

---

# Kotlin을 이용한 Android 프로그래밍

## 구글지도와 위치 추적 앱 만들기

---

# Contents

---

- I. Google Map과 Gps를 이용한 트래킹 앱 만들기

# 지도와 GPS

▶ 프로젝트 명 : GpsMap

▶ 기능

- ▶ 구글맵에 GPS로 현재 위치 정보를 얻어 지도에 표시
- ▶ 주기적으로 현재 위치를 갱신하며 위치 변경 내용을 선으로 표시(트래킹)

▶ 구성요소

- ▶ Google Maps Activity : 지도를 표시하는 기본 템플릿
- ▶ FusedLocationProviderClient : 현재 위치 정보를 얻는 클래스

▶ 라이브러리 설정

- ▶ Anko 라이브러리
- ▶ play-services-maps : 구글 지도 라이브러리(Google Maps Activity 선택으로 자동 설정)
- ▶ play-services-location : 위치 정보 라이브러리

# 지도와 GPS

## ▶ 프로젝트 설계

- ▶ 액티비티 하나로 구성되며 Google Maps Activity 템플릿을 활용
- ▶ 지도를 활용하여 현재 위치 정보를 사용하려면 실행 중 권한 요청을 수행해야 함
- ▶ 주기적으로 현재 위치를 업데이트하려면 액티비티 생명주기에 따라 위치 업데이트 리스너를 등록 및 해제

## ▶ 구현 순서

- 1) 프로젝트 생성
- 2) 구글 지도 표시
- 3) 현재 위치 정보 가져오기
- 4) 주기적으로 현재 위치 정보 업데이트
- 5) 위치 변경 정보를 선으로 표시

# 지도와 GPS

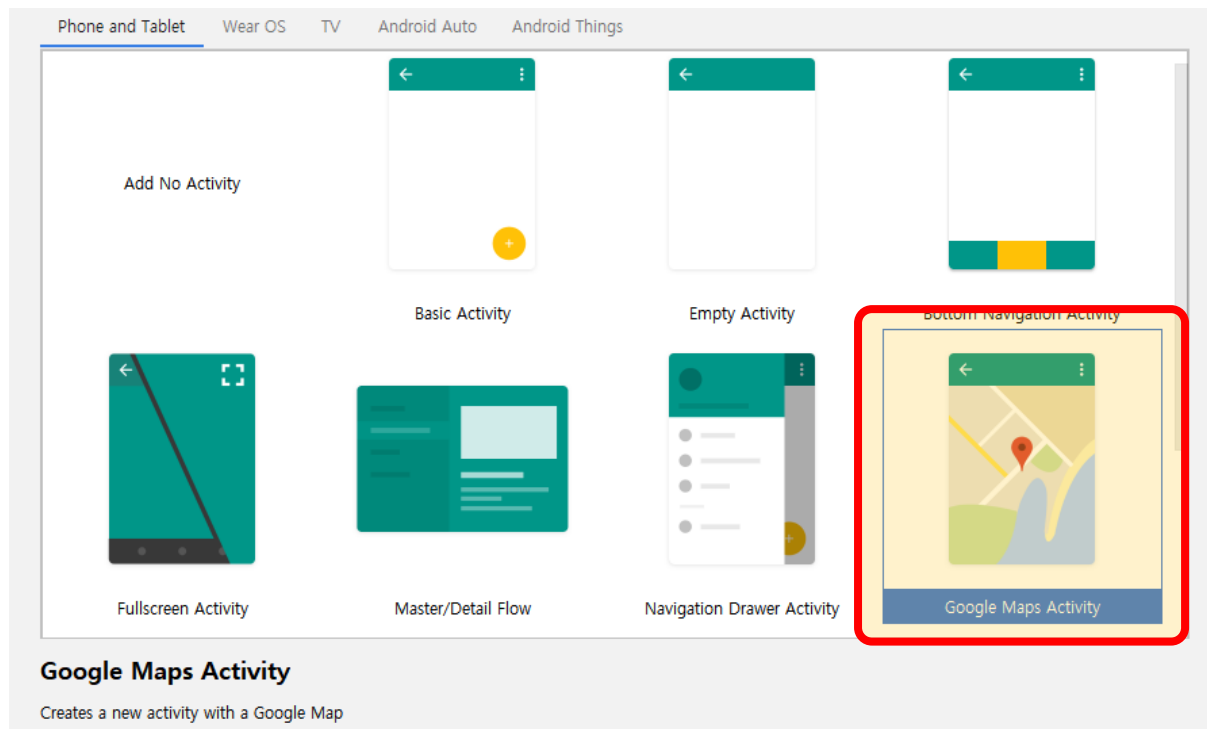
## ▶ 프로젝트 생성

▶ 프로젝트 명 : GpsMap

▶ minSdkVersion : 19(android 4.4)

