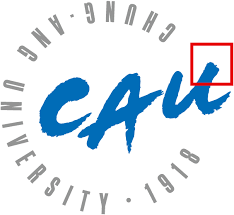
Data Structure Design(02)\_ Implementation

Individual Project – 유동인구와 미세먼지 그리고 관심사를

이용한 스마트 데이트 코스 추천 Application



|  |  |
| --- | --- |
| Department | Computer Science Engineering |
| Name(Student Number) | Kwon Dokyeong (20154077) |

# Project Description

## Project’s Overview

DDARAWA – 유동인구와 미세먼지 그리고 관심사를 이용한 스마트 데이트 코스 추천 App

## Projects’ Limits

* 데이트 장소



Figure 1

데이트 장소는 그래프의 복잡성에 대한 문제를 간소화 하기 위해, “데이트팝”이라는 어플이 제공하는 장소 중 각 카테고리별 5개씩으로 한정 짓는다. 그리고 더 나아가 서울시 각 행정구역에 대한 영화관 정보도 추가하였다.

* 데이트 시간

데이트 시간에서 평균적으로 12:00~14:00(PM)를 점심 식사 시간, 18:00~20:00(PM)은 저녁 식사 시간이라 가정한다.

* 인구 데이터

이 어플리케이션은 데이트를 하는 커플들을 대상으로 하고 있다. 특히 주 타겟층은 20-30대로 한정 지을 것이므로, 서울시 인구 Data중, 20-30대의 인구 데이터만을 이용한다.

* 노드(장소)간 시간(Edge)

정보 제공의 시간과 데이터베이스의 양을 절약하기 위해 두 장소간 거리를 x,y좌표로 계산하여 어림잡은 직선거리의 시간을 제공 한다.

## Project’s Environments



Figure 2

* Software Tool : Android Studio – API 21 : Android5.0(Lolipop)
* Device Environment

기기 명 : Galaxy Note Fan Edition

안드로이드 버전 : 8.0.0

안드로이드 Application개발이 최종 목적이므로 툴은 Android Studio를 사용 하였다.

## Project’s Advantage

* FUNNY : 평소 가던 데이트 장소만 가기엔 지루할 때, 색다른 데이트 장소를 찾아볼 수 있는 Application이다.
* CONVINIOUS : 데이트 코스를 짜기 귀찮은 사람들에게 편리함을 줄 수 있는 Application이다.
* DIFFERENCE : 기존에 있는 데이트 장소 홍보성 Application보다는 사용자들에게 편리함을 주고자 하는데 목적을 두고 있다.

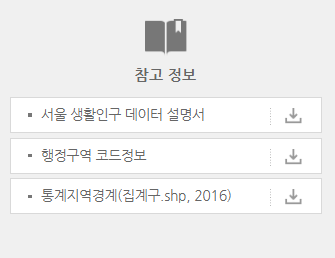
# Data Analysis

## Data Analysis’s Setting



Figure 3

Figure 4



안드로이드에서 통신을 위해서 정보를 받아오는 형식은 JSON을 선택하였다. 그래서 API에서 받아오는 정보를 테스팅 하기 위해 POSTMAN이라는 소프트웨어를 이용하여 테스트 하고 JSON 형식을 확인하였다. 또한, 받아온 정보 중 미세먼지 데이터에는 ‘측정소명’ 그리고 인구정보 데이터에는 ‘행정구역 코드 정보’가 있었는데, 이 코드에 대한 정보는 Figure 4에 있는 서울 열린데이터 광장의 참고 정보를 활용하였다.

## Population Data Analysis



Figure 5

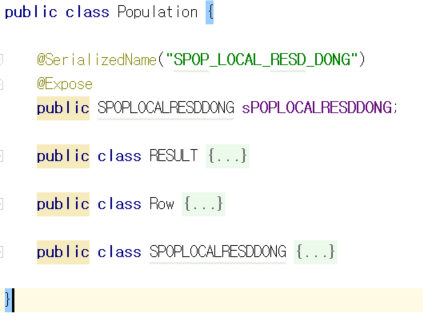
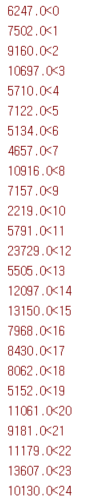
 

Figure 6

먼저 Figure 5는 유동인구에 대한 데이터를 Postman으로 받아 온 결과이다. 이 결과를 바탕으로 필요한 Class를 Figure 6과 같이 Class화 하여 사용하였으며 실제적으로 받아 온 결과는 Figure 6의 오른쪽과 같이 총 25개의 행정구역의 인구 정보를 받아 올 수 있었다.

## Fine Dust Data Analysis

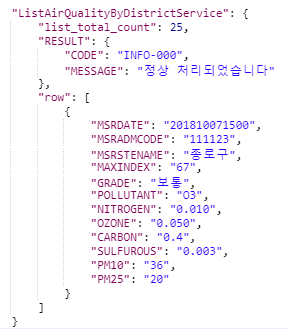


Figure 7



Figure 8

Figure 7은 서울시의 각 행정코드에서 미세먼지를 받아 온 결과이다. 본 프로젝트에서는 이 내용 중 GRADE항목만 사용하였다. 그리고 Figure 7에 대한 내용을 Class화 한 내용은 Figure 8에 있다. 마지막으로 Figure 8에 있는 Retrofit\_City는 실제로 안드로이드에서 통신을 잘 받았음을 보여주는 그림이다.

## Seoul Fine Dust Data Analysis

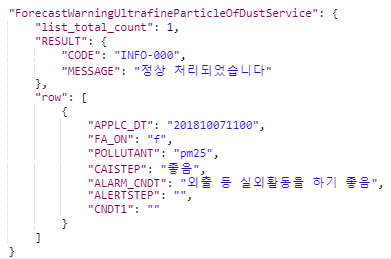


Figure 9

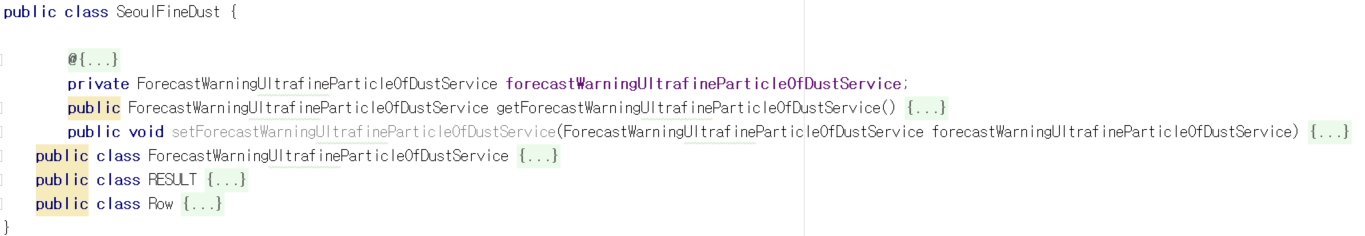


Figure 10

Figure 9는 서울에 대한 전체적 미세먼지 정보를 받아 온 결과이다. C에서 언급한 API에서 각 측정소 명 중 ‘점검중’이라고 띄우는 문제가 있었다. 이 문제를 해결하기 위해 Figure 9와 같이 서울시 평균 미세먼지 정보를 받아와 ‘점검중’인 경우 이 정보로 대체하기로 결정하였다.

Figure 10은 Figure 9의 내용을 Class화 한 결과이다. 그리고 Retrofit\_Seoul과 같이 실제로 통신이 잘 되었음을 확인할 수 있다.

