4 = dbHelperSeller.getWritableDatabase();

        // SellerInfo 테이블 열기
        dbHelperSeller.onCreate(db4);

        // 기존 db 검사용 Cursor
        String count = "SELECT count(*) FROM sellerInfo";
        Cursor cursor = db4.rawQuery(count, null);
        cursor.moveToFirst();
        int countF = cursor.getInt(0);

        // 기존 db 를 검사해 비어 있는 경우에만 데이터를 삽입
```

```
        if(countF <= 0) {
            dbHelperSeller.insertSellInfo(db4);
        }else {}

        textView_AllSellers.setText( dbHelperSeller.selectSellerInfo());

    }

}
```

Helper

① DBHelperFruit.java

```
package info.androidhive.sqlite.helper;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import info.androidhive.sqlite.model.Fruit;

import androidx.annotation.Nullable;

public class DBHelperFruit extends SQLiteOpenHelper { //////과일만을 하는
헬퍼!!!!!!!!!!!!!!!!!!!!

    //Logcat tag
    public static final String LOG = "DatabaseHelper";
    //database version
    public static final int DATABASE_VERSION = 1;
    //database name
    public static final String DATABASE_NAME = "season.db";


    //create statement -과일
    public static final String CREATE_TABLE_FRUIT = "CREATE TABLE if not exists
" + Fruit.fruit.TABLE_FRUIT +
            " (" + Fruit.fruit._ID + " INTEGER PRIMARY KEY autoincrement, "
            + Fruit.fruit.COLUMN_NAMEF + " TEXT , " + Fruit.fruit.COLUMN_STARTF
+ " INTEGER , "
            + Fruit.fruit.COLUMN_FINISHF + " INTEGER )";

    public DBHelperFruit(@Nullable Context context, String name,
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
```

```
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_FRUIT); //테이블 생성
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + Fruit.fruit.TABLE_FRUIT); //갱신 시
삭제 후 다시 생성

        onCreate(db);
    }

    //데이터 삽입
    public void insertF(SQLiteDatabase db) { //데이터 삽입
        insertFruit(db,"감", 10, 11);
        insertFruit(db, "귤", 11, 1);
        insertFruit(db, "딸기", 12, 2);
        insertFruit(db, "매실", 5, 7);
        insertFruit(db, "멜론", 7, 10);
        insertFruit(db, "배", 9, 11);
        insertFruit(db, "복숭아", 6, 8);
        insertFruit(db, "블루베리", 7, 9);
        insertFruit(db, "사과", 10, 12);
        insertFruit(db, "석류", 9, 12);
        insertFruit(db, "유자", 11, 12);
        insertFruit(db, "자두", 7, 8);
        insertFruit(db, "체리", 5, 6);
        insertFruit(db, "키위", 5, 11);
        insertFruit(db, "포도", 8, 8);
        insertFruit(db, "한라봉", 12, 2);

    }
    //위 insertF 에서 받아서 테이블에 생성
    public void insertFruit(SQLiteDatabase db, String name, int start, int
finish) {
        ContentValues values = new ContentValues();

        values.put(Fruit.fruit.COLUMN_NAMEF, name);
        values.put(Fruit.fruit.COLUMN_STARTF, start);
        values.put(Fruit.fruit.COLUMN_FINISHF, finish);

        db.insert(Fruit.fruit.TABLE_FRUIT, null, values);
    }

    //과일에 테이블 출력 함수
    public String selectFruit(){
        SQLiteDatabase db = getReadableDatabase(); //저장된 데이터 읽어오는 함수
        String result = "";

        Cursor cursor = db.rawQuery("SELECT * FROM fruit", null); //과일
테이블의 전체 정보 삽입
```

```
        while(cursor.moveToNext()) {
            result += " " + cursor.getString(1) //column(0) 은 id 정보이므로
1 부터 얻어온다.
                    +", 제철시기: " + cursor.getString(2)
                    + "월 ~ " + cursor.getString(3)
                    +"월 \n";
        }
        return result; //return 하여 텍스튜에 전달
    }
}
```

② DBHelperSea.java