▶ 기본 액티비티 Google Maps Activity

▶ 구글 지도를 편리하게 사용할 수 있도록 구글 맵 액티비티를 선택



# 지도와 GPS

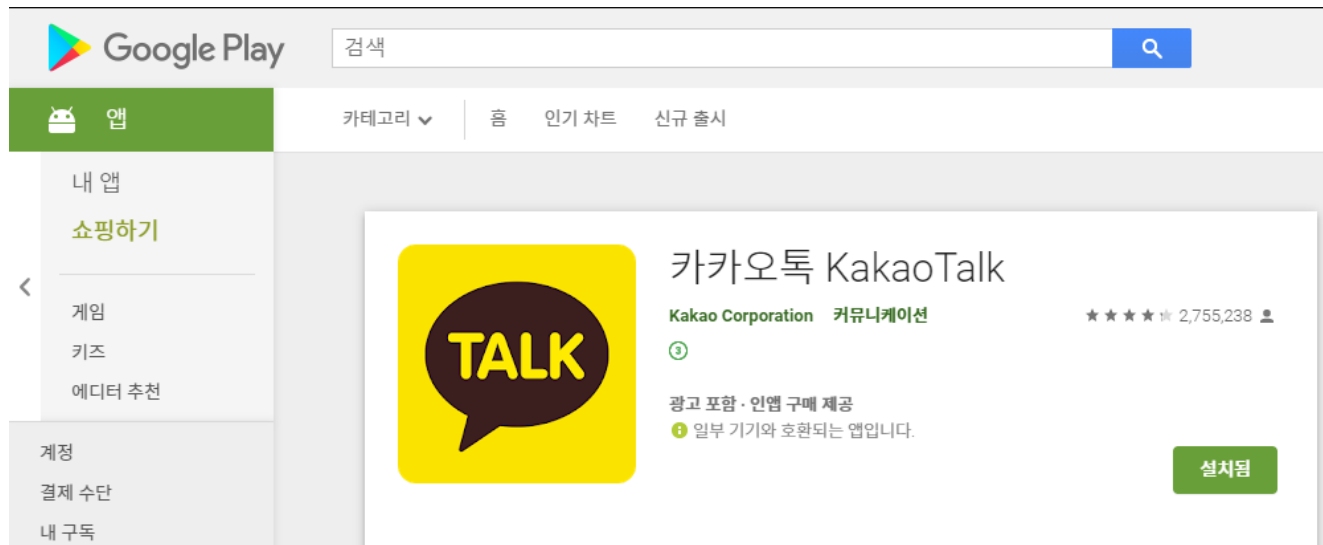
## ▶ 프로젝트 생성 정보

- ▶ package name은 안드로이드 앱의 기본 아이디이므로 반드시 유니크한 이름으로 설정
- ▶ 구글 플레이 스토어에서 동일한 아이디의 앱을 간단하게 검색하는 방법
  - ▶ 간혹 앱이 등록된 후 공개를 중지하면 해당 페이지에서는 나오지 않음

`https://play.google.com/store/apps/details?id=[application id]`

- ▶ 카카오톡의 링크는 아래와 같음

`https://play.google.com/store/apps/details?id=com.kakao.talk`



# 지도와 GPS

## ▶ 프로젝트 생성 정보

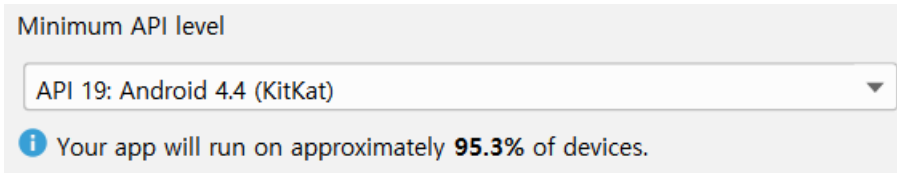
▶ minSdkVersion은 최소 지원버전을 나타냄

▷ 앱이 지원하는 안드로이드 최하 버전

▶ 최하 버전 미만의 기기에서는 플레이 스토어에서 앱이 검색되지 않고 설치도 불가

▶ 낮을 수록 하위 버전에 대한 고려로 인하여 개발이 까다롭지만 더 많은 폰을 지원

▷ 버전 설정 시 시중에 설치할 수 있는 기기 비율을 확인 할 수 있음



▷ 높은 버전일수록 개발은 용이

▶ 구글 지도를 앱에서 사용할 때 가장 쉬운 방법은 맵 액티비티를 사용하는 것

▷ 기본 액티비티 화면에서 Google Maps Activity를 선택

# 지도와 GPS

## ▶ Google Maps Activity

- ▶ MapsActivity를 선택하면 지도를 사용하는 play-service-maps 라이브러리가 모듈 수준의 build.gradle에 자동으로 추가됨
- ▶ 위치 정보를 사용하기 위해 Play-service-location 라이브러리를 모듈 수준의 build.gradle에 추가
- ▶ `implementation 'com.google.android.gms:play-services-location:15.0.1'`

dependencies {

```
implementation 'com.google.android.gms:play-services-location:15.0.1'
```