## City

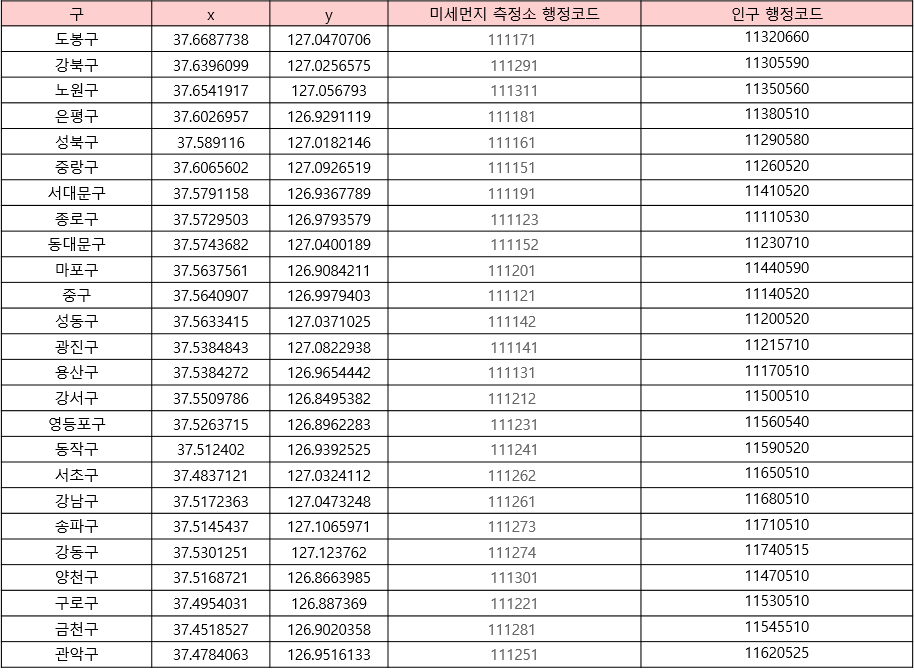


Figure 11

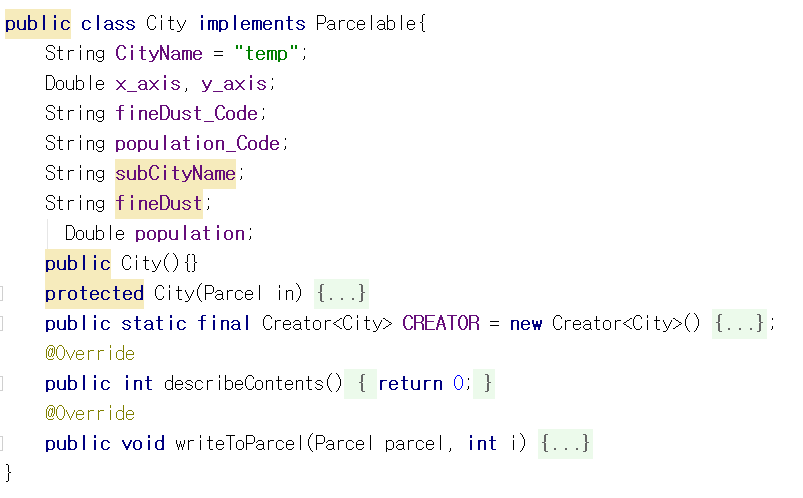


Figure 12

Figure 11은 실제로 서울시 행정 구역에 대한 정보를 모은 엑셀 파일을 캡쳐 해 온 결과이다. 그리고 이 내용을 실제로 Class화 한 것은 Figure 12에서 확인할 수 있다.

## Place

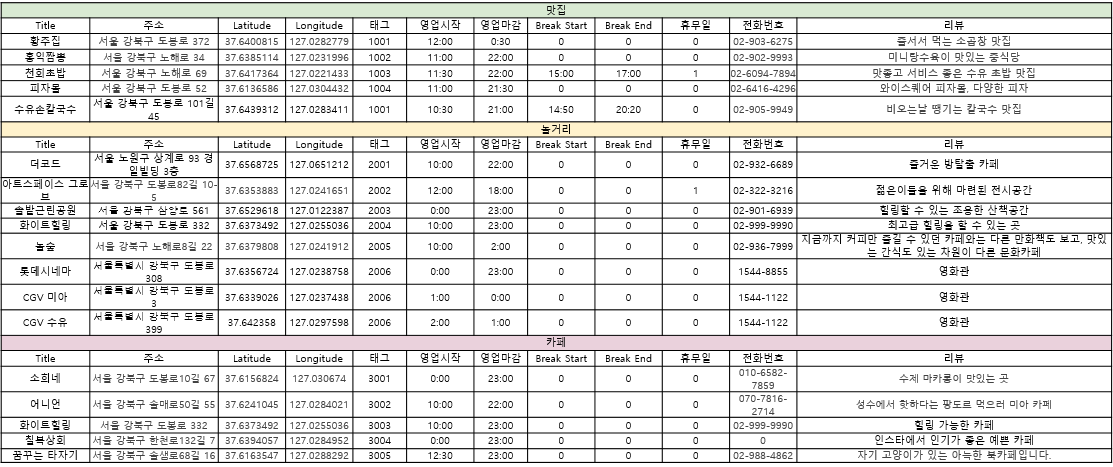


Figure 13

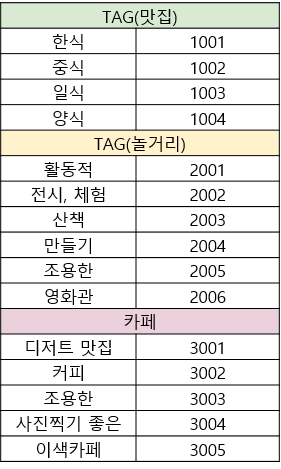


Figure 14

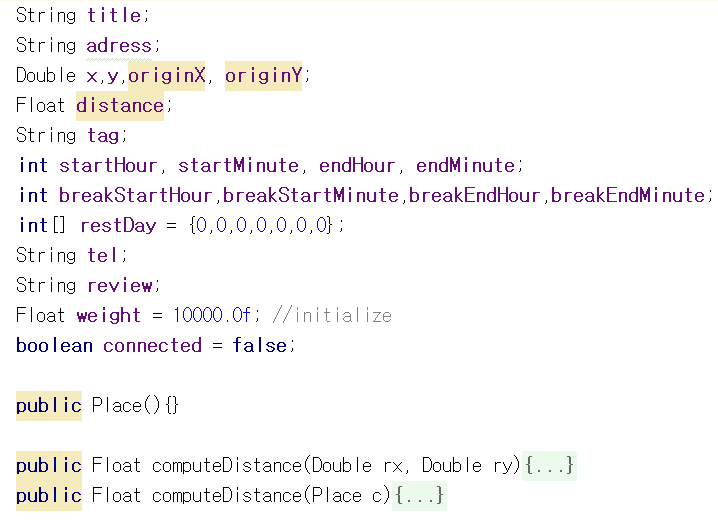


Figure 15

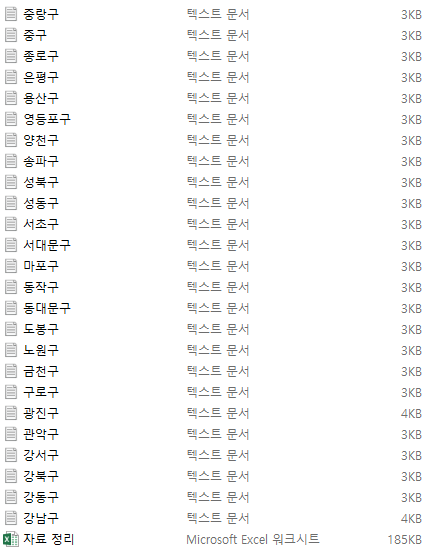


Figure 16

Figure 13은 데이트 장소에 대한 자료를 엑셀로 정리한 결과이다. 이 결과는 서울시 많은 행정구역 중 ‘강북구’에 대한 예시이다. Tag에 대한 정보는 Figure 14에 나와있다. 이는 구현 중 사용자가 선호하는 데이트 Case에 대한 Weight을 주기 위한 Tag정보이다. 그리고 Figure 15는 Figure 13의 내용을 Class화 한 결과이다. 모든 행정구역에 대한 데이트 장소는 Figure 16과 같이 text문서를 작성하여 저장하였다.

# Implementation

## Scenario 기반 전체 Flow 설명

사진으로 설명하기 앞서, 전체적인 흐름을 간략하게 설명하면 다음과 같다.

