ANDROID TERMPROJECT

누나의 일기예보 어플리케이션 개발

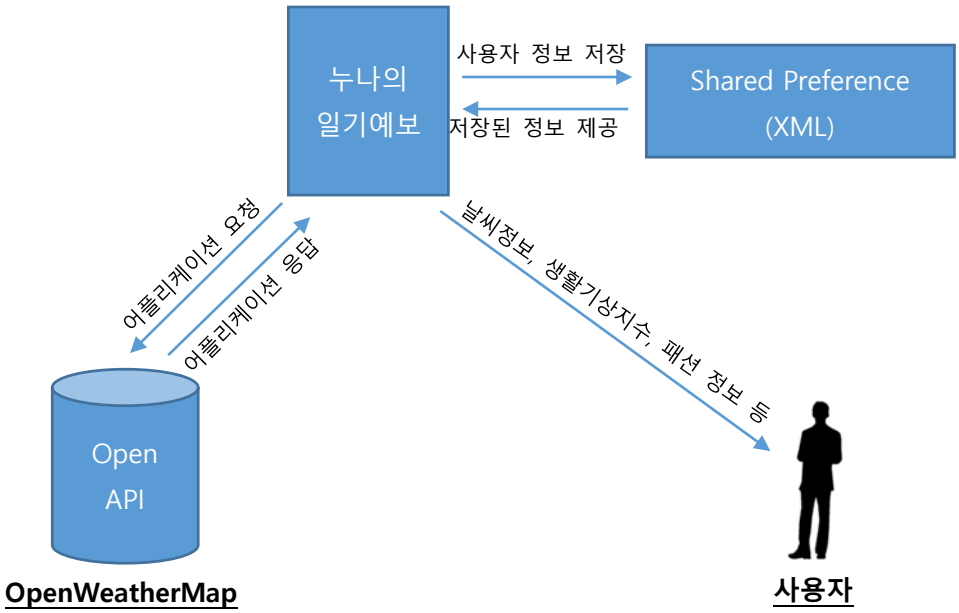
이름	학번
이상윤	201017093
박세호	201017041

담당교수: 고명철

목차

1. 어플리케이션 구성도

1-1. 시스템 구성도



- 누나의 일기예보 어플리케이션이 OpenAPI에 날씨 정보를 요청하면, OpenWeatherMap의 API는 RESTful한 응답으로 어플리케이션에게 날씨정보를 json(Javascript Object Notation) 형태로 응답해줌.
- 어플리케이션은 사용자 정보를 어플리케이션의 내부에 xml 형식으로 저장함. (Shared Preference를 이용함)
- 사용자는, 어플리케이션을 통해서 날씨정보(현재날씨, 7~14일간의 날씨정보), 생활기상지수(부패지수, 불쾌지수), 패션 정보를 제공받음.

1-2. 개발환경 및 테스트환경

개발언어	Java
최소 안드로이드 SDK 버전	Android 4.0 (Ice-cream Sandwich)
타겟 안드로이드 SDK 버전	Android 4.3 (Jelly Bean)
테스트 환경	Samsung Galaxy S3 (4.3 version)

1-3. 어플리케이션 기능 명세

분류	주 기능	세부 설명
스플래시	어플리케이션 로딩화면	
현재 날씨	현재 날씨	· 현재 온도 · 체감온도 · 최고/최저 온도 · 풍속

		· 기압 · 패션 코디
주간 날씨	주간 날씨	설정에 따라서 7일에서 14일 간의 날씨
설정	어플리케이션 설정	· 사용자 정보 · 날씨 정보 · 알림 정보 · 카피라이트

1-4. 어플리케이션 세부 기능

1. 스플래시화면

액티비티 명	SplashActivity
상세 기능	- 어플리케이션 로딩화면 - 어플리케이션 로고 - 어플리케이션 카피라이트
상세 설명	3~5초간, 어플리케이션 로딩 이미지를 표시함.
주요 사용 기능	- Handler - Intent

2. 현재 날씨

액티비티 명	MainActivity
상세 기능	- 현재 날씨정보 - 날씨에 맞는 패션 코디
상세 설명	현재 날씨 정보(현재 온도, 체감 온도,. 기온, 습도, 풍향 등) 현재 날씨에 어울릴 만한 패션 코디
주요 사용 기능	- Location API - AsyncTask - Preference - Intent - Action Button - Notification

3. 주간 날씨

액티비티 명	WeatherActivity
상세 기능	- 7~14일간의 날씨정보
상세 설명	7일(어플리케이션 설정 시 최대 14일까지 가능) 간의 날씨 정보를 리스트 형식으로 출력.