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
```

```
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
```

```
implementation 'com.android.support:appcompat-v7:28.0.0'
```

```
implementation 'com.google.android.gms:play-services-maps:16.1.0'
```

```
testImplementation 'junit:junit:4.12'
```



# 지도와 GPS

## ▶ Gradle에 Anko 라이브러리 의존성 추가

▶ 프로젝트 창에서 모듈 수준의 build.gradle 파일에 아래코드 삽입 후 sync 클릭

▶ implementation "org.jetbrains.anko:anko-commons:0.10.5"

### ▼ Gradle Scripts

build.gradle (Project: BmiCalculator)

build.gradle (Module: app)

gradle-wrapper.properties (Gradle Version)

proguard-rules.pro (ProGuard Rules for app)

gradle.properties (Project Properties)

settings.gradle (Project Settings)

local.properties (SDK Location)

dependencies {

implementation "org.jetbrains.anko:anko-commons:0.10.5"

implementation fileTree(dir: 'libs', include: ['\*.jar'])

implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:\$kotlin\_version"

implementation 'com.android.support:appcompat-v7:28.0.0'

implementation 'com.android.support.constraint:constraint-layout:1.1.3'

testImplementation 'junit:junit:4.12'

androidTestImplementation 'com.android.support.test:runner:1.0.2'

androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

}

▶ 싱크 클릭

Sync Now

# 지도와 GPS

## ▶ 구글 지도를 표시하는 과정

▶ 구글 지도를 쉽게 사용할 수 있도록 기본 액티비티로 MapsActivity를 선택

▶ 구글 지도를 사용하려면 API키를 발급 받아야 함

▷ 2018년 7월 13일부터 Google 지도를 사용하는 사이트 중 개별 API키를 적용하지 않은 경우 지도 표시가 되지 않음

▷ 2018년 7월 16일부터 Google 지도 API 정책 변경에 따른 새로운 요금제가 적용되고, 무료 제공 한도(일간 25,000건에서 월간 28,500건 지도 로드로 변경)가 매우 축소

▶ 구현 순서

1) 구글 지도 API 키 발급 받기

2) MapsActivity 기본 코드 분석

# 지도와 GPS

## ▶ 구글 지도 API 키 발급 받기

▶ 기본적으로 Google\_maps\_api.xml 파일이 열려 있으며 google\_maps\_key 문자열 리소스에 API키를 입력해야 함

▶ 해당 API키 가 있어야 구글 맵을 사용할 수 있음

▶ API키를 발급 받으면 아래 표시한 YOUR\_KEY\_HERE 에 키를 입력

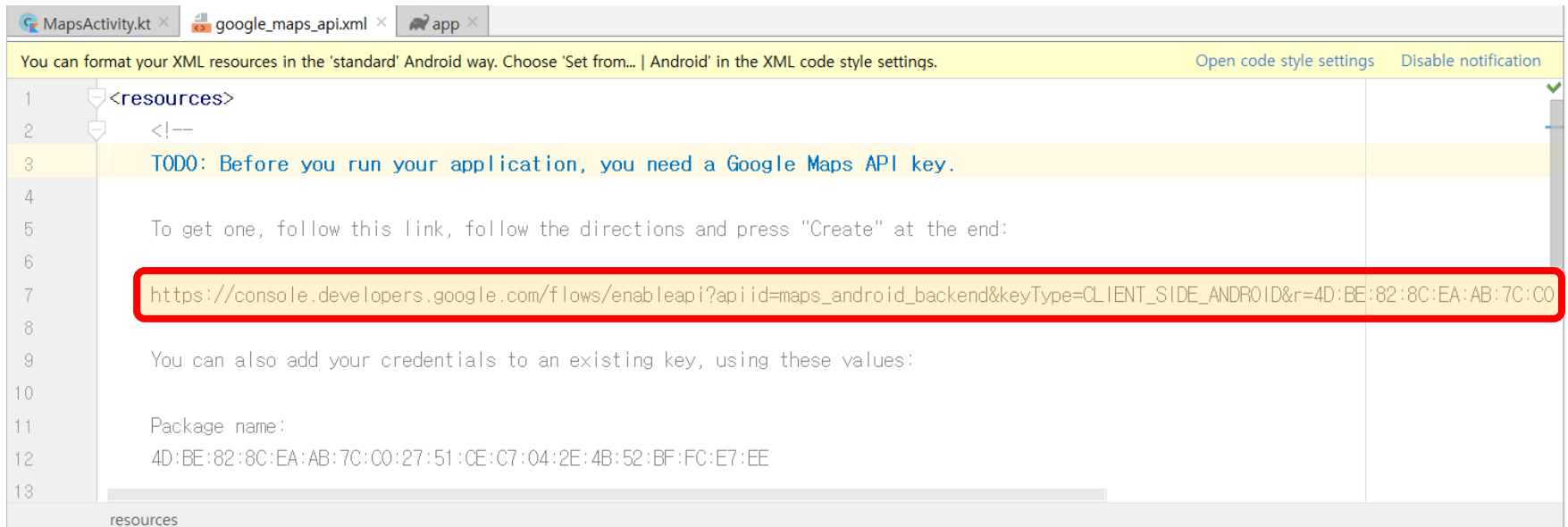


```
14  SHA-1 certificate fingerprint:
15  4D:BE:82:8C:EA:AB:7C:C0:27:51:CE:C7:04:2E:4B:52:BF:FC:E7:EE
16
17  Alternatively, follow the directions here:
18  https://developers.google.com/maps/documentation/android/start#get-key
19
20  Once you have your key (it starts with "Alza"), replace the "google_maps_key"
21  string in this file.
22  -->
23  <string name="google_maps_key" translatable="false" templateMergeStrategy="preserve" YOUR_KEY_HERE />
24  </resources>
```

# 지도와 GPS

## ▶ 구글 지도 API 키 발급 받기

- ▶ Google\_maps\_api.xml 파일에서 http로 시작하는 링크를 복사하여 웹 브라우저에서 해당 페이지를 표시

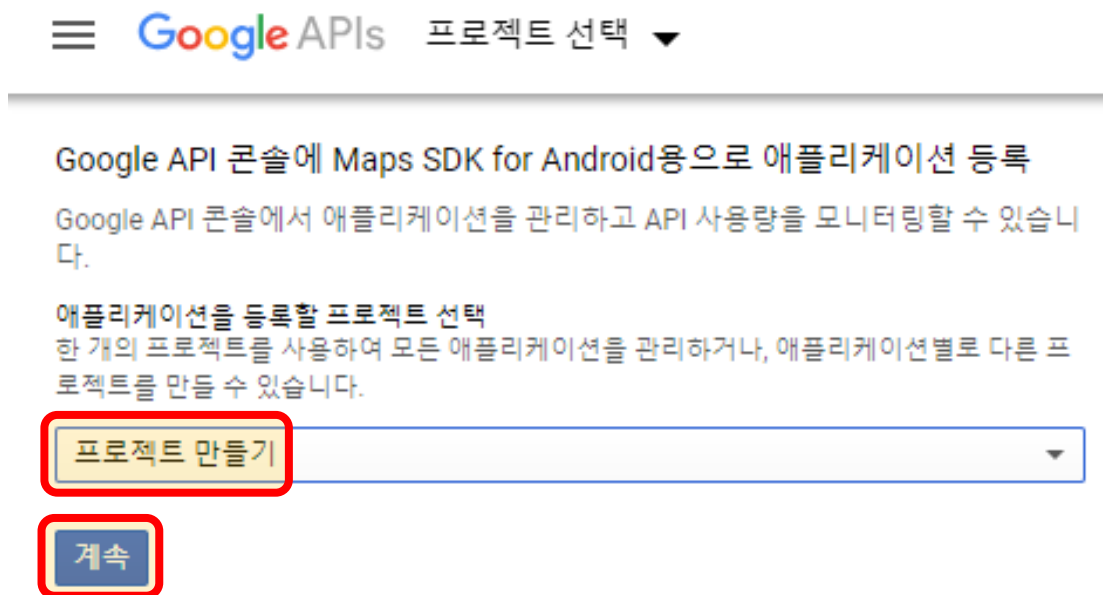


