

모바일프로그래밍

ASSIGNMENT

기말 보고서



과목명	모바일프로그래밍
담당교수	이상홍 교수님
전공	컴퓨터공학전공
이름	김이삭
학번	2017E7441

목차

1. 기말 과제 소개
2. 디테일 설명 및 알고리즘
3. 소스코드(주요코드)
4. 결과
5. 소감
6. 참고자료

1, 기말 과제 소개

GPS기반 다이어트 앱

앱 명 : 지피어트 (GPS + 다이어트를 합하여 만들어보았습니다.)

이번 학기말 과제의 주제는 GPS or 스마트폰 Sensor를 활용한 어플리케이션을 개발하는 것입니다. 저는 GPS를 활용한 응용 Application을 계획했습니다. GPS와 관련된 나침반 Sensor도 함께 활용하게 되었습니다.

지피어트는 GPS를 기반으로 한 걷기 운동을 도와주는 Application입니다. 목적지 검색하고, 목적지를 설정하고, 해당 목적지 정보를 받아 경로를 선택하고 걷기를 시작합니다.

목적지를 선택하여 걷기를 시작하면 걸어가는 모습을 GPS맵 상에서 트래킹할 수 있으며, 거리 측정에 대한 부분은 background Service 로도 위치와 걷는 거리를 확인하여 휴대폰으로 알림을 줄 수 있도록 하였습니다.

목적지 도달까지 3분의 1 지점, 3분의 2지점, 목적지 부근 지점에 도착할 경우에 알림이 울리도록 했습니다.

걷기를 마치면 background Service에서 데이터베이스에 자동으로 데이터(목적지, 거리, 퍼센티지 등)를 저장하고, 리스트로 관리할 수 있습니다.

운동기록을 확인할 수 있는 리스트 뷰를 구현하여 기록을 삭제할 수 있도록 구현했습니다.

2. 디테일 설명 및 알고리즘

1. IntroActivity

다른 상용 Application처럼 Intro 화면을 먼저 보여주고, MainActivity로 자연스럽게 넘어가는 기능을 구현하고자 했습니다.

프로젝트에 anim이라는 폴더에 fade in, fade out 애니메이션에 대한 정보 파일(XML)을 넣고, 인트로에서 메인 액티비티로 이동하는 과정을 시간을 지정하여 자동으로 넘어가도록 했고, 이때 이전 액티비티와 다음 액티비티로 넘어가는 과정에 페이드 인 아웃을 교차하도록 했습니다.

2. MainActivity

메인 액티비티는 심플하고 깔끔하게 구성했습니다. 운동기록 버튼과, 목적지입력창, 검색버튼이 있고, 가운데에 GIF의 동적인 이미지를 넣어보았습니다. GIF를 로딩하기 위해 Glide 라이브러리를 사용합니다.

3. 목적지를 입력하고 검색 버튼을 누르면 나오는 SearchResultListActivity

먼저 이 앱에서는 위젯에 대한 초기화를 진행하고, Map을 초기화합니다. Map을 표현하는 API는 다음 카카오의 지도 API를 사용했습니다. 그리고 목적지 데이터를 검색하는 API는 SK open API, Tmap API를 활용했습니다. 검색어를 통해 Tmap API, SK open API에서 REST 프로토콜을 활용하여 목적지에 대한 정보를 API 제공처 측으로 보내면, 요청한 데이터에 의해 검색된 데이터를 반환하여 줍니다. JSON타입으로 데이터를 받도록 설정을 하고 요청합니다. JSON을 gson이라는 라이브러리를 이용하여 데이터들을 파싱하여 각 아이템을 ArrayList에 담아서 UI를 표현합니다.

검색 위치 요청에 대한 알고리즘 설명입니다.

3-1) 먼저 마지막 위치를 요청하는 함수를 호출하고, 성공 시 REST통신 작업을 수행합니다.

3-2) REST로 TMAP_API에 데이터를 요청하기 위해 UI 멈춤 등 상호작용에 영향을 주지 않도록 쓰레드를 생성하여 요청합니다.

3-3) REST통신으로 데이터를 얻기 위해 요청할 URI와 필요한 파라미터 값을 Request 객체에 세팅합니다.

3-4) Request에 담은 값(uri+요구된 파라미터)을 http통신을 실행하여 결과 값을 받습니다.

3-5) 받은 결과 값은 JSON타입이기 때문에 JSONObject 객체를 통해 먼저 필요한 아이템을 구분할 수 있도록 분해 작업을 합니다.

3-6) 각 분해된 JSONArray를 JAVA가 쉽게 읽을 수 있도록 파싱합니다. GSON라이브러리 사용

3-7) 파싱된 값에 맞는 POIModel객체에 담고(Java에서 VO, DTO, beans로 표현되는 객체 모델) POIModel 인스턴스를 List에 담습니다.

3-8) 뷰를 표현할 어댑터에 데이터 변화를 알리고, 검색된 위치를 지도에 마킹합니다. UI에서 돌아가는 작업이므로 Thread를 이용합니다.

4. 개별 아이템 클릭 시, 경로 및 목적지까지의 거리를 표현하는 SummaryRouteActivity
이전 액티비티에서 목적지 정보와, 마지막 현재 위치를 Intent에 담아서 받습니다. 위에서 사용했던 방법과 같이 Tmap 보행자 경로 표시 API를 사용하여 데이터를 받아옵니다. 경로를 표시하는 Polyline 정보를 맵에 표시합니다. 앞장에서 비슷한 알고리즘으로 설명됩니다. 조금 다른 점은 추천경로, 최단경로 등에 요청하는 값에 따라 다른 하나의 루트를 받게 되어 http 파라미터에 다른 옵션으로 네 번 요청하도록 합니다.

주요 알고리즘

4-1) 3에서 REST http 프로토콜 통신을 이용해 받았던 것과 비슷하게 Url 요청을 통해 JSON 값을 받아옵니다.

4-2) 루트를 표현하기 위해 저장할 수 있는 별도의 객체를 생성해야 합니다.