주요 사용 기능	- ListActivity - AsyncTask - Action Button

4. 설정

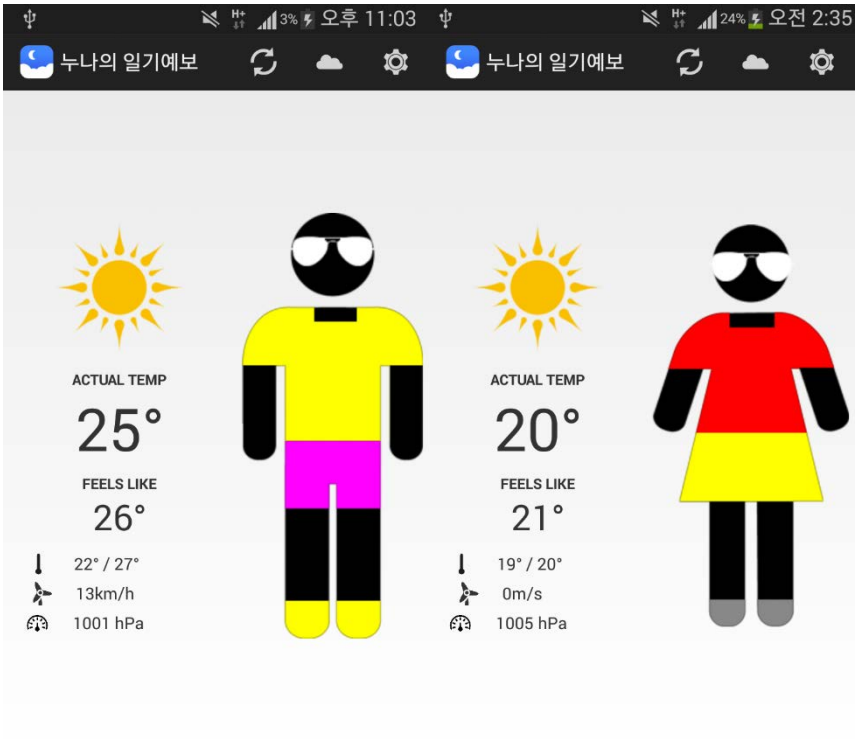
액티비티 명	SettingActivity
상세 기능	- 어플리케이션 설정
상세 설명	개인화 정보(사용자이름, 성별), 날씨정보(날씨 단위 설정, 출력 개수 설정), 알림정보(각종 노티피케이션 알림 출력여부)를 설정할 수 있으며, 카피라이트도 보여줌.
사용 기능	- PreferenceFragment

2. 어플리케이션 동작 화면

2-1. 스플래시 화면

[그림1 스플래시 화면]

2-2. 메인화면 (MainActivity)

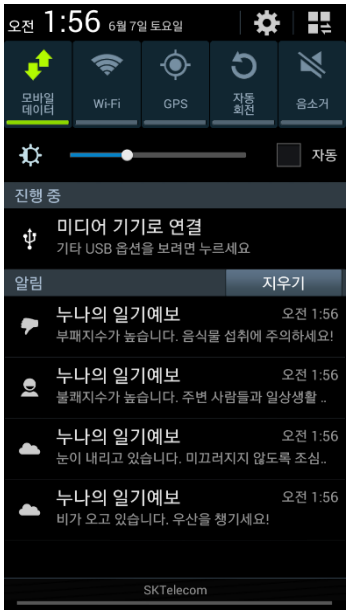


[그림2 메인화면]

i) 사용자 성별에 따라서, 아바타가 바뀐다. 기본적으로 설정 값이 없을 때(초기 어플리케이션 실행 시)는 남자 아바타가 출력된다.