```
1 <resources>
2 <!--
3 TODO: Before you run your application, you need a Google Maps API key.
4
5 To get one, follow this link, follow the directions and press "Create" at the end:
6
7 https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=4D:BE:82:8C:EA:AB:7C:00
8
9 You can also add your credentials to an existing key, using these values:
10
11 Package name:
12 4D:BE:82:8C:EA:AB:7C:00:27:51:CE:C7:04:2E:4B:52:BF:FC:E7:EE
13
```

# 지도와 GPS

## ▶ 구글 지도 API 키 발급 받기

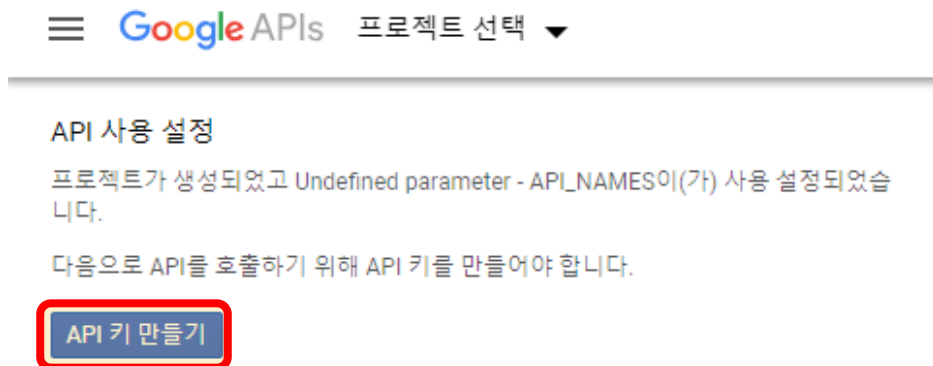
- ▶ 해당 링크로 이동하면 구글 지도를 사용하는 앱을 등록하는 구글 API 콘솔 화면이 표시됨
- ▶ [프로젝트 만들기]에서 새로운 이름의 프로젝트를 생성할 수 있고 이전에 만든 프로젝트가 있으면 선택할 가능
- ▶ 예제에서는 화면에 있는 [계속]을 클릭하여 넘어 감
  - ▶ 자동으로 프로젝트명이 결정



# 지도와 GPS

## ▶ 구글 지도 API 키 발급 받기

### ▶ 다음 화면에서 API 키 만들기 클릭



### ▶ 생성된 키를 복사 아이콘을 클릭하여 API 클립 보드에 복사

#### ▶ API 키마다 사용횟수 제한이 있기 때문에 노출하지 않도록 함

#### ▶ 안전하게 메모장에 복사

### API 키 생성 완료

애플리케이션에서 이 키를 사용하려면 키를 `key=API_KEY` 매개변수로 전달하세요.



⚠ 키를 제한하여 프로덕션 환경에서 무단 사용을 방지하세요.

# 지도와 GPS


## ▶ 구글 지도 API 키 발급 받기

▶ API 키는 분실하거나 노출하지 않도록 주의

▶ API 키마다 사용회수에 제한이 있음

▶ 2018년부터 유료화

▶ API 키를 Google\_maps\_api.xml 파일에 붙여넣기



```
MapsActivity.kt x google_maps_api.xml x app x
You can format your XML resources in the 'standard' Android way. Choose 'Set from... | Android' in the XML code style settings. Open code style settings
14 SHA-1 certificate fingerprint:
15 4D:BE:82:8C:EA:AB:7C:C0:27:51:CE:C7:04:2E:4B:52:BF:FC:E7:EE
16
17 Alternatively, follow the directions here:
18 https://developers.google.com/maps/documentation/android/start#get-key
19
20 Once you have your key (it starts with "Alza"), replace the "google_maps_key"
21 string in this file.
22 →
23 <string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">Key</string>
24 </resources>
```

# 지도와 GPS

## ▶ 실행해 보기

▶ 앱을 실행하여 지도가 표시되는지 확인

▶ 아래와 같이 보이면 성공





# 지도와 GPS

## ▶ MainActivity 코드 분석

- ▶ 구글 맵을 사용하기 위하여 OnMapReadyCallback을 implements

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
```

- ▶ onMapReady(googleMap: GoogleMap)가 이미 재정의되어 있음

- map이 사용가능한 상태가 되면 호출되는 콜백 메소드
- marker나 camera 이동 등을 설정

- ▶ 기본적으로 호주의 시드니로 맵을 표시

- 마커와 카메라를 시드니로 표시

```
override fun onMapReady(googleMap: GoogleMap) { 지도가 준비되면 GoogleMap 객체를 가져옴
    ■Map = googleMap

    // Add a marker in Sydney and move the camera
    val sydney = LatLng(-34.0, 151.0) 호주 시드니의 위도, 경도 / 서울역 37.555744, 126.970431
    ■Map.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))
    ■Map.moveCamera(CameraUpdateFactory.newLatLng(sydney))
}
```