- 루트를 표현하기 위한 모델 SummaryRouteItemModel 클래스와 루트의 표현을 도와줄 좌표를 담을 수 있는 클래스 LocationItemModel를 구현합니다.
- 이유 : 루트를 표현하는 폴리라인은 여러 개의 선으로 되어 있는데, 여러 개의 선은 여러 개의 점으로 구성되어 있습니다.
- 루트 하나는 여러 개의 점을 갖고 있기 때문에, 루트 하나는 점들의 집합
- 루트들은 여러 개의 루트를 갖고 있기 때문에, 루트들의 집합으로 곧 집합들의 집합으로 관리

4-3) JSON에서 받아온 값 루트(점들의 집합)를 구한 뒤, 점들을 JAVA 코드로 변환(LocationItemModel 객체에 넣고)하고, 점들을 하나의 집합으로 만들어줍니다. (리스트에 담는다.)

- 2중 for문으로 구현
- 외부 for문으로 JSON의 루트에 관한 값, 각 루트에 대하여
- 내부 for문으로 각 루트에 관한 값의 점들을 구하고 자바에서 인식할 수 있는 집합(리스트)에 저장

5. NavigationActivity

이전 액티비티에서 루트와 목적지에 대한 정보를 받아옵니다.

간단한 위젯을 초기화하고, 맵을 표현합니다. 다음 API를 활용해 트래킹모드로 실시간 GPS 위치를 화면에 표시합니다. Timer와 TimerTask를 활용하여 시간과 거리를 1초마다 구하여 화면에 나타냅니다. 이는 스레드에서 동작합니다. 루트를 그리는 방법은 이전 액티비티에서 같은 알고리즘을 사용합니다.

이 액티비티에서 백그라운드 서비스를 동작시킵니다. 루트정보와 목적지 정보를 인텐트에 담아 보냅니다.

6. MyService 와 ServiceThread

서비스에서 계속 데이터를 체크하고 값을 받아서 앱에 보내야 하는 요소로 작동합니다. 인텐트를 통해 받은 목적지와 루트정보를 토대로 이동거리를 계산합니다. 주기적으로 현 위치와 목적지까지의 남은 거리를 계산하기 위해 쓰레드로 동작됩니다.

백그라운드 서비스에서 주기적인 동작을 위해 Handler 객체를 사용해야 하므로 MyService 내부에 별도의 객체를 생성하여 Handler를 상속받아 내부의 실제 동작을 구현합니다. 백그라운드 서비스에서 주기적으로 알림을 보내는 동작입니다.

목적지와 남은 거리를 계산하여 약 [1/3], [2/3], [3/3] 지점에 도착하였을 때 알림이 오도록 설정되어있습니다.

비정상종료 혹은 이전 액티비티에서 벗어나게 되면, 서비스를 자동으로 정지됩니다. 정지되기 직전에 목적지, 이동거리 등에 대한 데이터를 데이터베이스에 저장합니다.

7. ReportListActivity

실제로 목적지를 선택하고 이동하였던 데이터들의 기록을 볼 수 있도록 관리하는 화면입니다. 데이터베이스에서 불러와지고, 목록에서 데이터를 삭제할 수 있습니다. 삭제한 후 화면의 자연스러운 처리를 하도록 구현했습니다.

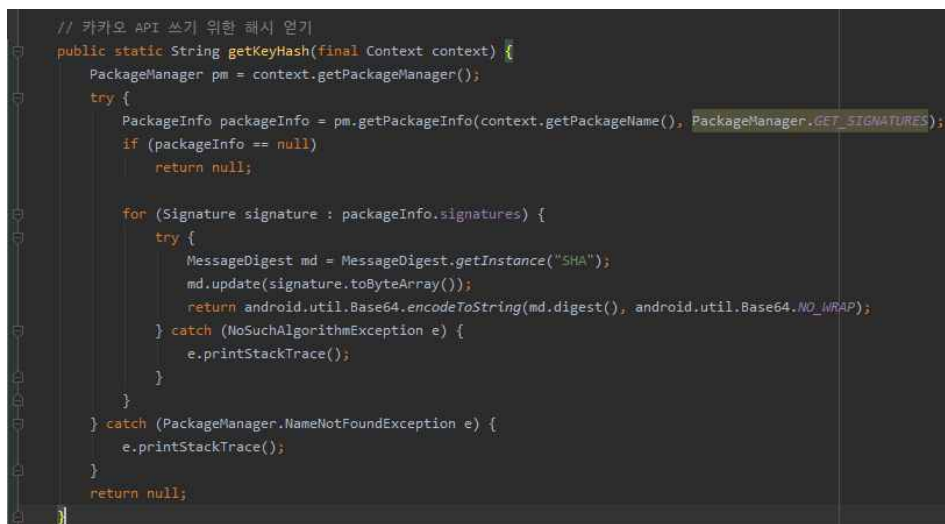
그 외에도 다양한 클래스가 존재하나 개발 도중에 사용하지 않게 된 클래스나, DBHelper 와 같은 Util 과 같은 클래스는 언급하지 않았습니다. 앱 로직에 대한 설명을 주로 하고자 했습니다.

3. 소스코드 (주요코드)

다음 카카오 API를 사용하기 위해 앱의 Hash Key를 구하는 함수

```
// 카카오 API 쓰기 위한 해시 얻기
public static String getKeyHash(final Context context) {
    PackageManager pm = context.getPackageManager();
    try {
        PackageInfo packageInfo = pm.getPackageInfo(context.getPackageName(),
PackageManager.GET_SIGNATURES);
        if (packageInfo == null)
            return null;

        for (Signature signature : packageInfo.signatures) {
            try {
                MessageDigest md = MessageDigest.getInstance("SHA");
                md.update(signature.toByteArray());
                return android.util.Base64.encodeToString(md.digest(),
android.util.Base64.NO_WRAP);
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    } catch (PackageManager.NameNotFoundException e) {
        e.printStackTrace();
    }
    return null;
}
```



```
// 카카오 API 쓰기 위한 해시 얻기
public static String getKeyHash(final Context context) {
    PackageManager pm = context.getPackageManager();
    try {
        PackageInfo packageInfo = pm.getPackageInfo(context.getPackageName(), PackageManager.GET_SIGNATURES);
        if (packageInfo == null)
            return null;

        for (Signature signature : packageInfo.signatures) {
            try {
                MessageDigest md = MessageDigest.getInstance("SHA");
                md.update(signature.toByteArray());
                return android.util.Base64.encodeToString(md.digest(), android.util.Base64.NO_WRAP);
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    } catch (PackageManager.NameNotFoundException e) {
        e.printStackTrace();
    }
    return null;
}
```