1. 서울시 25개 구역에 대한 유동 인구 데이터를 받아와 City Class에 입력
2. 사용자의 현재 위치를 받아 옴
3. 사용자의 위치와 유동인구 정보를 바탕으로 최적의 행정구역을 선택
4. 해당 행정구역에 대한 미세먼지 정보를 받아 옴
5. 사용자의 Tag 선택 정보를 받아 옴
6. Tag 와 미세먼지를 바탕으로 최적의 데이트 코스 추천

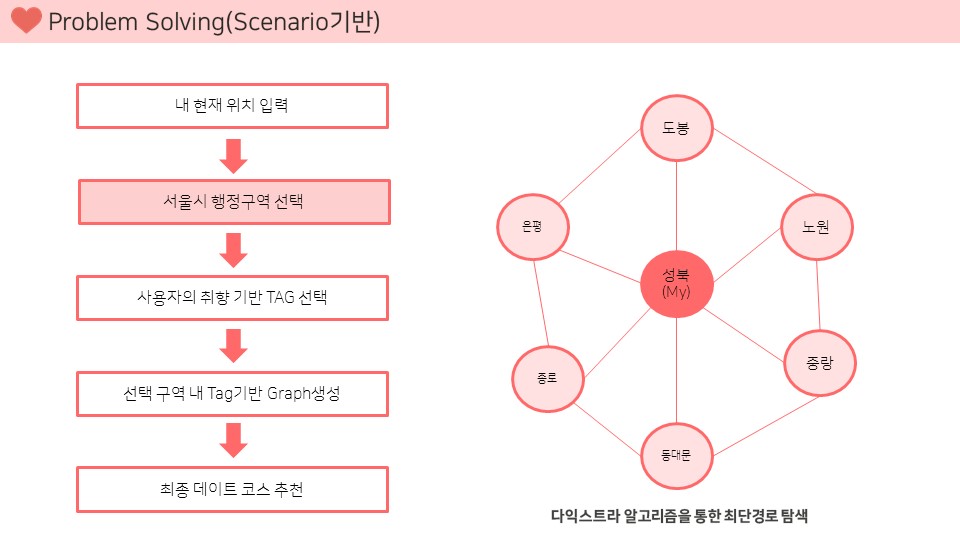


Figure 17

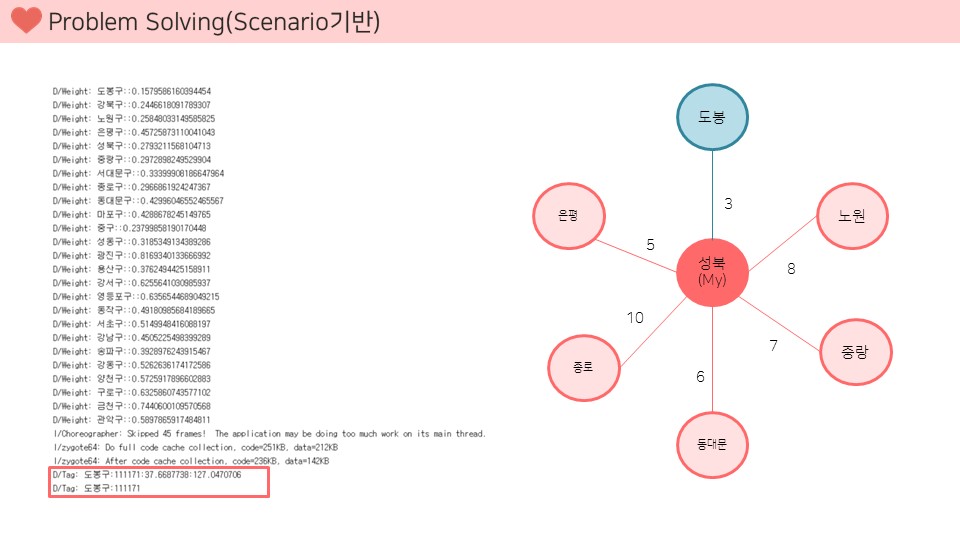


Figure 18

Figure 17, Figure 18은 사용자의 위치를 받아 최적의 행정구역을 선택하기 위한 상황을 표현한 것이다. Figure 17과 같이 현재 위치를 선택하게 되면 현재 위치로부터 떨어진 거리와 각 구의 20~30대 유동인구 정보를 바탕으로 Figure 18의 오른쪽 같이 Graph를 그릴 수 있다. Figure 18의 왼쪽 정보는 실제 Android에서 테스트를 해본 결과이다. ‘D/Weight’은 Weight에 관한 정보를 표현하고 있다.

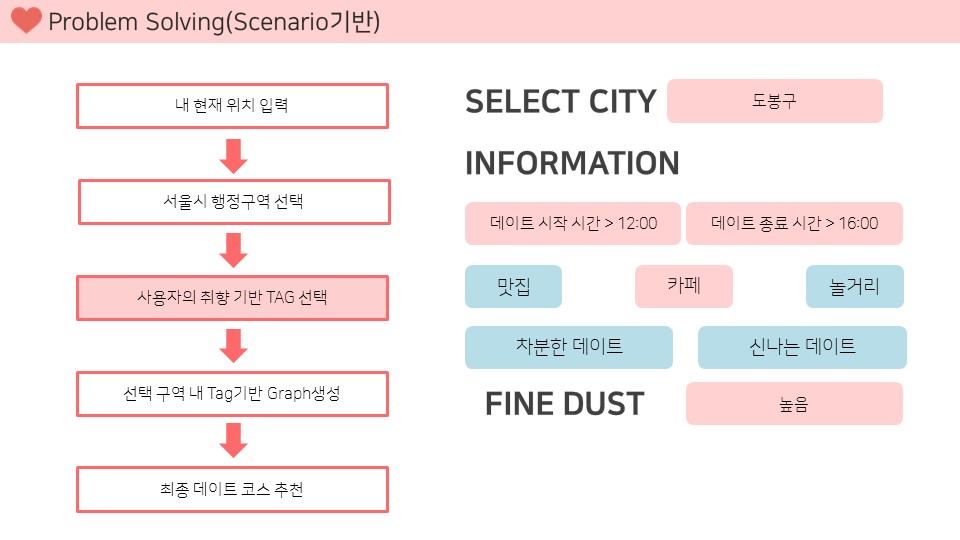


Figure 19

Figure 19는 사용자의 현재 위치를 바탕으로 정한 최적의 위치로부터 미세정보를 받아오고, 또한 사용자의 데이트에 관한 기본적 정보와 Tag를 받은 상황을 그림으로 표현 한 것이다. Figure 19와 같이 Tag정보가 입력되면 이 정보를 바탕으로 최적의 코스를 정하게 된다.

최적의 코스를 정하는 Flow는 아래와 같다.

1. Category에 대한 우선순위 결정을 통해 Start Node 선택
2. Start Node기반 장소 추천
3. 최종 Result 저장

본 프로젝트를 진행하면서 Category에 대한 정보를 조금 더 유용하게 사용할 수 있는 방법에 대해 생각을 해 보았고, ‘맛집, 카페, 놀거리’에 대한 우선순위를 부여하고자 하였다. 우선순위를 결정하는 Algorithm은 E. Algorithm부분에서 참고할 수 있다.

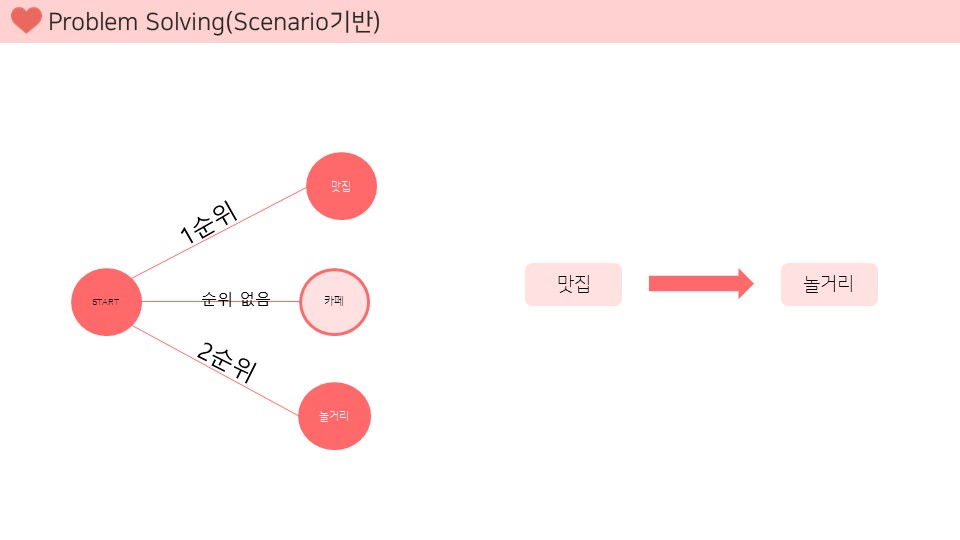


Figure 20

먼저 Figure 20은 순위를 결정하는 상황을 그림으로 나타낸 것이다. 위 그림과 같은 상황이 되면, 맛집을 들린 후 놀거리를 즐기러 갈 수 있도록 데이트 경로를 짜주는 것이 본 프로젝트의 최종 목표이다.