# 지도와 GPS

## ▶ MainActivity 코드 분석

### ▶ onCreate()

▶ 프래그먼트 매니저로부터 SupportMapFragment를 가져옴

- findFragmentById(R.id.map)으로 미리 만들어진 지도 화면을 가져옴 - MapView

▶ getMapAsync()메서드로 GoogleMap 객체를 화면에 표시하고 알림

```
val mapFragment = supportFragmentManager
    .findFragmentById(R.id.map) as SupportMapFragment
mapFragment.getMapAsync(this)
```

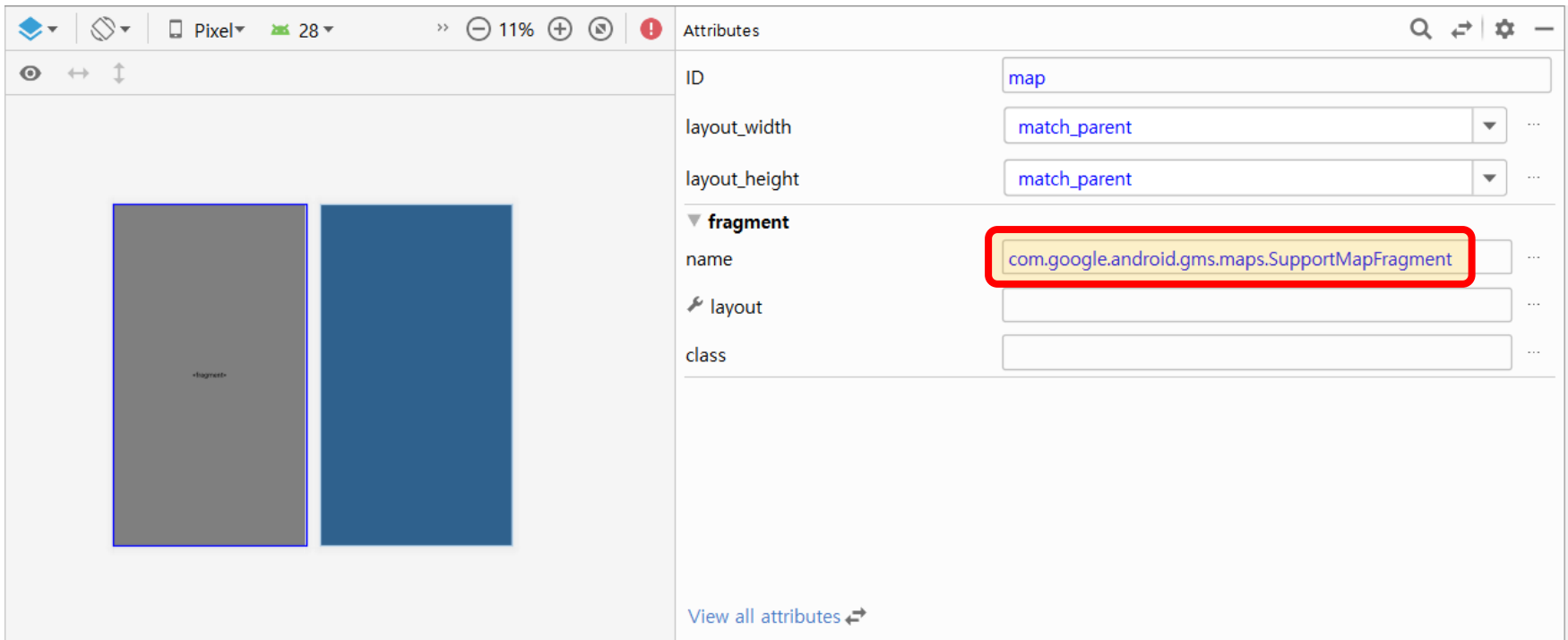
# 지도와 GPS

## ▶ 레이아웃 파일 분석

▶ activity\_maps.xml에 화면에 가득찬 프래그먼트 확인

▶ name : com.google.android.gms.maps.SupportMapFragment

▶ 구글 지도가 내장된 프래그먼트로 play\_services-maps 라이브러리에서 제공됨



# 지도와 GPS

## ▶ 레이아웃 파일 분석

▶ 구글 지도가 내장된 프래그먼트로 play\_services-maps 라이브러리에서 제공됨

▶ build.gradle 파일을 열어보면 play-services-maps 라이브러리의 의존성이 추가되어 있음



# 지도와 GPS

## ▶ 주기적으로 현재 위치 정보 업데이트하기

### ▶ 구현 순서

- 1) 매니페스트에 위치 권한 추가
- 2) onResume() 메서드에서 위치 정보 요청
- 3) 위치 정보 갱신 콜백 정의
- 4) onPause () 메서드에서 위치 정보 요청 중지

# 지도와 GPS

## ▶ 위치 권한 확인

▶ 이 권한은 위험 권한이므로 사용할 때는 실행 중 권한을 요청해야 함

▶ 위험 권한은 실행 중 권한 요청이 필요

▶ MapActivity를 선택하였으므로 이미 메니페스트에 권한이 추가되어 있음

▶ `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>`

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kr.ac.kpu.gpsmaps">
```

```
<!--
```

```
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
Google Maps Android API v2, but you must specify either coarse or fine
location permissions for the 'MyLocation' functionality.
```