REST http 통신을 위한 Request url세팅

```
fusedLocationClient.getLastLocation()
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getApplicationContext(), "GPS를 얻는데 실패했습니다. Wn검색을 재시도
합니다.", Toast.LENGTH_SHORT).show();
        }
    })
    .addOnSuccessListener(new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(final Location location) {    // GeoJSON 타입으로 자료 요청하고 받
기| Protocol REST / Post Method
        // SK open API : openapi.sk.com/resource/apidbc/indexView -> 명칭(POI) 통합검색
        if (location != null) {
            Thread t1 = new Thread(new Runnable() {
                //Background Thread Task
                public void run() {
                    Request request = new Request.Builder()
                        .url(DEFINES.TMAP_SCH_POI_API_URL + search_keyword +
// 데이터 요청 TAMP
                        "&centerLon=" + location.getLongitude() +
                        "&centerLat=" + location.getLatitude() +
                        "&searchtypCd=R&" +
                        "radius=30")    // 반경 확인하기
                        .header("Accept", "application/json")    // JSON 포맷의 데이터
                        .header("Content-Type", "application/json; charset=UTF-8")
                        .build();

                    try (Response response = mHttpClient.newCall(request).execute()) {    //
REST API

                    String result = response.body().string();
                    //json 파싱
                    try {
                        parsePOIDataFromJsonResult(result); // 파싱
                    } catch (final Error e) {
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                Toast.makeText(getApplicationContext(), "검색중에 에러가
발생했습니다.Wn" + e.toString(), Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                }
            });
        }
    });
}
```

REST http 통신을 위한 Request url세팅

```

    })
    .addOnSuccessListener((OnSuccessListener) (location) -> { // GeoJSON 타입으로 자료 요청하고 받기 Protocol REST / Post Method
        // SK open API : openapi.sk.com/resource/apidbc/indexView -> 명칭(POI) 통합검색
        if (location != null) {
            Thread t1 = new Thread((Runnable) () -> {
                Request request = new Request.Builder()
                    .url(DEFINES.TMAP_SCH_POI_API_URL + search_keyword + // 데이터 요청 TAMP
                        "&centerLon=" + location.getLongitude() +
                        "&centerLat=" + location.getLatitude() +
                        "&searchtypeCd=R2" +
                        "&radius=30") // 반경 확인하기
                    .header( NAME, "Accept", value: "application/json") // JSON 포맷의 데이터
                    .header( NAME, "Content-Type", value: "application/json; charset=UTF-8")
                    .build();

                try (Response response = mHttpClient.newCall(request).execute()) { // REST API

                    String result = response.body().string();
                    //json 파싱
                    try {
                        parsePOIDataFromJsonResult(result); // 파싱
                    } catch (final Error e) {
                        runOnUiThread(() -> {
                            Toast.makeText(getApplicationContext(), text: "검색중 에러가 발생했습니다.\n" + e.toString(), Toast.LENGTH_SHORT).show();
                        });
                    }
                }
            });
        }
    });

```

JSON 파싱, GSON 라이브러리 사용하여 간단하게 처리

```

private void parsePOIDataFromJsonResult(String res) throws JSONException {
    JSONObject jsonObject = new JSONObject(res);
    JSONObject jsonSchPOIInfo = (JSONObject)jsonObject.get("searchPoiInfo");
    JSONObject jsonPOIs = (JSONObject)jsonSchPOIInfo.get("pois");
    JSONArray jsonPOIArr = (JSONArray)jsonPOIs.get("poi"); // 실질적 실제 데이터 Array

```

Gson gson = new Gson(); // GSON : Json -> string -> Model 반대로 toJson을 사용하면 json으로 만드는 것도 가능하다 함.

```

mPOIModelList.clear();
for( int idx = 0 ; idx < jsonPOIArr.length() ; idx++){
    POIModel single_item = gson.fromJson(jsonPOIArr.get(idx).toString(), POIModel.class);
    single_item.kcal = Double.parseDouble(single_item.radius) * 57.7; // 칼로리
    mPOIModelList.add(single_item);
}
}

```

```

// JSON 파싱 반환된 데이터타입이 json 타입이기 때문에
private void parsePOIDataFromJsonResult(String res) throws JSONException {
    JSONObject jsonObject = new JSONObject(res);
    JSONObject jsonSchPOIInfo = (JSONObject)jsonObject.get("searchPoiInfo");
    JSONObject jsonPOIs = (JSONObject)jsonSchPOIInfo.get("pois");
    JSONArray jsonPOIArr = (JSONArray)jsonPOIs.get("poi"); // 실질적 실제 데이터 Array

    Gson gson = new Gson(); // GSON : Json -> string -> Model 반대로 toJson을 사용하면 json으로 만드는 것도 가능하다 함.
    mPOIModelList.clear();
    for( int idx = 0 ; idx < jsonPOIArr.length() ; idx++){
        POIModel single_item = gson.fromJson(jsonPOIArr.get(idx).toString(), POIModel.class);
        single_item.kcal = Double.parseDouble(single_item.radius) * 57.7; // 칼로리
        mPOIModelList.add(single_item);
    }
}