```
package info.androidhive.sqlite.helper;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import info.androidhive.sqlite.model.Seafood;

import androidx.annotation.Nullable;

public class DBHelperSea extends SQLiteOpenHelper {  ////해산물!!!!! 디비헬퍼
    //database version
    public static final int DATABASE_VERSION = 1;
    //database name
    public static final String DATABASE_NAME = "season.db";
    //create statement -해산물
    public static final String CREATE_TABLE_SEAFOOD = "CREATE TABLE if not
exists " + Seafood.seafood.TABLE_SEAFOOD +
            " (" + Seafood.seafood._ID + " INTEGER PRIMARY KEY autoincrement, "
+ Seafood.seafood.COLUMN_NAMES + " TEXT , "
            + Seafood.seafood.COLUMN_STARTS + " INTEGER , " +
Seafood.seafood.COLUMN_FINISHS + " INTEGER )";


    public DBHelperSea(@Nullable Context context, @Nullable String name,
@Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_SEAFOOD);
    } //해산물 테이블 생성

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
        db.execSQL("DROP TABLE IF EXISTS " + Seafood.seafood.TABLE_SEAFOOD);
//갱신 시 삭제후 다시 생성
        onCreate(db);
    }

    //데이터 삽입
    public void insertS(SQLiteDatabase db) {
        insertSeafood(db, "갈치", 7, 10);
        insertSeafood(db, "감태", 12, 1);
        insertSeafood(db, "게", 9, 10);
        insertSeafood(db, "고등어", 9, 11);
        insertSeafood(db, "과메기", 12, 2);
        insertSeafood(db, "굴", 9, 12);
        insertSeafood(db, "꼬막", 11, 3);
        insertSeafood(db, "꽁치", 10, 11);
        insertSeafood(db, "농어", 6, 8);
        insertSeafood(db, "대하", 9, 12);
        insertSeafood(db, "도미", 1, 3);
        insertSeafood(db, "매생이", 11, 2);
        insertSeafood(db, "메기", 7, 7);
        insertSeafood(db, "민어", 8, 8);
        insertSeafood(db, "바지락", 2, 4);
        insertSeafood(db, "방어", 11, 2);
        insertSeafood(db, "삼치", 10, 2);
        insertSeafood(db, "전복", 8, 10);
        insertSeafood(db, "전어", 9, 11);
        insertSeafood(db, "주꾸미", 3, 5);
        insertSeafood(db, "키조개", 4, 5);
        insertSeafood(db, "파래", 12, 2);
        insertSeafood(db, "홍합", 10, 12);
    }

    //insertS 에서 받아서 테이블에 삽입
    public void insertSeafood(SQLiteDatabase db, String name, int start, int
finish) {
        ContentValues values = new ContentValues();

        values.put(Seafood.seafood.COLUMN_NAMES, name);
        values.put(Seafood.seafood.COLUMN_STARTS, start);
        values.put(Seafood.seafood.COLUMN_FINISHS, finish);

        db.insert(Seafood.seafood.TABLE_SEAFOOD, null, values);
    }

    //해산물에 대한 전체정보가 알고 싶을 때
    public String selectSeafood() {
        SQLiteDatabase db = getReadableDatabase();
        String result = "";

        Cursor cursor = db.rawQuery("SELECT * FROM seafood", null); //전체 출력
sql 문
```

```
        while(cursor.moveToNext()) {
            result += " " + cursor.getString(1)
                    +", 제철시기: " + cursor.getString(2)
                    + "월 ~ " + cursor.getString(3)
                    +"월 \n";
        }
        return result; //return 하여 텍스튜에 전달
    }
}
```

③ DBHelperVegetable.java

```
package info.androidhive.sqlite.helper;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import info.androidhive.sqlite.model.Vegetable;

import androidx.annotation.Nullable;

public class DBHelperVegetable extends SQLiteOpenHelper {  ////채소
ㄷ데베헬퍼!!!!!!!!!

    //Logcat tag
    public static final String LOG = "DatabaseHelper";
    //database version
    public static final int DATABASE_VERSION = 1;
    //database name
    public static final String DATABASE_NAME = "season.db";

    //create statement -채소
    public static final String CREATE_TABLE_VEGETABLE = "CREATE TABLE if not
exists " + Vegetable.vegetable.TABLE_VEGETABLE +
            " (" + Vegetable.vegetable._ID + " INTEGER PRIMARY KEY
autoincrement, "
        + Vegetable.vegetable.COLUMN_NAMEV + " TEXT , " +
Vegetable.vegetable.COLUMN_STARTV + " INTEGER , "
            + Vegetable.vegetable.COLUMN_FINISHV + " INTEGER )";

    public DBHelperVegetable(@Nullable Context context, @Nullable String name,
@Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //테이블 생성
```