```
-->
```

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

```
<application
```

```
    android:allowBackup="true"
```

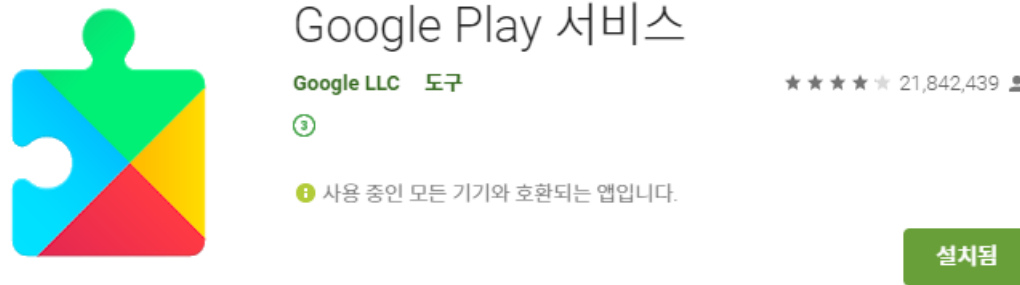
```
    android:icon="@mipmap/ic_launcher"
```

# 지도와 GPS

## ▶ 위치 정보 요청

▶ 구글 플레이 서비스를 최신 버전으로 업데이트 해야 위치 서비스에 연결됨

▷ <https://play.google.com/store/apps/details?id=com.google.android.gms>



▷ 구글 플레이 서비스는 안드로이드 계정 동기화나 앱 업데이트 설치 등을 관리하는 서비스

# 지도와 GPS

## ▶ 위치 정보 요청

- ▶ 위치 서비스에 연결된 앱은 FusedLocationProviderClient 클래스의 requestLocationUpdates () 메서드를 호출하여 위치 정보를 요청할 수 있음

```
requestLocationUpdates(locationRequest : LocationRequest,  
                        locationCallback : LocationCallback, looper : Looper)
```

- ▷ locationRequest : 위치 요청 객체
- ▷ locationCallback : 위치가 갱신되면 호출되는 콜백
- ▷ looper : 특정 루퍼 스레드를 지정함. 특별한 경우가 아니라면 null 지정



# 지도와 GPS

## ▶ 위치 정보 요청

▶ 위 코드는 위치 정보를 요청하는 코드로, 액티비티가 화면에 보일 때만 수행

▷ onResume() 에서 위치 정보를 요청하고 onPause() 에서 위치 정보 요청을 중단

```
override fun onResume() {  
    super.onResume()  
    addLocationListener()  
}  
  
private fun addLocationListener() {  
    fusedLocationProviderClient.requestLocationUpdates(locationRequest,  
        locationCallback,  
        null);  
}
```

# 지도와 GPS

## ▶ 위치 정보 요청

▶ requestLocationUpdates () 메서드의 첫번째 인자

▷ LocationRequest 객체는 위치 정보를 요청하는 시간 주기를 설정하는 객체

```
locationRequest = LocationRequest()  
// GPS 우선  
locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY  
// 업데이트 인터벌  
// 위치 정보가 없을 때는 업데이트 안 함  
// 상황에 따라 짧아질 수 있음, 정확하지 않음  
// 다른 앱에서 짧은 인터벌로 위치 정보를 요청하면 짧아질 수 있음  
locationRequest.interval = 10000  
// 정확함. 이것보다 짧은 업데이트는 하지 않음  
locationRequest.fastestInterval = 5000
```

▷ 참고: <https://developer.android.com/training/location/change-location-settings>

# 지도와 GPS

## ▶ 위치 정보 요청

### ▶ requestLocationUpdates ()의 두번째 인자

▷ LocationCallback 객체는 위도와 객체 정보를 가지고 있음

▷ lastLocation 프로퍼티로 최근 현재 위치에 대한 Location 객체 얻음

```
inner class MyLocationCallback : LocationCallback() {  
    override fun onLocationResult(locationResult: LocationResult?) {  
        super.onLocationResult(locationResult)  
    }  
}
```

# 지도와 GPS

## ▶ 위치 정보 요청

### ▶ MapsActivity.kt 파일을 열고 다음과 같이 코드 추가

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {  
    private lateinit var mMap: GoogleMap  
    //위치 정보를 주기적으로 얻는데 필요한 객체들을 선언  
    //위치 정보를 요청하는 requestLocationUpdates()메서드 사용을 위하여 아래 객체 선언(늦은 초기화)  
    private lateinit var fusedLocationProviderClient: FusedLocationProviderClient  
    //객체의 위치정보 요청의 시간 주기를 설정하는 객체 선언  
    private lateinit var locationRequest: LocationRequest  
    //최근 위치에 대한 정보를 가져올 수 있도록 locationCallBack클래스를 상속한 MyLocationCallBack클래스 타  
    입의 객체 선언  
    private lateinit var locationCallback: MyLocationCallBack  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)
```

# 지도와 GPS

## ▶ 위치 정보 요청

▶ MyLocationCallback 은 MapsActivity 클래스의 내부 클래스로 생성

▶ LocationResult 객체를 반환

▶ locationCallBack은 requestLocationUpdates ()의 두번째 인자

▶ LocationCallBack 객체는 위도와 객체 정보를 가지고 있음

▶ lastLocation 프로퍼티로 가장 최근의 현재 위치에 대한 Location 객체 얻음

▶ 아래 코드는 맨아래 MapsActivity클래스 내부에 작성

```
inner class MyLocationCallback : LocationCallback() {  
    override fun onLocationResult(locationResult: LocationResult?) {  
        super.onLocationResult(locationResult)  
        val location = locationResult?.lastLocation  
        //기기의 GPS설정이 꺼져있거나 정보를 얻을 수 없는 경우 null일 수 있음  
        //그러므로 null이 아닐때 위치정보를 가져오고 해당 위치로 카메라를 이동  
        location?.run {  
            val latLng = LatLng(latitude, longitude)  
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 17f))//줌 레벨 : 153m정도  
        }  
    }  
}  
}  
} //MapsActivity 클래스 블록
```