```


위 4-3)에서 설명한 2중 for문을 활용한 route 데이터 리스트 저장

```
else{ // json_coordinates는 Route
    for(int idx_coordinates = 0 ; idx_coordinates < json_coordinates.length() ; idx_coordinates++){
        JSONArray single_location = json_coordinates.getJSONArray(idx_coordinates);
        for(int idx_location = 0 ; idx_location < single_location.length() ; idx_location++){
            Log.i("kis",single_location.get(idx_location).toString());
            double lon = (Double) single_location.get(0); // x
            double lat = (Double) single_location.get(1); // y

            LocationItemModel item_location = new LocationItemModel(lon,lat); //하나의 지점
            items_location_list.add(item_location);
        }
    }

    item_route_item.addCoordinate(items_location_list);
}
```

```
//좌표가 여러개 있는 경우
else{ // json_coordinates는 Route
    for(int idx_coordinates = 0 ; idx_coordinates < json_coordinates.length() ; idx_coordinates++){
        JSONArray single_location = json_coordinates.getJSONArray(idx_coordinates);
        for(int idx_location = 0 ; idx_location < single_location.length() ; idx_location++){
            Log.i( tag: "kis",single_location.get(idx_location).toString());
            double lon = (Double) single_location.get(0); // x
            double lat = (Double) single_location.get(1); // y

            LocationItemModel item_location = new LocationItemModel(lon,lat); //하나의 지점
            items_location_list.add(item_location);
        }
    }

    item_route_item.addCoordinate(items_location_list);
}
```

MyService 클래스 : 목적지와 남은 거리 사이의 퍼센티지 구하기

```
public double updatePercentage(int dest, int remain) {
    double p = 0;
    if (dest >= remain) { // ex 목적지까지 거리가 123, 남은거리가 122이나 123이면 실행
        //Log.e(TAG, "updatePercentage: dest랑remain 제대로 들어오는지 dest : "+dest+", remain : "+remain);
        nDistanceDifference = (dest-remain);
        if (nDistanceDifference > 0) {
            p = nDistanceDifference/(double)dest * 100.0;
            p =(int) (p * 100);
            p = p / 100.0;
            Log.e(TAG, "updatePercentage: 퍼센티지는 구해지는지 p : "+p+", nDistanceDifference: "+nDistanceDifference);
        }
    }

    percentage = p;
    return percentage;
}
```



```
public double updatePercentage(int dest, int remain) {
    double p = 0;
    if (dest >= remain) { // ex 목적지까지 거리가 123, 남은거리가 122이나 123이면 실행
        //Log.e(TAG, "updatePercentage: dest랑remain 제대로 들어오는지 dest : "+dest+", remain : "+remain);
        nDistanceDifference = (dest-remain);
        if (nDistanceDifference > 0) {
            p = nDistanceDifference/(double)dest * 100.0;
            p =(int) (p * 100);
            p = p / 100.0;
            Log.e(TAG, "updatePercentage: 퍼센티지는 구해지는지 p : "+p+", nDistanceDifference: "+nDistanceDifference);
        }
    }

    percentage = p;
    return percentage;
}
```

서비스 쓰레드, 서비스 핸들러 주기적 메시지 호출

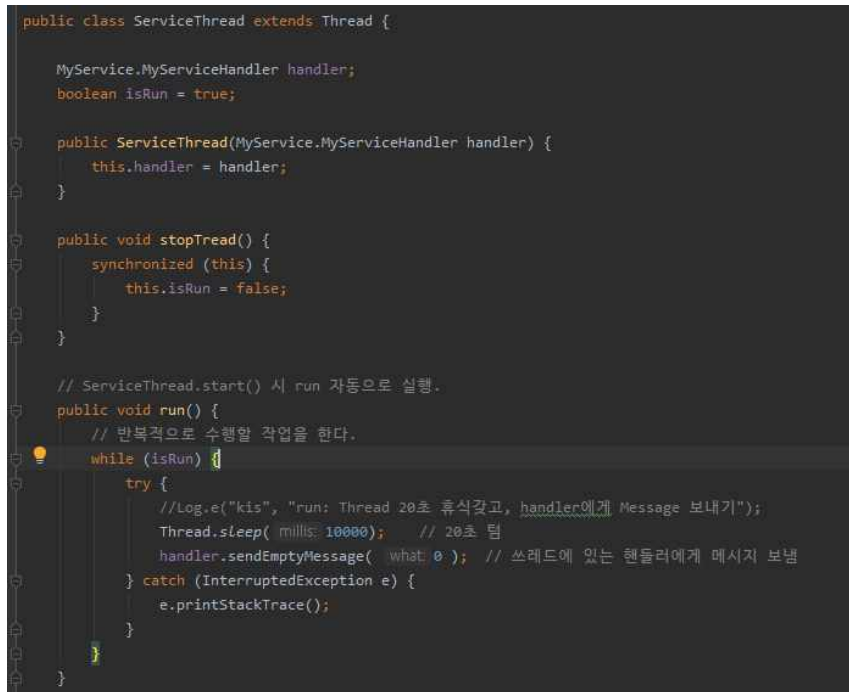
```
public class ServiceThread extends Thread {

    MyService.MyServiceHandler handler;
    boolean isRun = true;

    public ServiceThread(MyService.MyServiceHandler handler) {
        this.handler = handler;
    }

    public void stopTread() {
        synchronized (this) {
            this.isRun = false;
        }
    }

    // ServiceThread.start() 시 run 자동으로 실행.
    public void run() {
        // 반복적으로 수행할 작업을 한다.
        while (isRun) {
            try {
                //Log.e("kis", "run: Thread 20초 휴식갖고, handler에게 Message 보내기");
                Thread.sleep(10000); // 20초 텀
                handler.sendEmptyMessage( 0 ); // 쓰레드에 있는 핸들러에게 메시지 보냄
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
public class ServiceThread extends Thread {

    MyService.MyServiceHandler handler;
    boolean isRun = true;

    public ServiceThread(MyService.MyServiceHandler handler) {
        this.handler = handler;
    }

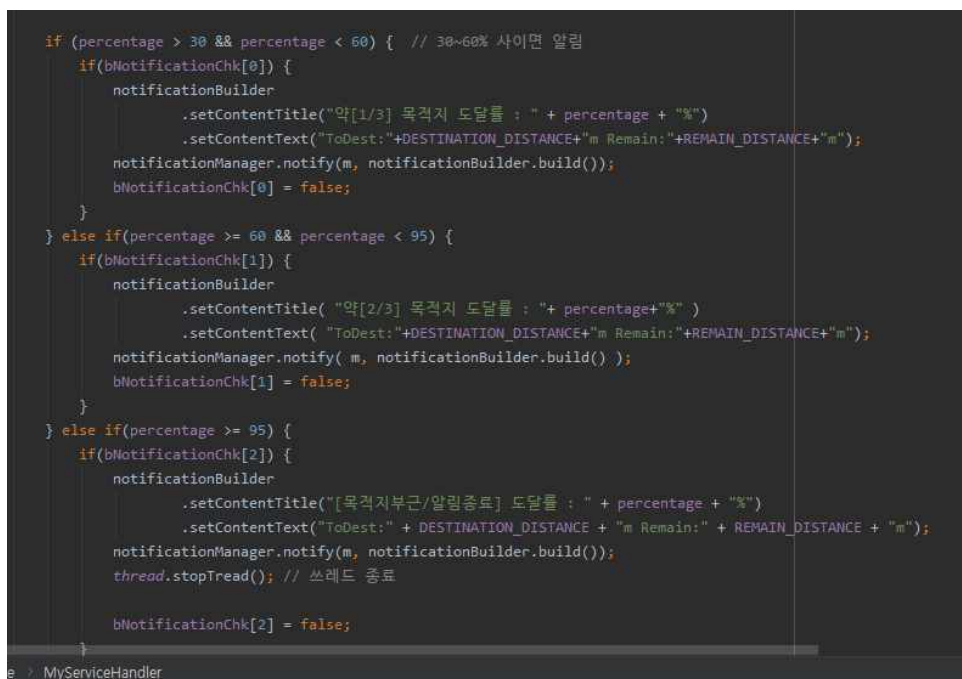
    public void stopTread() {
        synchronized (this) {
            this.isRun = false;
        }
    }

    // ServiceThread.start() 시 run 자동으로 실행.
    public void run() {
        // 반복적으로 수행할 작업을 한다.
        while (isRun) {
            try {
                //Log.e("kis", "run: Thread 20초 휴식갖고, handler에게 Message 보내기");
                Thread.sleep( mills: 10000); // 20초 텀
                handler.sendEmptyMessage( what: 0 ); // 쓰레드에 있는 핸들러에게 메시지 보냄
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

MyService클래스 내부 핸들러 클래스 : 거리 계산하여 알림메시지 보내기