```
        db.execSQL(CREATE_TABLE_VEGETABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " +
Vegetable.vegetable.TABLE_VEGETABLE); //갱신 시 삭제 후 다시 생성
        onCreate(db);
    }

    //데이터 삽입
    public void insertV(SQLiteDatabase db) {
        insertVeg(db, "감자", 6, 9);
        insertVeg(db, "고구마", 8, 10);
        insertVeg(db, "냉이", 4, 5);
        insertVeg(db, "늙은 호박", 10, 12);
        insertVeg(db, "달래", 3, 4);
        insertVeg(db, "당근", 12, 3);
        insertVeg(db, "더덕", 1, 4);
        insertVeg(db, "두릅", 4, 5 );
        insertVeg(db, "무", 10, 12);
        insertVeg(db, "밤", 10, 11);
        insertVeg(db, "배추", 11, 12);
        insertVeg(db, "봄동", 11, 3);
        insertVeg(db, "수박", 7, 8);
        insertVeg(db, "시금치", 11, 2);
        insertVeg(db, "쑥", 3, 3);
        insertVeg(db, "옥수수", 7, 9);
        insertVeg(db, "우엉", 1, 3);
        insertVeg(db, "은행", 9, 9);
        insertVeg(db, "참외", 6, 6);
        insertVeg(db, "토마토", 7, 9);
    }

    //insertV 에서 받아서 테이블에 삽입
    public void insertVeg(SQLiteDatabase db, String name, int start, int
finish) {
        ContentValues values = new ContentValues();

        values.put(Vegetable.vegetable.COLUMN_NAMEV, name);
        values.put(Vegetable.vegetable.COLUMN_STARTV, start);
        values.put(Vegetable.vegetable.COLUMN_FINISHV, finish);

        db.insert(Vegetable.vegetable.TABLE_VEGETABLE, null, values);
    }

    //채소에 대한 전체 정보가 알고 싶을 때
    public String selectVeg() {
        SQLiteDatabase db = getReadableDatabase();
        String result = "";
```

```
        Cursor cursor = db.rawQuery("SELECT * FROM vegetable", null); //전체
출력 sql 문

        while(cursor.moveToNext()) {
            result += " " + cursor.getString(1)
                    +", 제철시기: " + cursor.getString(2)
                    + "월 ~ " + cursor.getString(3)
                    +"월 \n";
        }
        return result;

    }
}
```

④ DBHelperProduct.java

```
package info.androidhive.sqlite.helper;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

import info.androidhive.sqlite.model.ProductInfo;


//품목정보 테이블을 생성 관리하는 디비헬퍼
public class DBHelperProduct extends SQLiteOpenHelper {

    //Logcat tag
    public static final String LOG = "DatabaseHelper";
    //database version
    public static final int DATABASE_VERSION = 1;
    //database name
    public static final String DATABASE_NAME = "season.db";
    //create statement -판매 품목 생성 문구
    //_id 는 생성했을 때 목록을 각각 넘버링해주는 주는 개념이다.
    //존재하지 않을 때만 생성
    public static final String CREATE_TABLE_PRODUCT = "CREATE TABLE if not
exists " + ProductInfo.productInfo.TABLE_PRODUCTINFO +
            " (" + ProductInfo.productInfo._ID + " INTEGER PRIMARY KEY
autoincrement, "
            + ProductInfo.productInfo.COLUMN_SELLER + " TEXT , " +
ProductInfo.productInfo.COLUMN_PRODUCT + " TEXT , "
            + ProductInfo.productInfo.COLUMN_TYPE + " TEXT , " +
ProductInfo.productInfo.COLUMN_PRICE + " INTEGER , "
            + ProductInfo.productInfo.COLUMN_UNIT + " TEXT )";
```

```
    //생성자를 호출하여 상속받는다. 상속해서 여기 클래스에 있는 함수를 사용하게끔
한다.
    public DBHelperProduct(@Nullable Context context, @Nullable String name,
@Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //테이블 생성
        db.execSQL(CREATE_TABLE_PRODUCT); //테이블 생성
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //갱신 시 삭제후 다시 생성
        db.execSQL("DROP TABLE IF EXISTS " +
ProductInfo.productInfo.TABLE_PRODUCTINFO); //갱신 시 테이블 삭제 후 다시 생성
        onCreate(db);
    }