# 지도와 GPS

## ▶ inner 클래스

### ▶ 클래스 내부에 다른 클래스를 중첩 시킬 수 있음

```
class AAA {  
    private val aaa: Int = 10  
  
    class BBB {  
        fun bbb() = 100  
    }  
}
```

### ▶ 내부 클래스에 inner 키워드를 사용하면 바깥 클래스의 멤버에 접근할 수 있음

```
fun main(args: Array<String>) {  
    val ccc = AAA().BBB().bbb() //중첩 클래스처럼 AAA.BBB().bbb()로 접근하면 오류 발생  
}  
  
class AAA {  
    private val aaa: Int = 10  
  
    inner class BBB {  
        fun bbb(aaa) = 100 + aaa  
    }  
}
```

# 지도와 GPS

## ▶ 위치 정보 요청

### ▶ MapsActivity 클래스 내부에 locationInit 메소드 구현

#### ▶ LocationRequest 객체를 이용하여 위치 정보 요청에 대한 세부적인 설정

```
// 위치 정보를 얻기 위한 각종 정보 초기화
private fun locationInit() {
    fusedLocationProviderClient = FusedLocationProviderClient(this)
    //이후에 위치 정보 요청하는 함수에서 사용하므로 초기화에서 객체 생성

    locationCallback = MyLocationCallBack()
    //이후에 위치 정보 요청하는 함수에서 사용하므로 초기화에서 객체 생성

    locationRequest = LocationRequest()

    locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY
    // 업데이트 인터벌
    // 위치 정보가 없을 때는 업데이트 안 함
    // 상황에 따라 짧아질 수 있음, 정확하지 않음
    // 다른 앱에서 짧은 인터벌로 위치 정보를 요청하면 짧아질 수 있음
    locationRequest.interval = 10000
    // 정확함. 이것보다 짧은 업데이트는 하지 않음
    locationRequest.fastestInterval = 5000
}
```

# 지도와 GPS

## ▶ 위치 정보 요청

```
locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY  
locationRequest.interval = 10000  
locationRequest.fastestInterval = 5000
```

## ▶ 프로퍼티의 의미

### ▷ priority : 정확도 나타냄

- PRIORITY\_HIGH\_ACCURACY : 가장 정확한 위치 요청
- PRIORITY\_BALANCED\_POWER\_ACCURACY : '블록' 수준의 정확도 요청
- PRIORITY\_LOW\_POWER : '도시' 수준의 정확도 요청
- PRIORITY\_NO\_POWER : 추가 전력 소모 없이 최상의 정확도 요청

### ▷ interval : 위치를 갱신하는 데 필요한 시간은 밀리초 단위로 입력함

- 위 코드는 10초마다 위치정보 갱신

### ▷ fastestInterval : 다른 앱에서 위치를 갱신했을 때 그 정보를 가장 빠른 간격 (밀리초 단위) 으로 입력함

- 5초 이내에는 갱신하지 않음



# 지도와 GPS

## ▶ 위치 정보 요청

### ▶ 위치 정보를 위한 초기화 코드를 onCreate()에 입력

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_maps)  
  
    // SupportMapFragment를 가져와서 지도가 준비되면 알림을 받음  
    val mapFragment = supportFragmentManager  
        .findFragmentById(R.id.map) as SupportMapFragment  
    mapFragment.getMapAsync(this)  
  
    locationInit()  
}
```

# 지도와 GPS

## ▶ 위치 정보 요청

- ▶ 위치 정보 요청은 액티비티가 활성화되는 onResume()에서 수행하며 별도의 메서드로 처리
- ▶ MapsActivity에 현재 위치를 요청하는 코드 작성
- ▶ 퍼미션 설정을 아직 하지 않았기 때문에 코드에 빨간줄이 생성되지만 일단 무시

```
private fun addLocationListener() {  
    fusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback, null);  
}
```

## ▶ 위치 정보를 액티비티가 활성화되기 직전에 요청

- ▶ 예전 프로젝트에서 가속도 센서정보를 가져오는 타이밍과 동일

```
override fun onResume() {  
    super.onResume()  
    addLocationListener()  
}
```

# 지도와 GPS

## ▶ 위치 권한 요청

▶ addLocationListener () 메서드에 빨간 줄 에러

▷ 위치 정보 권한 요청 코드를 작성하지 않았기 때문

```
private fun addLocationListener() {  
    fusedLocationProviderClient.requestLocationUpdates(locationRequest,  
    locationCallback,  
    null);  
}
```

# 지도와 GPS

## ▶ 위치 권한 요청

### ▶ 권한요청이 필요한 이유를 설명 후 요청 확인 받는 코드

#### ▶ 다이얼로그로 설명을 하고 긍정/부정 버튼을 표시

```
private fun showPermissionInfoDialog() {  
    alert("현재 위치 정보를 얻기 위해서는 위치 권한이 필요합니다", "권한이 필요한 이유") { //다이얼로그 표시  
        yesButton {  
            // 권한 요청  
            //yes블록 내부에서는 this가 DialogInterface이므로 현재 액티비티를 명시적으로 표시  
            ActivityCompat.requestPermissions(this@MapsActivity,  
                arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),  
                REQUEST_ACCESS_FINE_LOCATION)  
        }  
        noButton { }  
    }.show()  
}
```

#### ▶ requestPermissions()

- 두번째 매개변수 : 요청하는 권한이 여러 개 일수 있으므로 배열로 인자를 처리
- 세번째 매개변수 : requestCode는 onRequestPermissionsResult()메소드에서 해당 권한이 승인되었는지 확인하기 위하여 사용