```
if (percentage > 30 && percentage < 60) { // 30~60% 사이면 알림
    if(bNotificationChk[0]) {
        notificationBuilder
            .setContentType("약[1/3] 목적지 도달률 : " + percentage + "%")
            .setConteNtText("ToDest:" + DESTINATION_DISTANCE + "m
Remain:" + REMAIN_DISTANCE + "m");
        notificationManager.notify(m, notificationBuilder.build());
        bNotificationChk[0] = false;
    }
} else if (percentage >= 60 && percentage < 95) {
    if(bNotificationChk[1]) {
        notificationBuilder
            .setContentType( "약[2/3] 목적지 도달률 : " + percentage + "%" )
            .setConteNtText(
                "ToDest:" + DESTINATION_DISTANCE + "m
Remain:" + REMAIN_DISTANCE + "m");
        notificationManager.notify( m, notificationBuilder.build() );
        bNotificationChk[1] = false;
    }
} else if (percentage >= 95) {
    if(bNotificationChk[2]) {
        notificationBuilder
            .setContentType("[목적지부근/알림종료] 도달률 : " + percentage + "%")
            .setConteNtText("ToDest:" + DESTINATION_DISTANCE + "m Remain:" +
REMAIN_DISTANCE + "m");
        notificationManager.notify(m, notificationBuilder.build());
        thread.stopTread(); // 쓰레드 종료

        bNotificationChk[2] = false;
    }
}
}
```



```
1  if (percentage > 30 && percentage < 60) { // 30~60% 사이면 알림
2      if(bNotificationChk[0]) {
3          notificationBuilder
4              .setContentType("약[1/3] 목적지 도달률 : " + percentage + "%")
5              .setConteNtText("ToDest:" + DESTINATION_DISTANCE + "m Remain:" + REMAIN_DISTANCE + "m");
6          notificationManager.notify(m, notificationBuilder.build());
7          bNotificationChk[0] = false;
8      }
9  } else if (percentage >= 60 && percentage < 95) {
10     if(bNotificationChk[1]) {
11         notificationBuilder
12             .setContentType( "약[2/3] 목적지 도달률 : " + percentage + "%" )
13             .setConteNtText(
14                 "ToDest:" + DESTINATION_DISTANCE + "m Remain:" + REMAIN_DISTANCE + "m");
15         notificationManager.notify( m, notificationBuilder.build() );
16         bNotificationChk[1] = false;
17     }
18 } else if (percentage >= 95) {
19     if(bNotificationChk[2]) {
20         notificationBuilder
21             .setContentType("[목적지부근/알림종료] 도달률 : " + percentage + "%")
22             .setConteNtText("ToDest:" + DESTINATION_DISTANCE + "m Remain:" + REMAIN_DISTANCE + "m");
23         notificationManager.notify(m, notificationBuilder.build());
24         thread.stopTread(); // 쓰레드 종료
25
26         bNotificationChk[2] = false;
27     }
28 }
29 }
```

MyServiceHandler

DEFINES Class : Tmap API key 값과 REST 요청 구문 관리

```
static String PROJECT_KEY = "17xxx00L";
static String SECRET_KEY = "5f933980";
static String TMAP_SCH_POI_API_URL;
static String TMAP_REQ_SUMMARY_ROUTES;

static {
    try { // Tmap search
        // 경로 추천 REST
        TMAP_REQ_SUMMARY_ROUTES = "https://apis.openapi.sk.com/tmap/routes/pedestrian?version=1&appKey="+URLEncoder.encode(PROJECT_KEY, "UTF-8");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}

static final int REQ_PERMISSION_COARSE_LOCATION = 1001;
static final int REQ_PERMISSION_FINE_LOCATION = 1002;

static {
    try { // Tmap search
        // 명칭(POI) 통합 검색
        // https://apis.openapi.sk.com/tmap/pois?version={version}&page={page}&count={count}&searchKeyword={searchKeyword}&areaLLCode={areaLLCode}&areaLMCode={areaLMCode}&resCoordType={resCoordType}&searchType={searchType}&searchctpCd={searchctpCd}&radius={radius}&reqCoordType={reqCoordType}&centerLon={centerLon}&centerLat={centerLat}&multiPoint={multiPoint}&callback={callback}&appKey={appKey}
        TMAP_SCH_POI_API_URL = "https://apis.openapi.sk.com/tmap/pois?version=1&appKey="+URLEncoder.encode(PROJECT_KEY, "UTF-8") + "&searchKeyword=";
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}
}
```