ii) 현재 날씨 정보들이 표시된다. (날씨아이콘, 현재 날씨, 체감온도, 최저/최고 기온, 풍속, 기압)

iii) 패션 코디가 표시된다. (선글라스, 상의, 하의, 양말, 우산 등)



[그림3 알림화면]

i) 비, 눈이 올 때 어플리케이션이 노티피케이션 바(알림바)에 알림을 띄워준다.

ii) 현재 불쾌지수가 높을 경우(기상청 제공 기준 5 이상 시 50% 이상이 불쾌감을 느낌) 알림을 띄워준다.

iii) 현재 부패지수가 높을 경우(기상철 제공 기준 75 이상) 알림을 띄워준다.

✸ *알림은 스크린 샷을 찍기 위하여, 임의로 출력한 것임.*

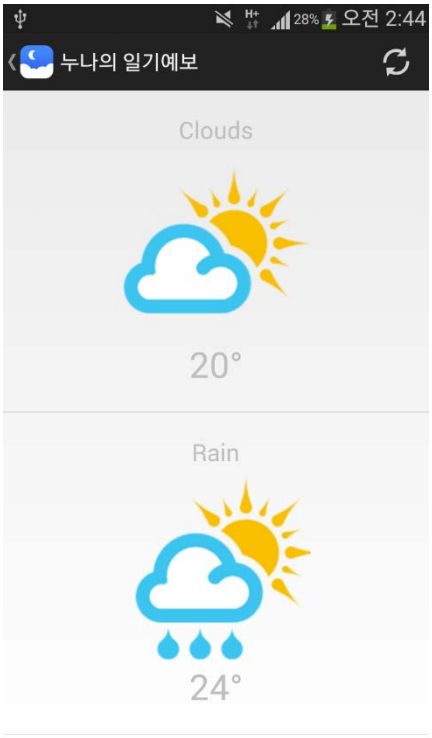
✸ *알림 테스트 여부를 거쳤으며, 적절한 상황에 적절하게 알림을 띄어줌.*

[그림4 메인 로딩 중 화면]

i) 데이터가 로딩 중일 때, Dialog를 통해 로딩 중임을 표시한다.

ii) 회원이 설정한 정보 중, 이름이 있을 경우 인사 Toast 메시지를 띄워준다.

2-3. 주간 날씨



[그림5 주간날씨 목록]

i) 7일(최대 14일) 간의 날씨를 현재 날씨, 아이콘, 온도와 함께 표시해준다.

ii) 설정에 따라서, 7일이 아닌, 14일까지 (7, 8, 9, … , 14일 까지 설정 가능) 날씨 목록이 출력된다.

iii) 새로고침을 통해서 날씨 정보를 갱신할 수 있다.

2-4. 설정화면

[그림6 설정화면]