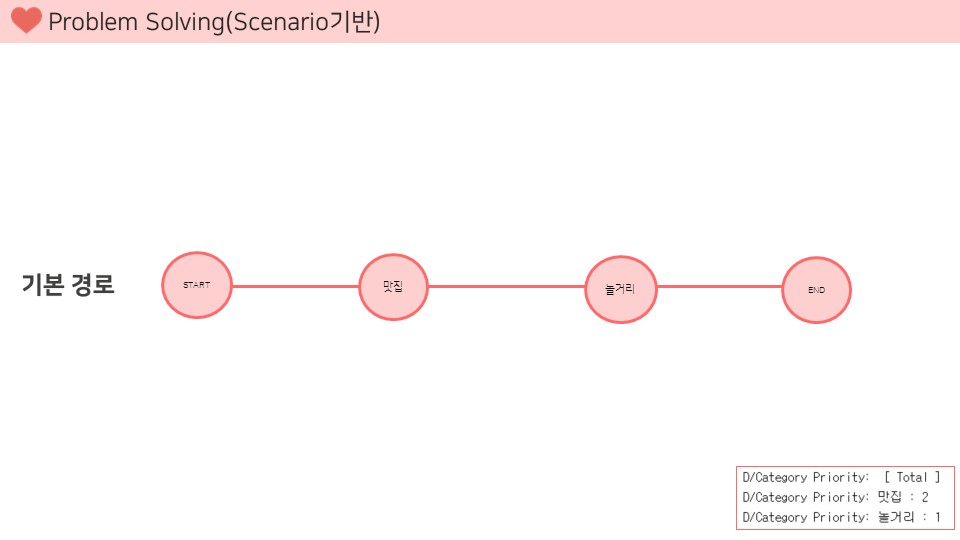


Figure 21

Figure 21은 Figure20의 상황에서 Base 경로를 표현 한 것이다. Start Node부터 End Node까지 이어지며, 중간에 맛집과 놀거리에 대한 장소가 선택이 될 것이다. 실제로 코딩한 결과는 Figure 21의 오른쪽 하단과 같이 우선순위가 표현되고 있다.

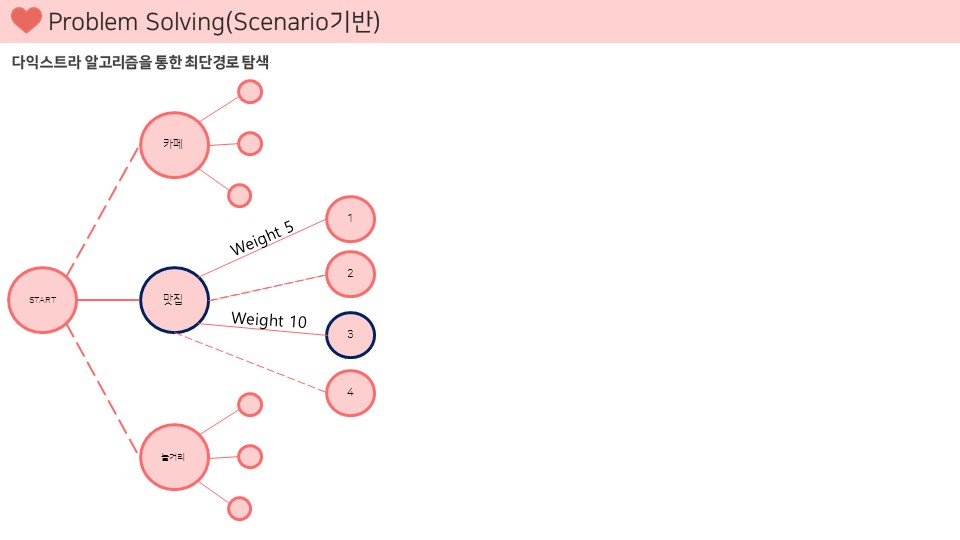


Figure 22

Figure 22에는 맛집에 대한 Node 중 최적의 데이트 장소 Node를 선택하는 상황이다. 점선으로 표시된 경우는 데이트 시간과 가게의 영업 시간이 맞지 않는 상황들이나 tag정보와 맞지 않은 것을 connection하지 않겠다는 것을 그림으로 표현 한 것이고, 이 부분이 Dynamic한 것을 잘 표현해주고 있다는 것을 확인할 수 있다.