```
static {
    try { // Tmap search
        // 경로 추천 REST
        TMAP_REQ_SUMMARY_ROUTES =
"https://apis.openapi.sk.com/tmap/routes/pedestrian?version=1&appKey="+URLEncoder.encode(PROJECT_KEY,"UTF-8");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}

static final int REQ_PERMISSION_COARSE_LOCATION = 1001;
static final int REQ_PERMISSION_FINE_LOCATION = 1002;

static {
    try { // Tmap search
        // 명칭(POI) 통합 검색
        //
https://apis.openapi.sk.com/tmap/pois?version={version}&page={page}&count={count}&searchKeyword={searchKey
word}&areaLLCode={areaLLCode}&areaLMCode={areaLMCode}&resCoordType={resCoordType}&searchType={searc
hType}&searchctpCd={searchctpCd}&radius={radius}&reqCoordType={reqCoordType}&centerLon={centerLon}&cent
erLat={centerLat}&multiPoint={multiPoint}&callback={callback}&appKey={appKey}
        TMAP_SCH_POI_API_URL
=
"https://apis.openapi.sk.com/tmap/pois?version=1&appKey="+URLEncoder.encode(PROJECT_KEY,"UTF-8")
+
"&searchKeyword=";
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}
}
```

DBHelper : sqlite 데이터베이스에 값을 저장하고 삭제하기 위한 util

```
public DBHelper(Context context) { super(context, DATABASE_NAME, null, 2); // 데이터베이스 생성 }

@Override
public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + REPORTS_TABLE_NAME + " ( " +
        REPORTS_COLUMN_IDX + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        REPORTS_COLUMN_DATE + " TEXT, " +
        REPORTS_COLUMN_DESTINATION + " TEXT, " +
        REPORTS_COLUMN_MUVEDDISTANCE + " TEXT, " +
        REPORTS_COLUMN_REACH + " TEXT )";
    db.execSQL(query);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS reports");
    onCreate(db);
}

// 삽입
public boolean insertReport(String _date, String _destination, String _moveddistance, String _reach) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("_date", _date);
    contentValues.put("_destination", _destination);
    contentValues.put("_moveddistance", _moveddistance);
    contentValues.put("_reach", _reach);
    db.insert("reports", null, contentValues);
    return true;
}
```

```
public DBHelper(Context context) {
    super(context, DATABASE_NAME, null, 2); // 데이터베이스 생성
}

@Override
public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + REPORTS_TABLE_NAME + " ( " +
        REPORTS_COLUMN_IDX + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        REPORTS_COLUMN_DATE + " TEXT, " +
        REPORTS_COLUMN_DESTINATION + " TEXT, " +
        REPORTS_COLUMN_MUVEDDISTANCE + " TEXT, " +
        REPORTS_COLUMN_REACH + " TEXT )";
    db.execSQL(query);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS reports");
    onCreate(db);
}

// 삽입
public boolean insertReport(String _date, String _destination, String _moveddistance, String _reach) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("_date", _date);
    contentValues.put("_destination", _destination);
    contentValues.put("_moveddistance", _moveddistance);
    contentValues.put("_reach", _reach);
    db.insert("reports", null, contentValues);
    return true;
}
```

시간 분, 초 / 위치 km, m 계산 편하게 하도록 도와주는 Util 클래스

```
public class Util {  
  
    static public String secToTimeText(int nSecTime){  
        int secToMinute = nSecTime / 60;  
  
        int hours = secToMinute / 60;  
        int minute = secToMinute - hours * 60;  
  
        if(hours != 0){  
            return hours + "시간 " + minute + "분";  
        }else{  
            return minute + "분";  
        }  
    }  
}  
  
static public String meterIntToDistanceText(int nDistance) {    // 현 위치  
    // 값이 크게 나오므로  
    int km = nDistance / 1000; // 킬로미터  
    int m = nDistance - (km * 1000); // 미터  
  
    if(km != 0){  
        return km + "." + m/100 + "km";  
    }else{  
        return m + "m"; // 킬로미터가 아니면 미터로 그냥 출력  
    }  
}
```

```
package com.example.afinal;  
import java.text.DecimalFormat;  
public class Util {  
    static public String secToTimeText(int nSecTime){  
        int secToMinute = nSecTime / 60;  
  
        int hours = secToMinute / 60;  
        int minute = secToMinute - hours * 60;  
  
        if(hours != 0){  
            return hours + "시간 " + minute + "분";  
        }else{  
            return minute + "분";  
        }  
    }  
}  
  
static public String meterIntToDistanceText(int nDistance) {    // 현 위치  
    // 값이 크게 나오므로  
    int km = nDistance / 1000; // 킬로미터  
    int m = nDistance - (km * 1000); // 미터  
  
    if(km != 0){  
        return km + "." + m/100 + "km";  
    }else{  
        return m + "m"; // 킬로미터가 아니면 미터로 그냥 출력  
    }  
}  
}
```

자바에서 VO, DTO, Beans, getter/setter는 별도 생성하지 않음

다음 API에서는 POI라는 구문을 사용하여, POIModel : 목적지 검색정보를 표현

Serializable 은 직렬화를 말하는데 JSON 변환, 통신할 때 직렬화 역직렬화 과정이 요구됨

```
package com.example.afinal;

import java.io.Serializable;

// POIModel 관점 지점(POI : Point of interest 데이터 모델
public class POIModel implements Serializable { // 직렬화 데이터 통신
    // Tmap response에서 참고
    String name; // 시설물 명칭
    String frontLat; // 시설물 입구 위도 좌표
    String frontLon; // 시설물 입구 경도 좌표
    String upperAddrName; //서울
    String middleAddrName; // ex) 용산구",
    String lowerAddrName; //동자동
    String radius; // 반지름, 반경