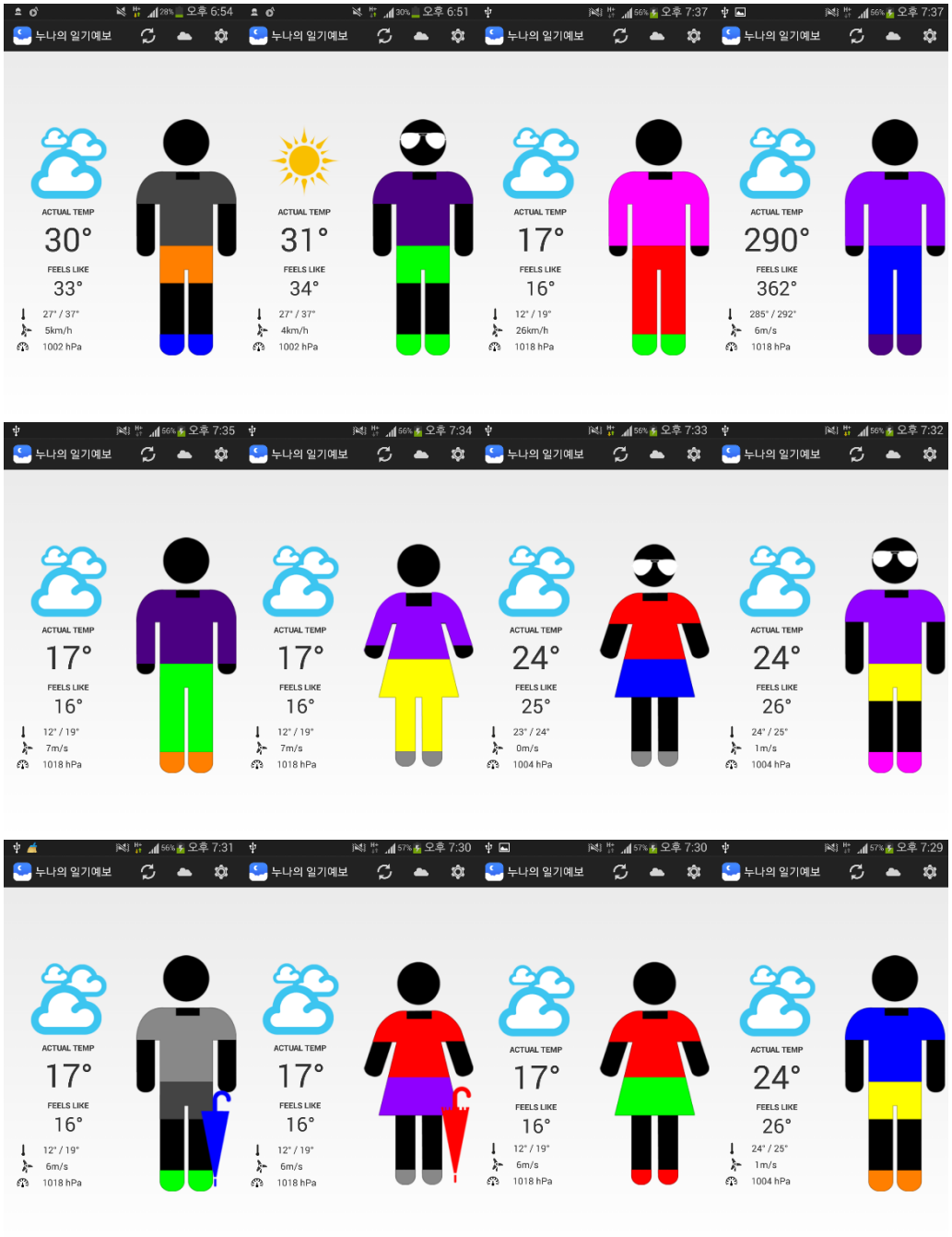
i) 사용자 프로필 정보를 설정할 수 있다. (이름과 성별)

ii) 날씨 정보를 설정할 수 있다. (날씨 단위, 풍속 단위, 날씨 개수)

iii) 알림 정보를 설정할 수 있다. (비, 눈, 불쾌지수, 부패지수 알림)

iv) 카피라이트를 볼 수 있다.

2-5 패션 코디 추천 예시



[그림7 패션 코디]

i) 날씨에 따라서, 선글라스를 추천할 수 있다.

ii) 날씨에 따라서 긴 옷, 짧을 옷을 추천할 수 있다.

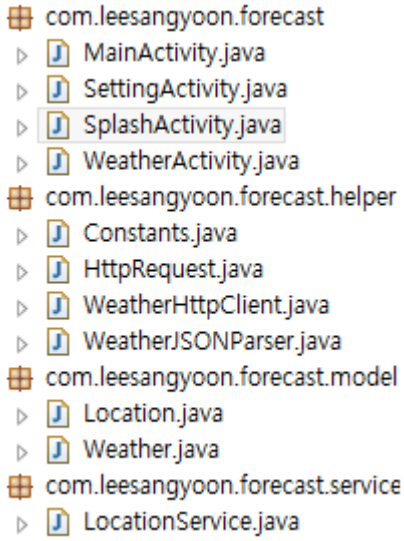
iii) 성별에 따라서 남자, 여자 옷을 추천할 수 있다.

iv) 옷 색깔을 추천할 수 있다. (간단하게 랜덤으로 추천함.)

v) 비가 오는지 여부에 따라서 우산을 추천할 수 있다.