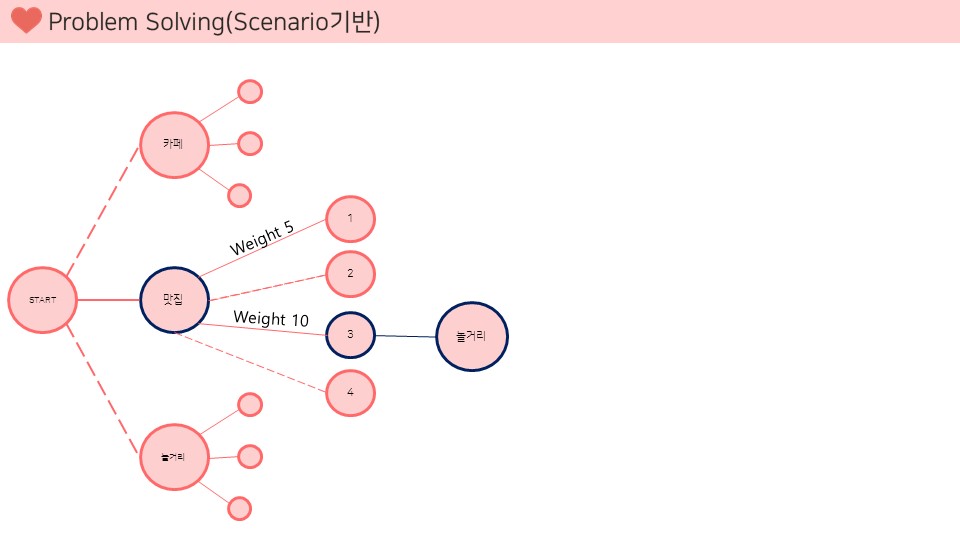


Figure 23

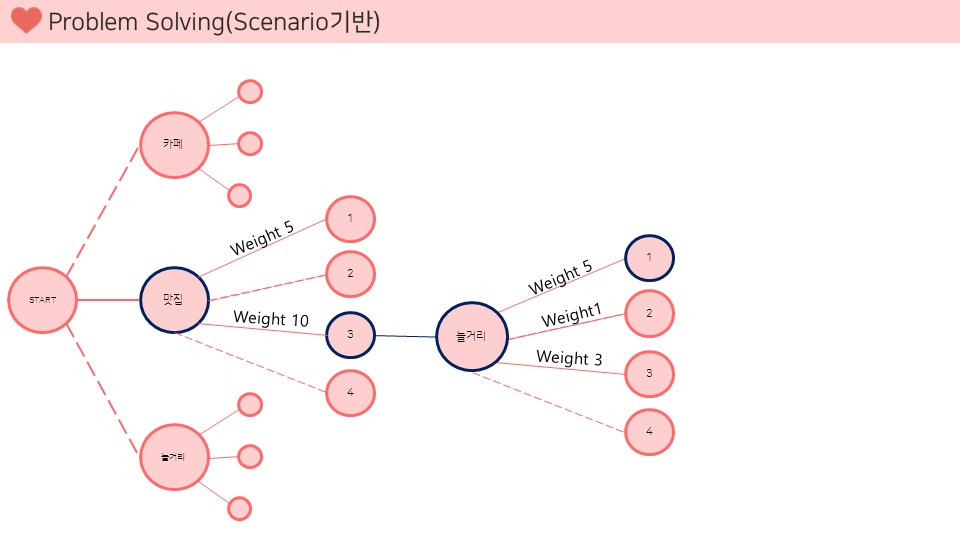


Figure 24

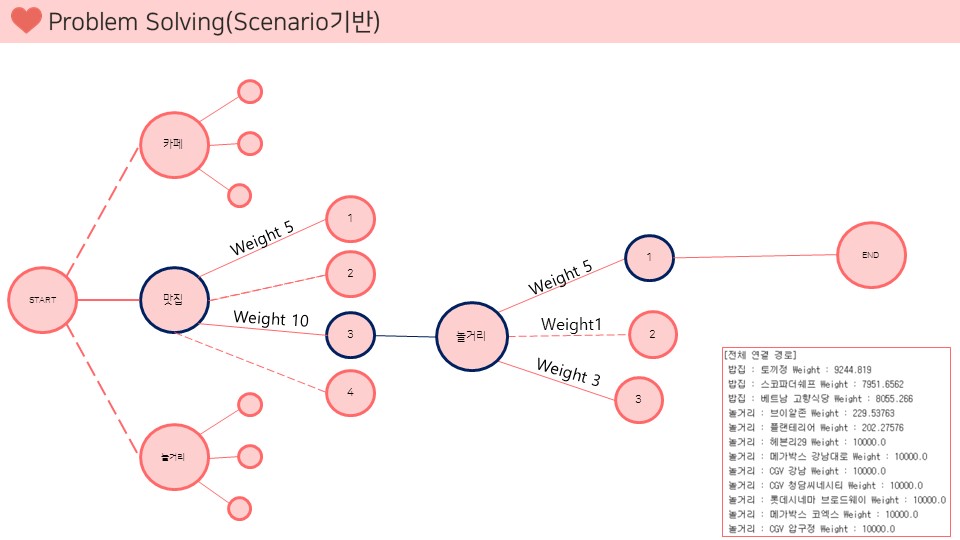


Figure 25

Figure 23~ Figure 25 역시 Figure 22에서 설명한 내용과 같은 것을 반복하며 End node까지 접근하는 상황을 그림으로 표현 한 것이다. 실제로 Figure 25의 오른쪽 하단과 같이 전체적으로 연결되는 그래프의 경로를 printGraph()라는 함수를 만들어 출력해보았다.

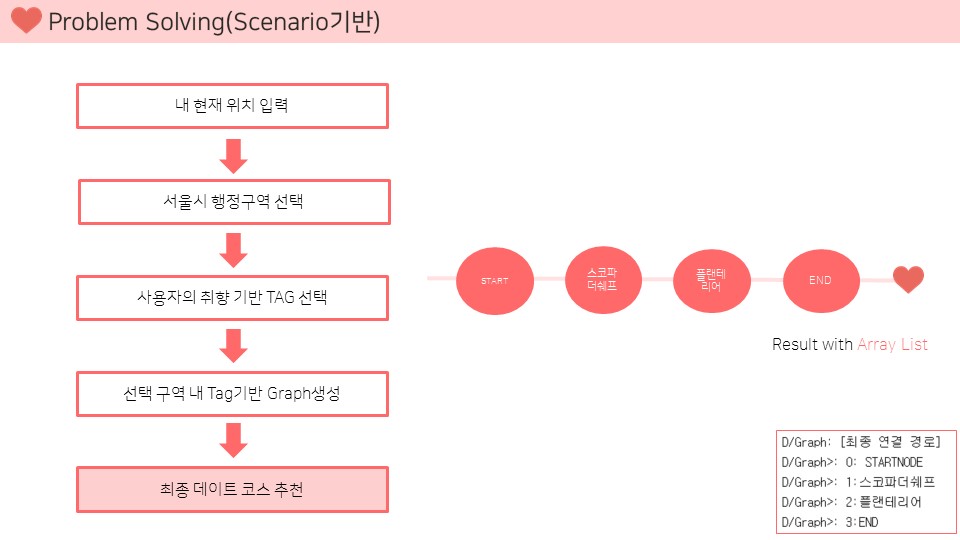


Figure 26

마지막으로 최적 루트에 대한 정보는 ResultNode라는 Class에 저장이 되어 결과를 출력하기 쉽게 하였다. Figure 26의 오른쪽 하단과 같이 최종 연결 경로를 프린트 할 수도 있으며 어플리케이션에도 띄워지는 것을 확인할 수 있다.

## Class UML

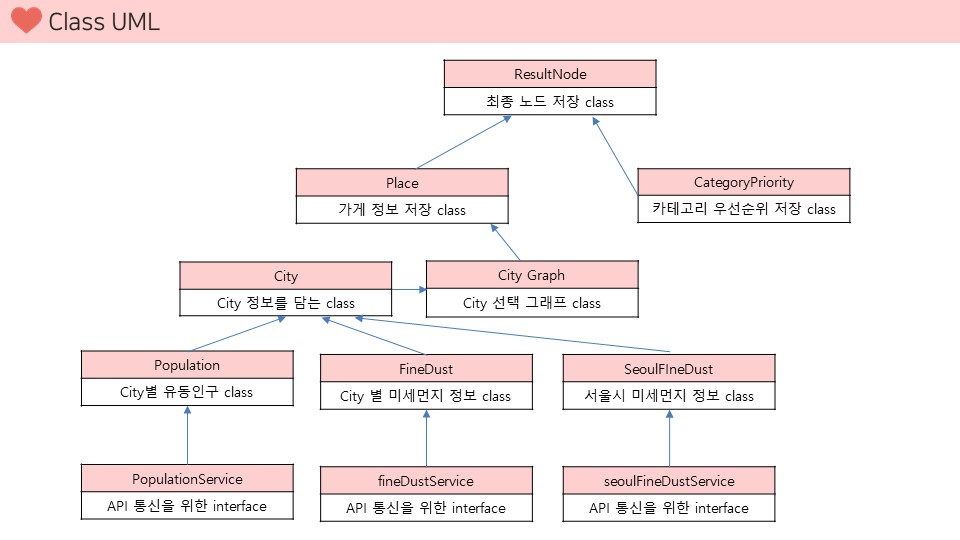


Figure 27

Figure 27은 전체적인 class에 대한 구조를 그림으로 표현 한 것이다. Android로 작성을 하여 화면에 표현하는 부분인 Activity는 제외하고 서로 연관이 있는 클래스들을 그림을 통해 표현하였다.

## Class Description

1. API Class

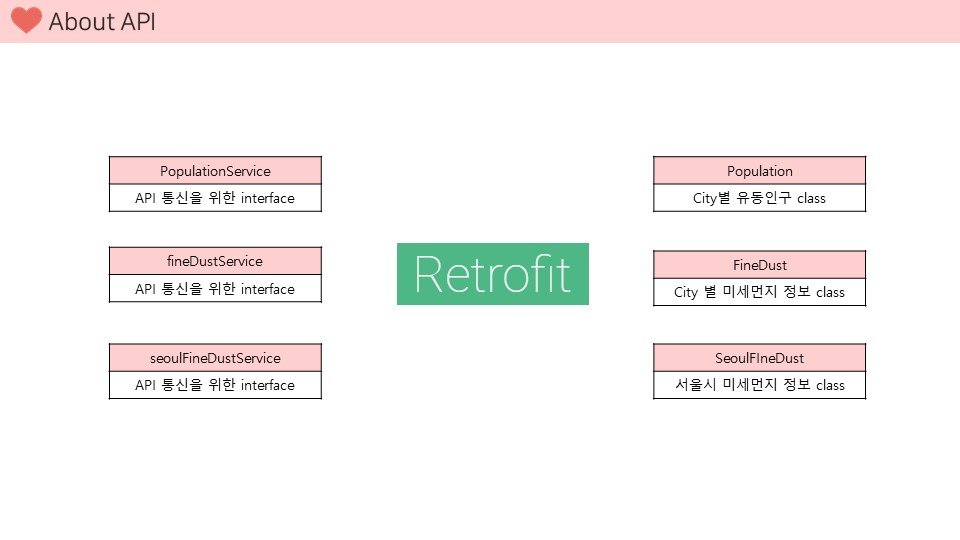


Figure 28

서울시로부터 총 3개의 공공데이터를 받아 오기 때문에 각 API당 2개씩의 클래스가 필요했다. 통신을 쉽게 하기 위해 Retrofit이라는 라이브러리를 이용하여 코딩을 진행하였으며 Http통신을 위한 Interface하나와 정보 저장 Class로 구성되어 있다.