    //삽입할 데이터 정의
    public void insertP(SQLiteDatabase db) {
        insertProduct(db, "A", "블루베리", "과일", 29000, "1kg");
        insertProduct(db, "A", "체리", "과일", 21000, "1kg");
        insertProduct(db, "B", "감", "과일", 17000, "10kg");
        insertProduct(db, "B", "유자", "과일", 33000, "5kg");
        insertProduct(db, "B", "키위", "과일", 44000, "10kg");
        insertProduct(db, "B", "사과", "과일", 39000, "10kg");
        insertProduct(db, "B", "자두", "과일", 19000, "5kg");
        insertProduct(db, "B", "복숭아", "과일", 23000, "4.5kg");
        insertProduct(db, "C", "멜론", "과일", 27000, "8kg");
        insertProduct(db, "C", "배", "과일", 36000, "15kg");
        insertProduct(db, "C", "매실", "과일", 22000, "10kg");
        insertProduct(db, "C", "석류", "과일", 30000, "5kg");
        insertProduct(db, "D", "귤", "과일", 15000, "5kg");
        insertProduct(db, "D", "한라봉", "과일", 16000, "2kg");
        insertProduct(db, "E", "딸기", "과일", 26000, "2kg");
        insertProduct(db, "E", "포도", "과일", 21000, "5kg");
        insertProduct(db, "F", "감자", "채소", 23000, "20kg");
        insertProduct(db, "F", "토마토", "채소", 22000, "10kg");
        insertProduct(db, "F", "더덕", "채소", 18000, "1kg");
        insertProduct(db, "F", "두릅", "채소", 32000, "1kg");
        insertProduct(db, "F", "배추", "채소", 7000, "10kg");
        insertProduct(db, "F", "옥수수", "채소", 12000, "8kg");
        insertProduct(db, "G", "시금치", "채소", 8000, "4kg");
        insertProduct(db, "G", "우엉", "채소", 12000, "2kg");
        insertProduct(db, "G", "은행", "채소", 17000, "1kg");
        insertProduct(db, "G", "참외", "채소", 45000, "10kg");
        insertProduct(db, "H", "냉이", "채소", 15000, "1kg");
```

```
        insertProduct(db, "H", "늙은 호박", "채소", 12000, "4kg");
        insertProduct(db, "H", "봄동", "채소", 10000, "1kg");
        insertProduct(db, "H", "쑥", "채소", 13000, "1kg");
        insertProduct(db, "I", "당근", "채소", 35000, "20kg");
        insertProduct(db, "I", "무", "채소", 20000, "18kg");
        insertProduct(db, "I", "갈치", "해산물", 30000, "1kg");
        insertProduct(db, "J", "고구마", "채소", 25000, "10kg");
        insertProduct(db, "J", "달래", "채소", 16000, "1kg");
        insertProduct(db, "J", "밤", "채소", 8000, "1kg");
        insertProduct(db, "J", "수박", "채소", 17000, "1 통");
        insertProduct(db, "K", "굴", "해산물", 20000, "1kg");
        insertProduct(db, "K", "도미", "해산물", 16000, "1kg");
        insertProduct(db, "K", "바지락", "해산물", 9000, "1kg");
        insertProduct(db, "K", "전어", "해산물", 13000, "1kg");
        insertProduct(db, "K", "고등어", "해산물", 33000, "2kg");
        insertProduct(db, "K", "과메기", "해산물", 20000, "10 마리");
        insertProduct(db, "K", "꽁치", "해산물", 10000, "1kg");
        insertProduct(db, "K", "민어", "해산물", 48000, "1kg");
        insertProduct(db, "L", "감태", "해산물", 10000, "1kg");
        insertProduct(db, "L", "꼬막", "해산물", 10000, "1kg");
        insertProduct(db, "L", "농어", "해산물", 50000, "3kg");
        insertProduct(db, "L", "대하", "해산물", 30000, "1kg");
        insertProduct(db, "L", "매생이", "해산물", 20000, "5 재기");
        insertProduct(db, "L", "메기", "해산물", 12000, "1kg");
        insertProduct(db, "L", "전복", "해산물", 70000, "1kg");
        insertProduct(db, "L", "홍합", "해산물", 10000, "5kg");
        insertProduct(db, "M", "방어", "해산물", 90000, "6kg");
        insertProduct(db, "M", "삼치", "해산물", 44000, "3kg");
        insertProduct(db, "M", "파래", "해산물", 3000, "1kg");
        insertProduct(db, "N", "게", "해산물", 25000, "1kg");
        insertProduct(db, "N", "주꾸미", "해산물", 28000, "1kg");
        insertProduct(db, "N", "키조개", "해산물", 13000, "1kg");
    }