    Double kcal; // 칼로리 계산
    Boolean isSelected = false; // 선택여부
}
```

// POIModel 관점 지점(POI : Point of interest 데이터 모델

public class POIModel implements Serializable { // 직렬화 데이터 통신

// Tmap response에서 참고

String name; // 시설물 명칭

String frontLat; // 시설물 입구 위도 좌표

String frontLon; // 시설물 입구 경도 좌표

String upperAddrName; //서울

String middleAddrName; // ex) 용산구",

String lowerAddrName; //동자동


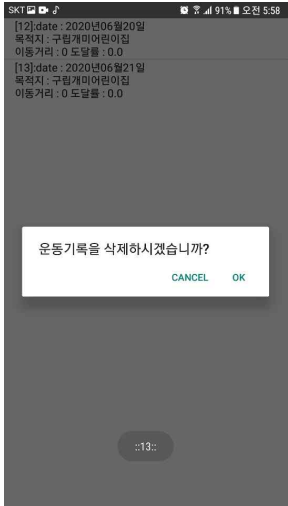
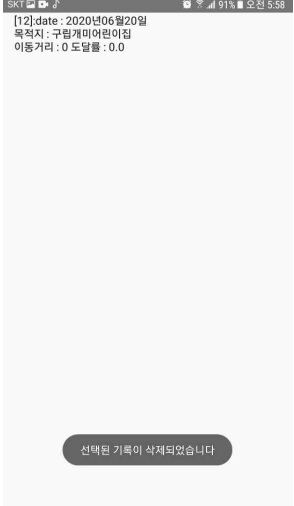
String radius; // 반지름, 반경

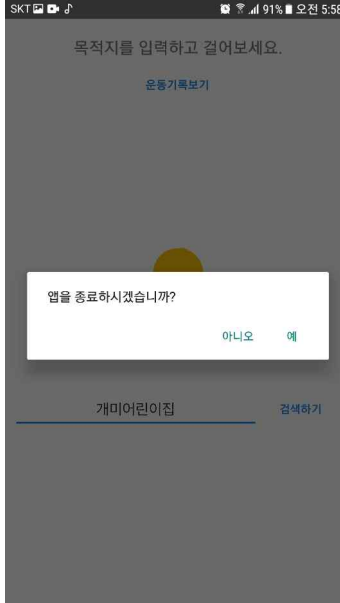
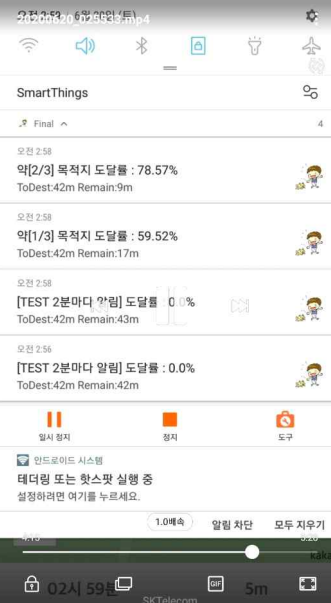

Double kcal; // 칼로리 계산

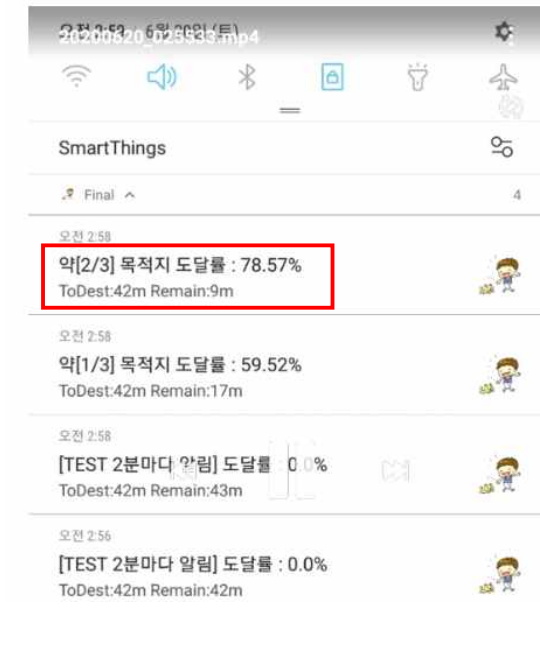
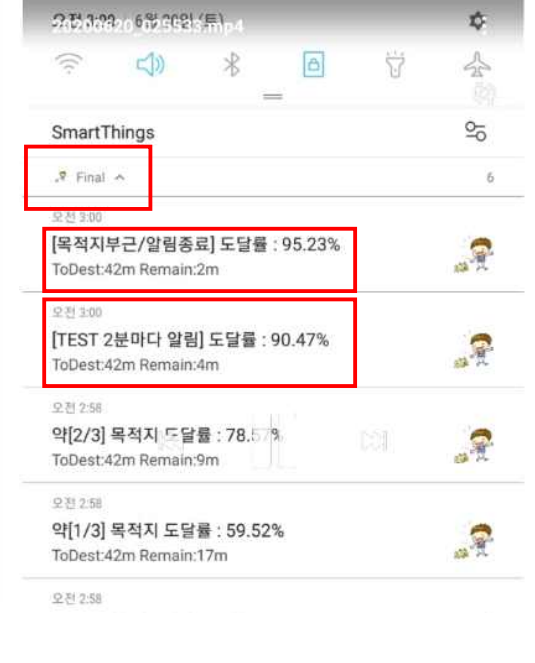
Boolean isSelected = false; // 선택여부

}

4. 결과

<p>1. Intro 화면</p> 	<p>2. Main 화면</p> 	<p>3. Search 결과 화면</p> 
<p>4. SummaryRoute 화면</p> 	<p>5. Navigation 화면</p> 	<p>6. 종료 다이얼로그 화면</p> 
<p>7. 운동기록 관리 화면</p> 	<p>8. 아이템 삭제 다이얼로그</p> 	<p>9. 삭제된 모습 나타낸 화면</p> 

10. 앱 종료 화면	11. 알림메시지 화면	12. 알림메시지 화면
		

13. 남은 거리에 따라 알림 메시지	
	

5. 소감

이번 학기말 프로젝트를 통해서 전 안드로이드 프로그래밍에 대해서 깊게 공부할 수 있는 시간이었다고 생각합니다. 다만 아쉬운 점은 짧은 기간 내에 많은 것을 담아야 하고, 많은 것을 처리해야 하다 보니 아쉬움으로 남습니다. 욕심은 많았지만 많은 것을 담아내지 못한 것 같습니다. 이 프로그램에는 아마도 아직 오류가 있을 것입니다. 하지만 짧은 기간 안에 많은 것을 쏟아내기 위해 많은 것을 공부하고 지식을 많이 채운 시간으로 삼고자 합니다. 이 프로젝트를 통해 다양한 API 사용방법과 다양한 유틸 사용방법 등 모바일 프로그래밍에 관해 구조적으로, 기술적으로 많은 것을 익힌 것 같아 뿌듯하게 생각합니다. 물론 짧은 기간에 많은 반복학습을 한 것은 아니지만, 방대한 양의 정보를 접하고 검색하고 덧붙이고 사용하고 하면서 정보 기술을 활용하는 스킬이 많이 늘은 것 같습니다.

오류에 대한 해결과 풀리지 않는 문제들에 대해 많은 고민을 하며 많은 시간을 썼던 것 같습니다. 프로그래밍을 하다보면 항상 테스트와 오류해결에도 많은 시간을 쓰게 됩니다.

스마트폰은 작지만 엄청나게 많은 기능을 담고 있고, 많은 것을 제공한다는 것을 몸소 경험하게 된 모바일 프로그래밍을 통해서 한 단계 더 성장한 것 같은 기분이 듭니다. 나중에 스마트폰 어플리케이션을 개발하는 멋진 개발자가 될 수 있도록 하나의 포트폴리오로 삼을 수 있도록 정성들인 만큼 결과도 좋았으면 좋겠다는 생각을 합니다.

코로나로 인해 대면수업을 하지는 못했지만, 그래도 많은 것을 알려주시려고 노력해주신 교수님께도 감사를 드립니다. 교과목 외에도 많은 소스코드를 보여주며 알려주셨기에 비대면 수업 상황에서도 많은 것을 얻어갈 수 있었습니다.

감사합니다.

안양대학교 컴퓨터공학전공 2017E7441 김이삭
isaac7263@naver.com

6. 참고자료

[책]그림으로 쉽게 설명하는 안드로이드 프로그래밍, Do it 안드로이드 앱 프로그래밍

[안드로이드 서비스 백그라운드 스레드 알림]

<https://twinw.tistory.com/50>

[Glide : 이미지 라이브러리, GIF 이미지 로딩]

<https://lakue.tistory.com/10>

<https://gun0912.tistory.com/17>

[키보드 안보임 처리]