1. City

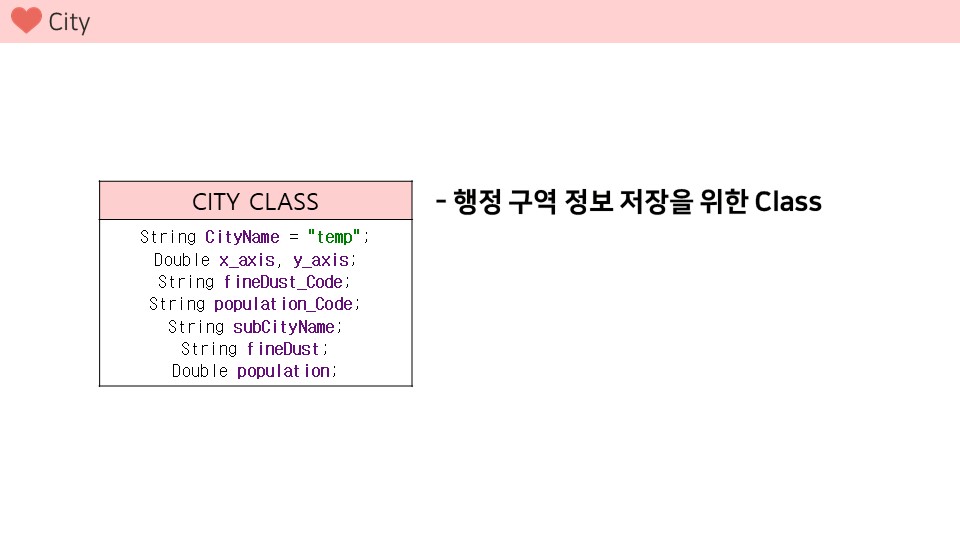


Figure 29

행정 구역 정보를 저장하기 위한 Class에 대한 자료는 Figure 29와 같다. 도시 이름과 위도 경도, 그리고 인구와 미세먼지 데이터를 저장할 수 있다.

1. City Graph

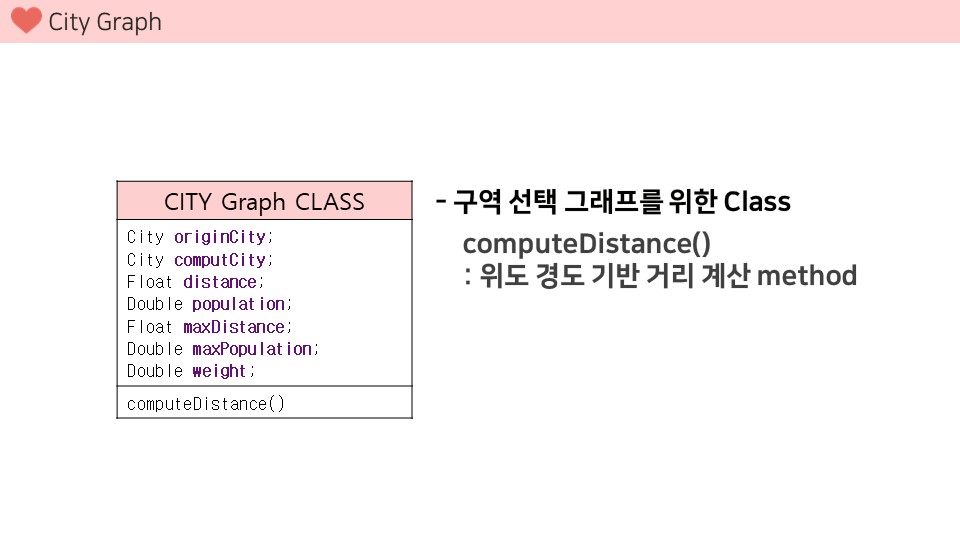


Figure 30

Figure 30은 City에서 최적의 행정구역을 선택할 때 그래프를 그리기 위해 필요한 Class이다. Origin City가 현재 사용자의 위치이고, 다음 Node는 computeCity에 들어 있으며, 두 Node 간 거리를 구하기 위한 computeDistance()라는 메소드도 구현을 해 놓았다.

1. Place

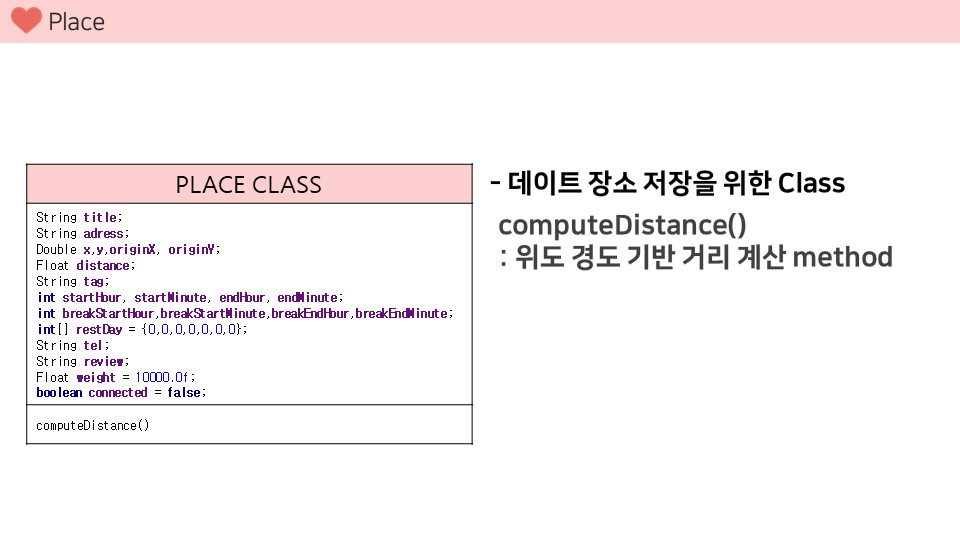


Figure 31

Figure 31에 나와있는 Place Class는 각 행정구역별 데이트 장소 저장을 위한 Class이다. 상호 명과 주소, 위도와 경도 그리고 tag정보, 영업시간, Break Time 등 가게와 관련된 정보를 저장할 수 있다. 그리고 computeDistance()라는 메소드를 통해 현 장소로부터 다른 장소까지의 거리를 계산할 수 있도록 하였다.

1. Category Priority

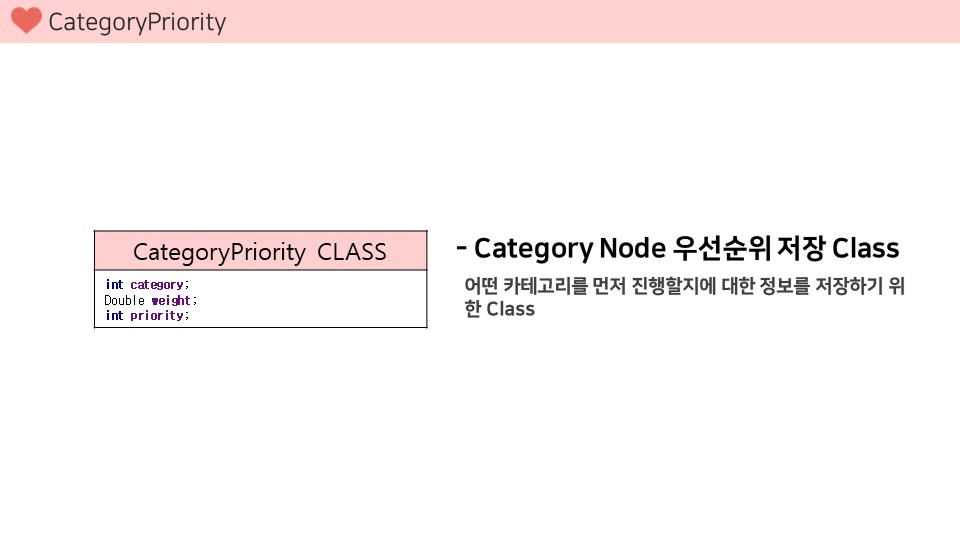


Figure 32

Figure32는 위에서 언급한 Category 우선순위를 저장하기 위한 Class를 그림으로 표현 한 것이다. 이 Class에서는 하나의 category에 대한 우선순위와 가중치를 저장할 수 있다.