    //insertP 에서 인자받아서 테이블에 삽입
    public void insertProduct(SQLiteDatabase db, String name, String product,
String type, int price, String unit) {
        ContentValues values = new ContentValues();
        //위 함수에 받은 정보를 각 애트리뷰트에 맞게 저장한다.
        values.put(ProductInfo.productInfo.COLUMN_SELLER, name);
        values.put(ProductInfo.productInfo.COLUMN_PRODUCT, product);
        values.put(ProductInfo.productInfo.COLUMN_TYPE, type);
        values.put(ProductInfo.productInfo.COLUMN_PRICE, price);
        values.put(ProductInfo.productInfo.COLUMN_UNIT, unit);
        //판매자정보 테이블에 values 값을 넣어준다.
        db.insert(ProductInfo.productInfo.TABLE_PRODUCTINFO, null, values);
    }

    /* sql 문 작성 시:
    columnindex 는 0 부터 시작인데 0 은 데이터베이스에서 목록의 수(id)를 나타냄 ex) 1
/ a / 전번 / 지역 <- 이런식으로
```

```
    그래서 꺼내고자하는 값은 1 부터 시작해야 한다~~~~~
    * */

    //판매품목에서 과일만 파는 판매자를 보고 싶을 때
    //보여주기 목록에는 과일,채소,해산물을 표기하는 type 은 출력하지 않아도 될 것 같아요
    public String selectSellF(){
        SQLiteDatabase db = getReadableDatabase();
        String result = "";
        Cursor cursor = db.rawQuery("SELECT * FROM productInfo WHERE type = '과일'", null); // 테이블의 전체 정보 삽입
        while(cursor.moveToNext()) {
            result += " 판매자: " + cursor.getString(1) //column(0) 은 id 정보이므로 1 부터 얻어온다.
                    +", " + cursor.getString(2)
                    + ", 가격: " + cursor.getString(4)
                    + ", 단위: " + cursor.getString(5)
                    +"\n";
        }
        return result;
    }
    //판매품목에서 채소만 파는 판매자를 보고 싶을 때
    public String selectSellV(){
        SQLiteDatabase db = getReadableDatabase();
        String result = "";
        Cursor cursor = db.rawQuery("SELECT * FROM productInfo WHERE type = '채소'", null); // 테이블의 전체 정보 삽입
        while(cursor.moveToNext()) {
            result += " 판매자: " + cursor.getString(1) //column(0) 은 id 정보이므로 1 부터 얻어온다.
                    +", " + cursor.getString(2)
                    + ", 가격: " + cursor.getString(4)
                    + ", 단위: " + cursor.getString(5)
                    +"\n";
        }
        return result;
    }
    //판매품목에서 해산물만 파는 판매자를 보고 싶을 때
    public String selectSellS() {
        SQLiteDatabase db = getReadableDatabase();
        String result = "";
        Cursor cursor = db.rawQuery("SELECT * FROM productInfo WHERE type = '해산물'", null); // 테이블의 전체 정보 삽입
        while(cursor.moveToNext()) {
            result += " 판매자: " + cursor.getString(1) //column(0) 은 id 정보이므로 1 부터 얻어온다.
                    +", " + cursor.getString(2)
                    + ", 가격: " + cursor.getString(4)
                    + ", 단위: " + cursor.getString(5)
                    +"\n";
        }
        return result;
```

```
    }
}
```

⑤ DBHelperSeller.java

```
package info.androidhive.sqlite.helper;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

import info.androidhive.sqlite.model.SellerInfo;


public class DBHelperSeller extends SQLiteOpenHelper {
    //Logcat tag
    public static final String LOG = "DatabaseHelper";
    //database version
    public static final int DATABASE_VERSION = 1;
    //database name
    public static final String DATABASE_NAME = "season.db";

    //create statement -과일
    //_id 는 생성했을 때 목록을 각각 넘버링해주는 주는 개념이다.
    //존재하지 않을 때만 생성
    public static final String CREATE_TABLE_SELLER = "CREATE TABLE if not
exists " + SellerInfo.sellerInfo.TABLE_SELLERINFO +
            " (" + SellerInfo.sellerInfo._ID + " INTEGER PRIMARY KEY
autoincrement, "
            + SellerInfo.sellerInfo.COLUMN_SELLER + " TEXT , " +
SellerInfo.sellerInfo.COLUMN_NUM + " TEXT , "
            + SellerInfo.sellerInfo.COLUMN_AREA + " TEXT )";