<https://kkangsnote.tistory.com/35>

[키 해쉬 구하기]

<https://thisisspear.tistory.com/46>

<https://lakue.tistory.com/11>

[카카오 지도 API]

<https://apis.map.kakao.com/android/guide/#introduce>

<https://apis.map.kakao.com/android/documentation/>

[FusedLocationProviderClient 현재 위치 얻기]

<https://developer.android.com/training/location/retrieve-current?hl=ko>

[길찾기 T map API, SK API (목적지)]

<https://openapi.sk.com/content/API>

<http://tmapapi.sktelecom.com/>

[명칭(POI) 통합검색 : REST / GET Method : SK 아이디 있어야 함.]

<https://openapi.sk.com/resource/apidoc/indexView?docSeq=7&type=1>

[T map 보행자 경로 안내 문서 (참고)]

<http://tmapapi.sktelecom.com/main.html#webservice/docs/tmapRoutePedestrianDoc>

[안드로이드 HTTP통신 Okhttp 사용 예제]

<https://thereclub.tistory.com/46>

[RequestBody 사용, Http 통신]

<https://m.blog.naver.com/PostView.nhn?blogId=netrance&logNo=220728243852&proxyReferer=https:%2F%2Fwww.google.com%2F>

6. 참고자료

[안드로이드 JSON 파싱]

<https://blog.naver.com/PostView.nhn?blogId=qbxlvnf11&logNo=221432824912&categoryNo=44&parentCategoryNo=0&viewDate=¤tPage=1&postListTopCurrentPage=1&from=postView>

[JSON 파싱이후 스트링을 -> 모델로 파싱 : 사용 라이브러리 GSON]]

<https://re-build.tistory.com/41>

[리사이클러 뷰 참고 메인, xml, 어댑터리스트, 아이템 목록]

<https://webnautes.tistory.com/1214>

[안드로이드 타이머 클래스-타이머 태스크]

<http://blog.naver.com/PostView.nhn?blogId=ssarang8649&logNo=220947884163>

[서비스가 작동중인지 확인하는 방법-싱글톤 패턴이랑 비슷함]

<https://c10106.tistory.com/2872>

[어플 강제 종료에 대한 이벤트 발생]

<https://m.blog.naver.com/PostView.nhn?blogId=monster7575&logNo=220430274475&proxyReferer=https:%2F%2Fwww.google.com%2F>

[스레드 죽이기]

<https://stackoverflow.com/questions/13416879/show-a-dialog-in-thread-setdefaultuncaughtexceptionhandler>

<https://www.androidpub.com/1302717>

<https://link2me.tistory.com/1343>

[Location 거리구하기]

<https://croak.tistory.com/113>

<https://developer88.tistory.com/70>

[문자열 추출]

<http://blog.naver.com/PostView.nhn?blogId=rorean&logNo=221582429295&categoryNo=16&parentCategoryNo=0&viewDate=¤tPage=1&postListTopCurrentPage=1&from=postView>

[액티비티 전달하며 기존 액티비티 제거하기]

<https://minyoungjung.github.io/%EC%95%88%EB%93%9C%EB%A1%9C%EC%9D%B4%EB%93%9C/2017/07/12/android-activity-close/>