1. Result Node

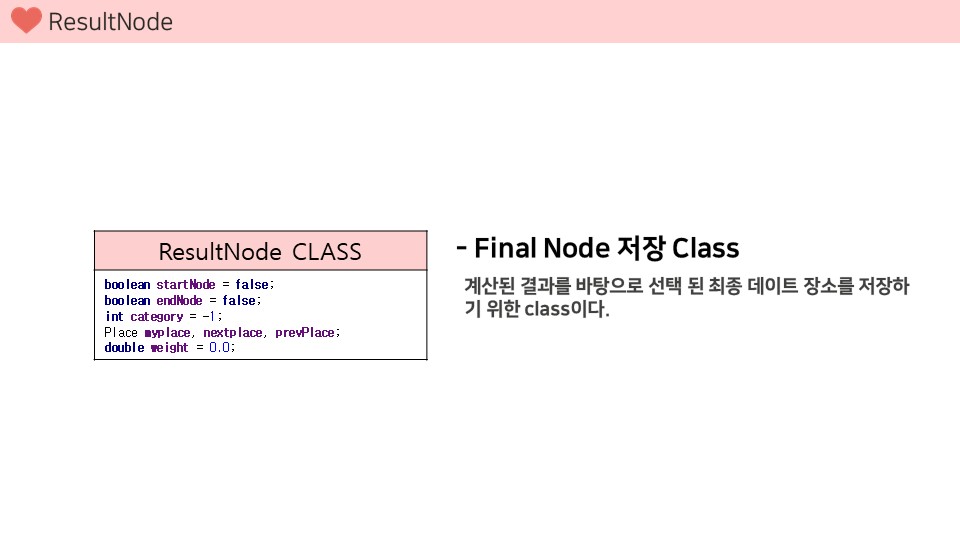


Figure 33

Figure 33은 최종 결과를 저장하기 위한 Class로 이전 Node와 다음 Node에 대한 정보와 현재 노드에 대한 정보를 담고 있는 것을 보여준다.

## Activity Description

1. Loading Activity



Figure 34

Figure 34는 Application 실행 시 가장 먼저 실행되는 Activity를 나타낸다. 이 Activity로부터 25개의 행정구역 정보와 유동인구 정보를 받아온다.

1. Main Activity

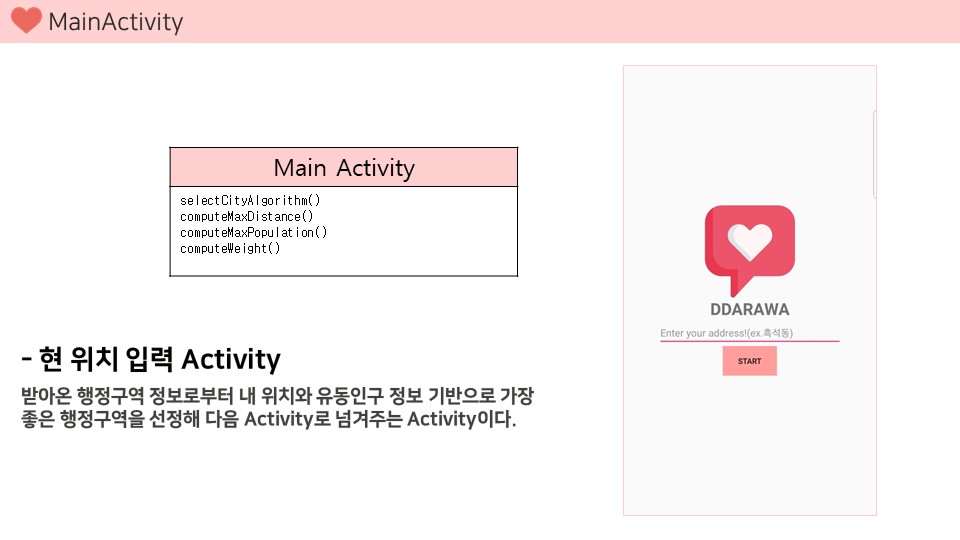


Figure 35

Figure 35는 사용자가 현재의 위치를 입력하는 Activity이다. 이 Activity로부터 입력 받은 정보를 바탕으로 최적의 행정 구역을 선택하여 다음 Activity로 넘겨주게 된다.

1. Tag Activity

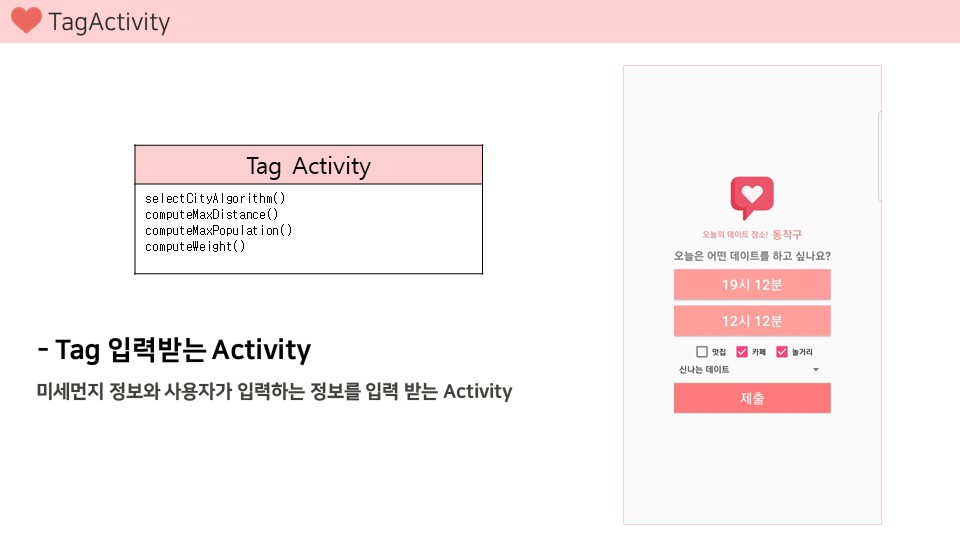


Figure 36

Figure 36은 사용자의 Tag정보를 입력 받는 곳으로, 데이트 시작과 종료 시간 그리고 Category와 Tag정보를 입력 받는다. 입력 받은 정보는 해당 행정구역에 대한 미세먼지 정보와 같이 다음 Activity로 넘겨진다.

1. Searching Activity

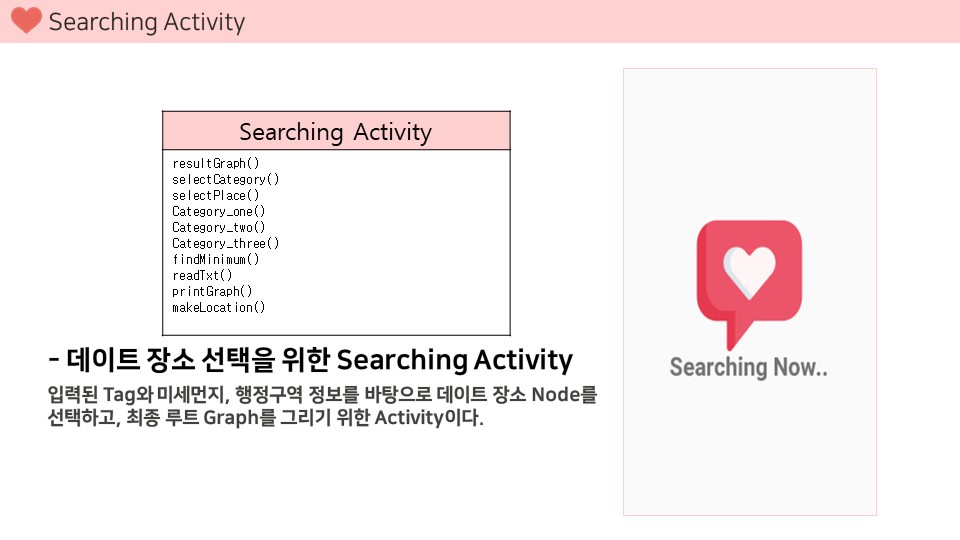


Figure 37

Figure 37은 데이트 장소 선택을 하기 위한 Algorithm이 진행되는 동안 보여주는 화면을 나타낸다. 이 Activity에서는 Figure 36에서 받은 정보를 바탕으로 최적의 데이트 장소를 선정하게 된다. 선정된 데이트 장소는 Result Node List에 들어가 최종 Activity로 넘겨진다.