3. 주요코드

3-1. 소스코드 구성



어플리케이션은 위와 같은 구조로 구성되어 있다.
Activity는 어플리케이션의 기본 패키지에서 관리하고 있다.

3-2. 모델(model)

```
public class Location implements Serializable
{
    private float latitude;
    private float longitude;

    private long sunset;
    private long sunrise;

    private String country;
    private String city;
```

```
public class Weather
{
    public Location location;

    public Condition condition = new Condition();
    public Temperature temperature = new Temperature();
    public Wind wind = new Wind();
    public Rain rain = new Rain();
    public Snow snow = new Snow()    ;
    public Clouds clouds = new Clouds();
```

OpenWeatherMap API에서 제공해주는 정보를 그대로, Class화 시켜 모델로 만들어 보았다. 위치정보에는 위도와 경도(나머지는 제공해주지만 어플리케이션에서는 활용하지 않았음), 날씨정보(현재 날씨, 온도, 바람, 눈, 구름, 비 지수들) 등을 제공해주어 이를 그대로 클래스화 시켰다.

3-3. 서비스

위치 정보를 얻어오기 위해서 두 가지 방법을 활용해 보았다.
처음은, 구글플레이서비스에서 제공해주는 위치정보인데, 이는 서비스로 등록되어 어플리케이션에서 백그라운드 서비스로 동작하게 된다.

```
@Override
public void onLocationChanged( Location location )
{
    latitude = location.getLatitude();
    longitude = location.getLongitude();

    Intent intent = new Intent();

    intent.setAction( ACTION );
    intent.putExtra( "latitude" , latitude );
    intent.putExtra( "longitude" , longitude );

    sendBroadcast( intent );
}
```

위치가 변하게 되면, 이를 latitude, longitude 변수에 저장하고 인텐트에 정보를 실어, 브로드캐스팅 한다. (Main, Weather Activity에서 이 정보를 받아 위치 정보를 받아온다.)

3-4. 헬퍼클래스

```
public static Weather getWeather( String data ) throws JSONException
{
    Weather weather = new Weather();

    JSONObject jObject = new JSONObject( data );

    Location location = new Location();

    JSONObject coord = getObject( "coord" , jObject );
    location.setLatitude( getFloat( "lat" , coord ) );
    location.setLongitude( getFloat( "lon" , coord ) );

    JSONObject sys = getObject( "sys" , jObject );
    location.setCountry( getString( "country" , sys ) );
    location.setSunrise( getInt( "sunrise" , sys ) );
    location.setSunset( getInt( "sunset" , sys ) );
    location.setCity( getString( "name" , jObject ) );

    weather.location = location;

    JSONArray jArray = jObject.getJSONArray( "weather" );

    JSONObject JSONWeather = jArray.getJSONObject( 0 );
    weather.condition.setWeatherId( getInt( "id" , JSONWeather ) );
    weather.condition.setDesc( getString( "description" , JSONWeather ) );
    weather.condition.setCondition( getString( "main" , JSONWeather ) );
    weather.condition.setIcon( getString( "icon" , JSONWeather ) );

    JSONObject main = getObject( "main", jObject );
    weather.condition.setHumidity( getInt( "humidity" , main ) );
    weather.condition.setPressure( getInt( "pressure" , main ) );
    weather.temperature.setMaxTemp( getFloat( "temp_max" , main ) );
    weather.temperature.setMinTemp( getFloat( "temp_min" , main  ));
    weather.temperature.setTemp( getFloat( "temp" , main ) );

    JSONObject wind = getObject( "wind" , jObject );
    weather.wind.setSpeed( getFloat( "speed" , wind ) );
    weather.wind.setDeg( getFloat( "deg" , wind ) );

    JSONObject clouds = getObject( "clouds" , jObject );
    weather.clouds.setPerc( getInt( "all" , clouds ) );

    return weather;
}
```