    //생성자를 호출하여 상속받는다. 상속해서 여기 클래스에 있는 함수를 사용하게끔
한다.
    public DBHelperSeller(@Nullable Context context, @Nullable String name,
@Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //테이블 생성
        db.execSQL(CREATE_TABLE_SELLER);
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //갱신 시 기존 테이블 삭제 후 다시 생성한다.
        db.execSQL("DROP TABLE IF EXISTS " +
SellerInfo.sellerInfo.TABLE_SELLERINFO);
        onCreate(db);
    }

    //삽입할 데이터를 정의한다.
    public void insertSellInfo(SQLiteDatabase db) {
        insertSell(db, "A", "010-1111-1111", "경기도");
        insertSell(db, "B", "010-2222-2222", "경상남도");
        insertSell(db, "C", "010-3333-3333", "전라남도");
        insertSell(db, "D", "010-4444-4444", "제주도");
        insertSell(db, "E", "010-55555-5555", "충청남도");
        insertSell(db, "F", "010-6666-6666", "강원도");
        insertSell(db, "G", "010-7777-7777", "경상북도");
        insertSell(db, "H", "010-8888-8888", "전라남도");
        insertSell(db, "I", "010-9999-9999", "제주도");
        insertSell(db, "J", "010-1010-1010", "충청북도");
        insertSell(db, "K", "010-1212-1212", "경상북도");
        insertSell(db, "L", "010-1313-1313", "전라북도");
        insertSell(db, "M", "010-1414-1414", "제주도");
        insertSell(db, "N", "010-1515-1515", "충청남도");
    }

    //insertSell 에서 받아서 테이블에 삽입
    public void insertSell(SQLiteDatabase db, String name, String num, String
area) {
        ContentValues values = new ContentValues();
        //위 함수에 받아온 정보를 각 애트리뷰트에 맞게 저장한다.
        values.put(SellerInfo.sellerInfo.COLUMN_SELLER, name);
        values.put(SellerInfo.sellerInfo.COLUMN_NUM, num);
        values.put(SellerInfo.sellerInfo.COLUMN_AREA, area);
        //판매자정보 테이블 values 값을 넣어준다.
        db.insert(SellerInfo.sellerInfo.TABLE_SELLERINFO, null, values);
    }

    /* sql 문 작성시!!! 필요한 정보
    columnindex 는 0 부터 시작인데 0 은 데이터베이스에서 목록의 수(id)를 나타냄 ex) 1
/ a / 전번 / 지역 <- 이런식으로
    그래서 꺼내고자하는 값은 1 부터 시작해야 한다~~~~~
    * */
    public String selectSellerInfo() { //잘 저장됐는지 확인용
        SQLiteDatabase db = getReadableDatabase(); //저장된 데이터 읽어오는 함수
        String result = "";

        Cursor cursor = db.rawQuery("SELECT * FROM " +
SellerInfo.sellerInfo.TABLE_SELLERINFO, null); //과일 테이블의 전체 정보 삽입
        while(cursor.moveToNext()) {
```

```
            result += " " + cursor.getString(1) //column(0) 은 id 정보이므로
1 부터 얻어온다.
                    +", 연락처: " + cursor.getString(2)
                    + ", 지역: " + cursor.getString(3)
                    +"\n";
        }
        return result;
    }
}
```

Model

① Fruit.java

```
package info.androidhive.sqlite.model;

import android.provider.BaseColumns;

//과일 테이블 정의
public final class Fruit {

    private Fruit(){ }

    public static class fruit implements BaseColumns {
        public static final String TABLE_FRUIT = "fruit";  //테이블 이름
        public static final String COLUMN_NAMEF = "name"; //과일 이름 attribute
        public static final String COLUMN_STARTF = "start"; //제철 시작
attribute
        public static final String COLUMN_FINISHF = "finish"; //제철 끝
attribute
    }
}
```

② Seafood.java

```
package info.androidhive.sqlite.model;

import android.provider.BaseColumns;