1. Result & Popup Activity

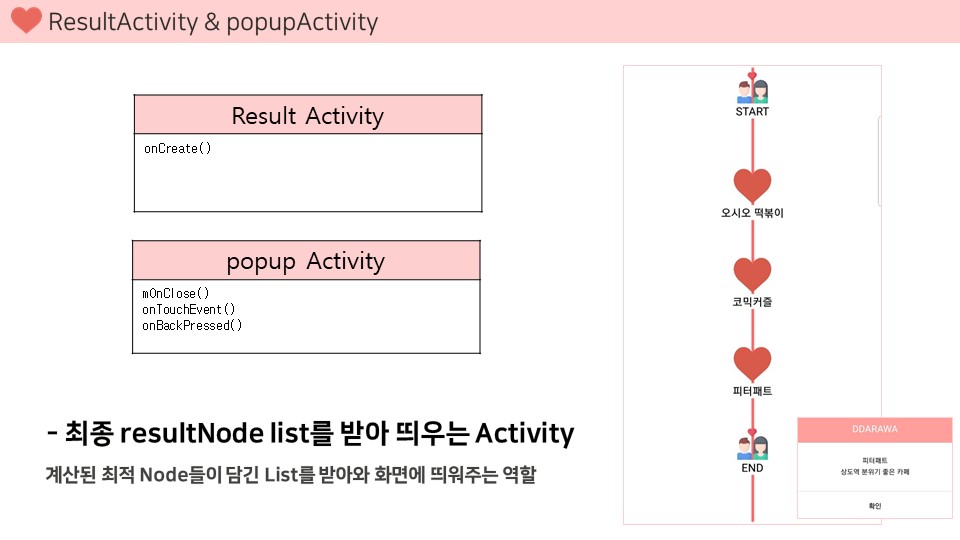


Figure 38

Figure 38은 최종 데이트 장소를 뿌려주는 Activity이다. Result Node List에 담긴 graph에서 필요한 정보를 가지고 와 Result Activity에 뿌려주고, 각 데이트 장소에 대한 자세한 정보는 PopupActivity를 통해 확인할 수 있다

## Algorithm

본 프로젝트를 진행하면서 크게 두가지 Graph가 사용이 되었다. 하나는 행정구역을 선택하는 상황, 그리고 다른 하나는 데이트 장소를 선택하는 상황이다.

1. City Select

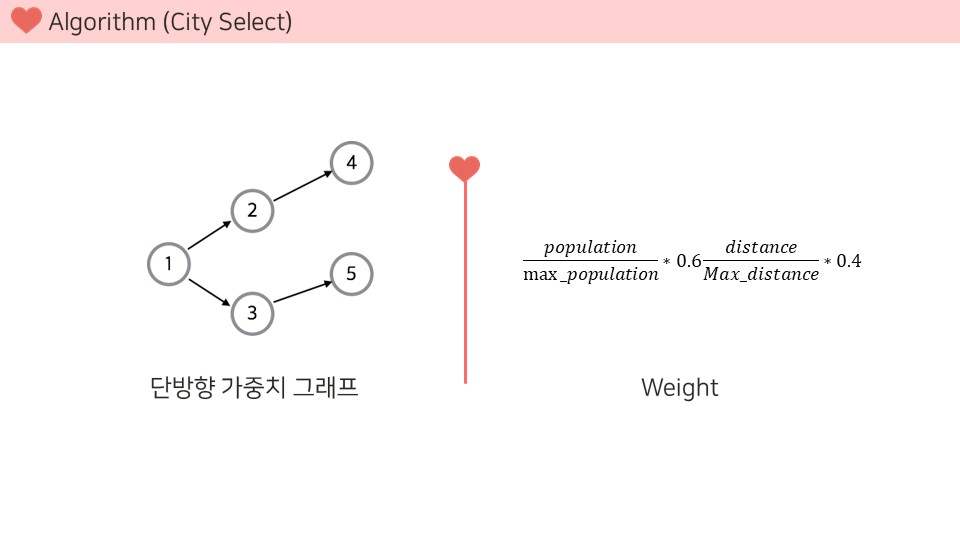


Figure 39

먼저 Figure 39에서는 city 선택 방법을 간략하게 나타내고 있다. City를 선택하는 Graph는 단방향 가중치 그래프를 그렸으며, 다익스트라 알고리즘을 이용하여 최단거리를 찾았다. 여기서 Weight을 선택하는 상황에서 Population과 Distance 두 가지 요소를 사용하였는데, 두 가지의 단위수가 다른 점을 고려하여 각 정보의 최대값으로 나눈 값을 Weight에 사용하였고, 그 식은 Figure 39의 오른쪽에 나타나 있다. 또한 population에 0.6, distance에 0.4를 곱한 이유는 Weight을 테스트 하며 교통이 잘 되어 있는 서울에서는 거리가중치보다 유동인구 가중치가 더 의미가 높을 것이라고 판단되어 위와 같이 수식을 작성하였다.

1. Place Select

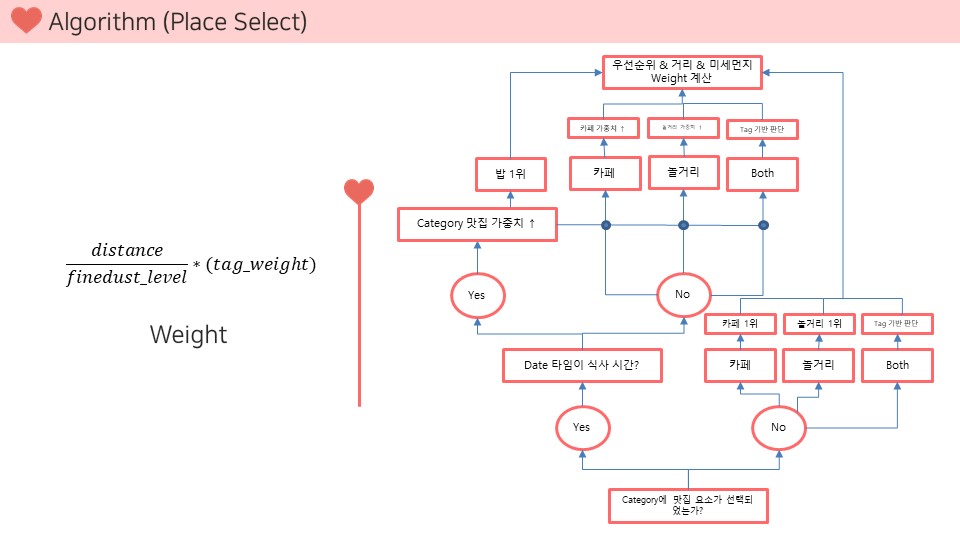


Figure 40

Figure 40은 최적의 데이트 장소를 찾는 과정에서 필요한 정보들을 표현한 것이다. 우선 City Select에서 선택한 것과 같은 단방향 가중치 그래프를 사용하였으며, Weight은 미세먼지 정보와 tag를 이용하여 왼쪽 부분과 같이 식을 작성하였다. 그리고 오른쪽은 카테고리의 우선순위를 선택하는 방법을 간략하게 Flow chart로 그려 놓은 것이다. 위 정보들을 바탕으로 다익스트라 알고리즘을 이용하여 가장 최적의 장소를 찾을 수 있게 된다.

## Problem and Solving

1. City Select시 Weight 결정 문제

E장 a절에서 언급한 것과 같이 유동인구와 거리는 단위수가 다른 문제가 있었다. 그래서 이 단위수를 맞추는 방법을 고민하다 E장 a절과 같은 수식으로 문제를 해결하였다. 또한 내 현재 위치에 대한 가중치가 너무 커서 결과에 다양성이 부족하였다. 그래서 교통문제를 고려하여 거리와 유동인구에 대한 weight을 다르게 설정하였다.

1. 미세먼지 정보

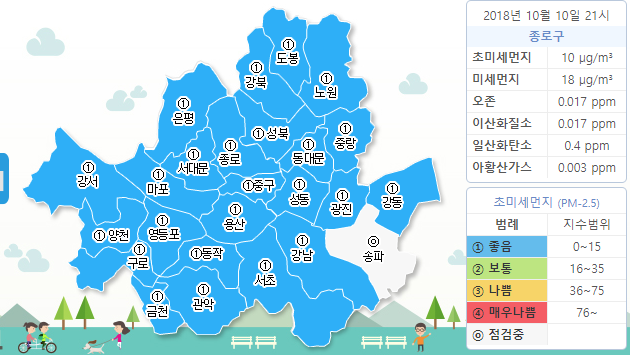


Figure 41

서울시 모든 행정구역에 대한 미세먼지 정보를 확인해보니 Figure 41의 송파구와 같이 점검중이라는 이유로 정보를 받아오지 못하는 경우가 발생하였다. 그래서 서울시 열린데이터 광장에서 서울시에 대한 전체적 미세먼지 정보를 받아와 대체하는 방법으로 문제를 해결하였다.