JSON을 해석하는 JsonParser 클래스에서는 위와 같이, 받아온 JSON 데이터를 해석해서, 원하는 정보를 추출한 후, Model에 맞춰 데이터를 집어 넣는 역할을 한다.

```
public String getWeatherData( double latitude , double longitude , String unit )
{
    String url = WEATHER_DATA_PROVIDER + "?lat=" + latitude + "&lon=" + longitude + "&units=" + unit;

    try
    {
        HttpRequest request = HttpRequest.get( url );

        if( request.ok() )
            return request.body().toString();
        else
            return null;
    }
    catch( Exception e )
    {
        e.printStackTrace();
    }

    return null;
}
```

Http를 요청하고 응답받는 클래스에서는 위와 같이 Http 요청을 받아온다.
이를 수월하게 사용하기 위해서, HttpRequest(Open source)를 활용했다.
위도, 경도, 단위를 바탕으로 날씨 제공자에게서 RESTful한 응답을 받고 (Json 형식) 이를 가져와 JSON 파서기에서 파서하게 되는 구조이다.

3-5. 설정정보 가져오기

```
prefs = PreferenceManager.getDefaultSharedPreferences( getApplicationContext() );
String unit = prefs.getString( "prefWeatherUnit" , "celsius" );
unit = ( unit.equalsIgnoreCase( "celsius" ) ) ? "metric" : "";
```

SharedPrefence에 저장된 내용을 가져온느 부분이다.
값을 가져와 이를 바탕으로 동적인 정보를 제공한다. (날씨 단위나 풍속, 아바타 변경 등에 활용되었다.)

3-6. 비동기 작업 요청

```
task = new WeatherTask();
task.execute( String.valueOf( latitude ) , String.valueOf( longitude ) , unit );
```

```
private class WeatherTask extends AsyncTask< String , Void , Weather >
{
    private ProgressDialog dlg = null;

    @Override
    protected void onPreExecute()
    {
        super.onPreExecute();

        dlg = ProgressDialog.show( MainActivity.this , "데이터 로딩중" , "날씨정보를 불러오고 있습니다." , true );
    }
```

```
@Override
protected Weather doInBackground( String... params )
{
    if( !isCancelled() )
    {
        Weather weather = new Weather();

        String data = ( new WeatherHttpClient() ).getWeatherData( Float.parseFloat( params[0] ) , Float.parseFloat( params[1] ) , params[2] );

        try
        {
            weather = WeatherJSONParser.getWeather( data );
        }
        catch( JSONException e )
        {
            e.printStackTrace();
        }

        return weather;
    }

    return null;
}
```

AsyncTask를 활용하여, 비동기 데이터 요청을 처리하고 있는 부분이다.
다이얼로그를 이용해서, 요청 중 여부를 표시해준다.
HTTP 데이터를 요청(RESTful)해 받아온 JSON 데이터를 파싱 한 후, 이를 활용한다.

3-7 인텐트필터 및 서비스

```
@Override
protected void onStart()
{
    locationReceiver = new LocationReceiver();

    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction( LocationService.ACTION );

    registerReceiver( locationReceiver , intentFilter );

    Intent intent = new Intent( this , LocationService.class );
    startService( intent );

    super.onStart();
}
private class LocationReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive( Context context , Intent intent )
    {
        latitude = intent.getDoubleExtra( "latitude" , 0.0 );
        longitude = intent.getDoubleExtra( "longitude" , 0.0 );
    }
}
```

onStart를 오버라이드 하여, 액티비티에서 서비스를 시작하는 부분이다.
또한 서비스에서 브로드캐스팅 한 정보를 LocationReceiver에서 받아오게 된다.

3-8. 컬러필터

```
Random random = new Random();
shirts.setColorFilter( recommanedColor[ random.nextInt( recommanedColor.length ) ] );
pants.setColorFilter( recommanedColor[ random.nextInt( recommanedColor.length ) ] );
socks.setColorFilter( recommanedColor[ random.nextInt( recommanedColor.length ) ] );
acc3.setColorFilter( recommanedColor[ random.nextInt( recommanedColor.length ) ] );
```

안드로이드에서 제공하는 Drawable의 컬러 값을 컬러필터를 통해서 수정하는 부분이다.
옷 색깔을 다양하게 추천해 줄 때 활용했다.