//해산물 테이블 정의
public final class Seafood {

    //constructors
    public Seafood() {
    }

    public static class seafood implements BaseColumns {
```

```
        public static final String TABLE_SEAFOOD = "seafood"; //테이블 이름
        public static final String COLUMN_NAMES = "name"; //해산물 이름
attribute
        public static final String COLUMN_STARTS = "start"; //해산물 시작 개월
attribute
        public static final String COLUMN_FINISHS = "finish"; //해산물 끝 개월
attribute
    }
}
```

③ Vegetable.java

```
package info.androidhive.sqlite.model;

import android.provider.BaseColumns;

//채소 테이블 정의
public final class Vegetable {

    //constructors
    public Vegetable() { }

    public static class vegetable implements BaseColumns {
        public static final String TABLE_VEGETABLE = "vegetable"; //채소 테이블
이름
        public static final String COLUMN_NAMEV = "name"; //채소 이름 attribute
        public static final String COLUMN_STARTV = "start"; //제철 시작
attribute
        public static final String COLUMN_FINISHV = "finish"; //제철 끝
attribute
    }
}
```

④ ProductInfo.java

```
package info.androidhive.sqlite.model;

import android.provider.BaseColumns;

public final class ProductInfo {

    //constructors
    public ProductInfo() {}

    public static class productInfo implements BaseColumns {
        public static final String TABLE_PRODUCTINFO = "productInfo"; //판매품목
테이블 이름 정의
```

```
        public static final String COLUMN_SELLER = "seller"; //판매자 이름
attribute
        public static final String COLUMN_PRODUCT = "product"; //품목 이름
attribute
        public static final String COLUMN_TYPE = "type"; //품목에 대한 유형(과일,
해산물, 채소) attribute
        public static final String COLUMN_PRICE = "price"; //가격 attribute
        public static final String COLUMN_UNIT = "unit"; //판매 단위 attribute
    }
}
```

⑤ SellerInfo.java

```
package info.androidhive.sqlite.model;

import android.provider.BaseColumns;

//판매자 개인정보 테이블 정의
public final class SellerInfo {

    //constructors
    public SellerInfo() {}

    public static class sellerInfo implements BaseColumns {
        public static final String TABLE_SELLERINFO = "sellerInfo"; //테이블
이블
        public static final String COLUMN_SELLER = "seller"; //판매자 이름
attribute
        public static final String COLUMN_NUM = "phoneNum"; //판매자 전화번호
attribute
        public static final String COLUMN_AREA = "area"; //판매자 지역 attribute
    }
}
```

Layout

해당 레이아웃들이 어플리케이션 실행 시 어떻게 출력되는지는 발표 영상에서 어플리케이션을 시행하여 다룰 예정이다.

① activity_home.xml

Mainactivity.java 에 대응하는 레이아웃 파일이다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#907A5D"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="196dp"
        android:layout_height="194dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.275"
        app:srcCompat="@drawable/food" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/font"
        android:text="제철제철"
        android:textColor="#8DB640"
        android:textSize="48sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView"
        app:layout_constraintVertical_bias="0.0" />

    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="#46312C"
        android:fontFamily="@font/font"
        android:text="시작하기"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.221" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

② activity_menu.xml

Menu.java 에 대응하는 레이아웃 파일이다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EED0AA"
    tools:context=".Menu">

    <Button
        android:id="@+id/button_Items"
        android:layout_width="320dp"
        android:layout_height="200dp"
        android:backgroundTint="#68842B"
        android:drawableTop="@drawable/groceries0"
        android:fontFamily="@font/font"
        android:text="제철음식을 \n 알고싶어요 "
        android:textColor="#FAFFFFFF"
        android:textSize="32sp"
        android:textStyle="bold"
        app:icon="@drawable/groceries0"
        app:iconTint="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.494"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.092" />

    <Button
        android:id="@+id/button_AllSellers"
        android:layout_width="320dp"
        android:layout_height="200dp"
        android:backgroundTint="#46312C"
        android:drawableTop="@drawable/truck"
        android:fontFamily="@font/font"
        android:text="모든 판매자 \n 리스트"
        android:textColor="#FFFFFF"
        android:textSize="32sp"
        android:textStyle="bold"
        app:icon="@drawable/list01"
        app:iconTint="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.494"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.913" />

    <Button
        android:id="@+id/button_Sellers"
        android:layout_width="320dp"
        android:layout_height="200dp"
        android:backgroundTint="#F8B62C"
        android:drawableTop="@drawable/truck"
        android:fontFamily="@font/font"
        android:text="구매하고 \n 싶어요"
        android:textColor="#FFFFFF"
        android:textSize="32sp"
        android:textStyle="bold"
        app:icon="@drawable/truck"
        app:iconTint="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.494"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.499" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

③ activity_items.xml

Items.java 에 대응하는 레이아웃 파일이다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EED0AA"
    tools:context=".Items">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
```

```
        android:background="#7768842B"
        android:fontFamily="@font/font"
        android:text="제철음식을 알고 싶어요 "
        android:textAlignment="textEnd"
        android:textColor="#46312C"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="86dp"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button_Fruit"
            android:layout_width="wrap_content"
            android:layout_height="70dp"
            android:layout_weight="1"
            android:backgroundTint="#46312C"
            android:fontFamily="@font/font"
            android:onClick="clickFruit"
            android:text="과일"
            android:textColor="#FFFFFF"
            android:textSize="30sp" />

        <Button
            android:id="@+id/button_Vegetable"
            android:layout_width="wrap_content"
            android:layout_height="70dp"
            android:layout_weight="1"
            android:backgroundTint="#46312C"
            android:fontFamily="@font/font"
            android:onClick="clickVegetable"
            android:text="채소"
            android:textColor="@color/white"
            android:textSize="30sp" />

        <Button
            android:id="@+id/button_Seafood"
            android:layout_width="wrap_content"
            android:layout_height="70dp"
            android:layout_weight="1"
            android:backgroundTint="#46312C"
            android:fontFamily="@font/font"
            android:onClick="clickSeafood"
            android:text="해산물"
            android:textColor="@color/white"
            android:textSize="30sp" />
```

```
        </LinearLayout>

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical">

                <TextView
                    android:id="@+id/textView_Items"
                    android:layout_width="match_parent"
                    android:layout_height="700dp"
                    android:fontFamily="@font/font"
                    android:text="text"
                    android:textColor="@color/black"
                    android:textSize="24sp"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintHorizontal_bias="0.496"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/button_Fruit" />
            </LinearLayout>
        </ScrollView>


    </LinearLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

④ activity_sellers.xml

Sellers.java 에 대응하는 레이아웃 파일이다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EED0AA"
    tools:context=".Sellers">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:background="#70F8B62C"
        android:fontFamily="@font/font"
        android:text="구매하고 싶어요  "
        android:textAlignment="textEnd"
        android:textColor="#46312C"
        android:textSize="24sp"
        android:textStyle="bold" />


<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="86dp"
    android:orientation="horizontal">


    <Button
        android:id="@+id/button_FruitSeller"
        android:layout_width="wrap_content"
        android:layout_height="70dp"
        android:layout_weight="1"

        android:backgroundTint="#46312C"
        android:fontFamily="@font/font"
        android:text="과일"
        android:textColor="#FFFFFF"
        android:textSize="30sp" />

    <Button
        android:id="@+id/button_VegetableSeller"
        android:layout_width="wrap_content"
        android:layout_height="70dp"
        android:layout_weight="1"

        android:backgroundTint="#46312C"
        android:fontFamily="@font/font"
        android:text="채소"
        android:textSize="30sp" />


    <Button
        android:id="@+id/button_SeafoodSeller"
        android:layout_width="wrap_content"
        android:layout_height="70dp"
```

```
            android:layout_weight="1"

            android:backgroundTint="#46312C"
            android:fontFamily="@font/font"
            android:text="해산물"
            android:textSize="30sp" />
    </LinearLayout>


        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical">

                <TextView
                    android:id="@+id/textView_sellers"
                    android:layout_width="match_parent"
                    android:layout_height="700dp"
                    android:fontFamily="@font/font"
                    android:text="TextView"
                    android:textSize="24sp" />
            </LinearLayout>
        </ScrollView>

    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

⑤ activity_all_sellers.xml

AllSellers.java 에 대응하는 레이아웃 파일이다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EED0AA"
    tools:context=".AllSellers">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
```

```
        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:background="#8546312C"
            android:fontFamily="@font/font"
            android:text="모든 판매자 리스트  "
            android:textAlignment="textEnd"
            android:textColor="#46312C"
            android:textSize="24sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="1.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.32" />

        <TextView
            android:id="@+id/textView_AllSellers"
            android:layout_width="match_parent"
            android:layout_height="682dp"
            android:fontFamily="@font/font"
            android:text="TextView"
            android:textSize="24sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.457"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.617" />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```