3-9. 노티피케이션 알림

```
// 부패지수가 높음
double putrefaction = ( ( humidity - 65 ) / 14 ) * Math.pow( 1.054 , c );

if( putrefaction >= 75 )
{
    Notification putAlert = new NotificationCompat.Builder( getApplicationContext() )
      .setContentTitle( "누나의 일기예보" )
      .setContentText( "부패지수가 높습니다. 음식물 섭취에 주의하세요!" )
      .setSmallIcon( R.drawable.ic_action_bad )
      .setAutoCancel( false )
      .setContentIntent( pendingIntent )
      .build();

    if( alert4 == true )
        notificationManager.notify( 4 , putAlert );
}
```

특정 상황에 대한 알림은 Notification을 통해서 알림해주는 부분이다.
이 부분에서 NotificationCompat를 이용해서 Notification을 생성해주고, Notification Manager를 통해서 알림해주는 부분이 코딩되어 있다.

4. 마무리

4-1. 기획서와 다른 점

Open API는 각 제공자 마다 다른 정보를 제공해준다. 이번에 사용한 Open Weather Map API는 외국에서 제공하는 무료 날씨정보를 제공해준다. 기본적으로 어플리케이션에서 위도와 경도를 통해서 날씨를 받아오는 방식의 설계는 동일 했으나, 오늘 밤, 내일 낮의 날씨와 같은 세세한 날씨정보는 제공하지 않기 때문에 이 부분을 부득이하게 빼버렸다.

또한, 오늘의 날씨를 시간대 별로 출력해주는 부분에서도 오늘의 날씨를 시간 별로 제공해주지 않아 기간 별 날씨(7일간 날씨)를 제공하도록 변경하였다.

그 외에, 조금 어색하게 설정된 부분들이 약간씩 수정되었다.
기획서에는 없지만, 추가적으로 노티피케이션 알림을 활용한 부분이 있다.
오늘 날씨에 따라서 비가오거나 눈이 오면 사용자에게 비가 온다 우산을 챙겨라라던지, 눈이오니 미끄러지지 않도록 주의하라는 알림을 띄워주게 된다. 또 생활기상지수인 부패지수와 불쾌지수를 활용하여 일정 수치 이상일 경우, 알림을 띄워주도록 해 보았다.

4-2. 개발을 하면서 느낀점

Open API를 RESTful하게 제공해주기 때문에, JSON 형식으로 데이터를 받아오는 부분과 그 데이터를 파싱하는 부분이 어플리케이션의 핵심 적으로 강조하고 싶은 부분이다. (프로그래밍 적인 스킬) 그 외에는 수업시간에 다루었던, 액션바(액션메뉴)나 인텐트, 액티비티, 생명주기, 리스트뷰를 활용했다. 이미지는 안드로이드에서 제공해주는 API에 따라서, 색상을 바꿔주는 부분(색상 필터)를 활용해보았다.

아쉬운 점은, 무료 API를 사용하다 보니 제공해주는 정보의 양이 적어, 기획서와 다르게 개발했다는 점이다. 사실, 이 점은 API 제공자를 바꾸면 되지만 무료인 데다가 깔끔하게 지원해주는 곳이 Open Weather Map 이기 때문에 이 곳을 활용해보았다.

어플리케이션에서 강조하고 싶은 부분은 Async Task를 통한 비동기 데이터 처리, Http 데이터를 가져오는 부분이 될 것 같다.
또, 설정정보에 따라서 동적으로 변하는 아바타나, 패션 코디 색상(컬러필터) 등도 강조하고 싶은 부분이다.
설정을 SQLite로 처리하려고 했지만, 이런 가벼운 정보를 굳이 Database까지 활용해 처리하기엔 비용적인 측면으로 좋지 않을 것 같아서 이를 Preference를 통해서 XML로 (map 형식으로 저장됨) 처리해 보았다.