```
private val REQUEST_ACCESS_FINE_LOCATION = 1000 //클래스의 멤버로 선언
```

# 지도와 GPS

## ▶ 위치 권한 요청

### ▶ onResume()에서 권한 요청

```
override fun onResume() {  
    super.onResume()  
  
    // 권한 요청  
    permissionCheck(cancel = {  
        // 위치 정보가 필요한 이유 다이얼로그 표시  
        showPermissionInfoDialog()  
    }, ok = {  
        // 현재 위치를 주기적으로 요청 (권한이 필요한 부분)  
        addLocationListener()  
    })  
}
```

# 지도와 GPS

## ▶ 위치 권한 요청

▶ 권한 선택에 대한 처리를 할 수 있도록 `permissionCheck()`를 오버라이드

▷ 처음과 매번 권한을 승인했을 경우에는 시스템에서 권한 허용 여부 요청

▷ 예전에 권한 허용을 거부했을 경우 사용자가 직접 허용 여부에 대한 메시지를 전달

```
//이 메서드는 함수 인자 두 개를 받음. 두 함수는 모두 인자가 없고 반환 값도 없음
private fun permissionCheck(cancel: () -> Unit, ok: () -> Unit) {
    // 위치 권한이 있는지 검사
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED) {
        // 권한이 허용되지 않음
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_FINE_LOCATION)) {
            // 이전에 권한을 한 번 거부한 적이 있는 경우에 실행할 함수
            cancel()
        } else {
            // 권한 요청
            ActivityCompat.requestPermissions(this,
                arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
                REQUEST_ACCESS_FINE_LOCATION)
        }
    } else {
        // 권한을 수락 했을 때 실행할 함수
        ok()
    }
}
```

# 지도와 GPS

## ▶ addLocationListener()메소드 수정

▶ 여전히 빨간줄로 표시되어 있으므로 메소드 상단에 어노테이션 추가

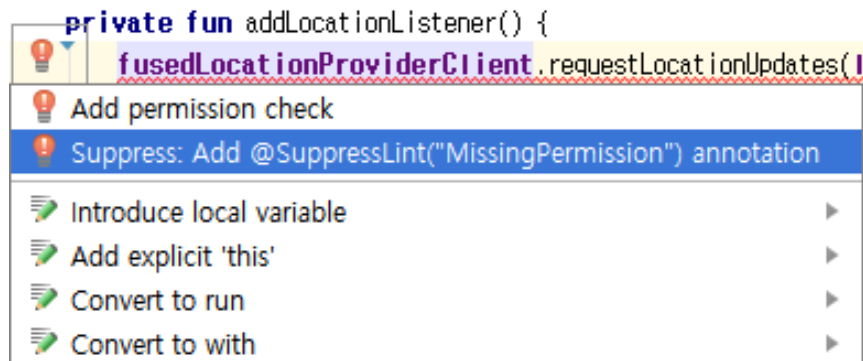
▶ @SuppressWarnings("MissingPermission")

```
private fun addLocationListener() {  
    fusedLocationProviderClient.requestLocationUpdates(locationRequest,  
        locationCallback,  
        null);  
}
```

▶ @SuppressWarnings(api)는 추후에 발생할 수 있는 잠재적인 문제를 개발자가 인지하여 warning을 제거하고 api를 사용할 수 있도록 함

▶ 개발자가 직접 입력하거나 안드로이드 스튜디오에서 선택

■ 권한 요청을 무시하는 주석 추가



# 지도와 GPS

## ▶ 권한 선택에 대한 처리

### ▶ onRequestPermissionsResult ()메서드 오버라이드

▶ 사용자가 권한 수락하면 addLocationListener() 메서드를 호출하여 위치 정보 갱신

▶ 거부하면 토스트 메시지 표시

```
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>,
grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    when (requestCode) {
        REQUEST_ACCESS_FINE_LOCATION -> {
            if ((grantResults.isNotEmpty()
                && grantResults[0] == PackageManager.PERMISSION_GRANTED)) {
                // 권한 허용됨
                addLocationListener()
            } else {
                // 권한 거부
                toast("권한 거부 됨")
            }
        }
    }
    return
}
```



# 지도와 GPS

## ▶ 위치 정보 요청 삭제

▶ onPause() 메서드에서 위치 요청 취소

▶ 위치 요청 취소하는 removeLocationListener()에서는 fusedLocationProviderClient객체의 removeLocationUpdates()에 LocationCallback 객체를 전달하여 주기적인 위치 정보 갱신 취소

```
override fun onPause() {  
    super.onPause()  
    removeLocationListener()  
}  
  
private fun removeLocationListener() {  
    // 현재 위치 요청을 삭제  
    fusedLocationProviderClient.removeLocationUpdates(locationCallback)  
}
```

## ▶ 실행하기 - 기기의 GPS정보 ON

▶ 권한 요청 및 위치 정보 요청의 추가와 삭제까지 구현했으므로 권한을 요청했을 때 현재 위치로 지도가 이동하면 성공

▶ 정확한 확인을 위하여 MyLocationCallBack 클래스의 맨 아래에 코드 추가  
Log.d("MapsActivity", "위도: \$latitude, 경도: \$longitude")

# 지도와 GPS

## ▶ 이동 자취를 선으로 그리기

### ▶ 구글 지도는 이동 자취를 그리는 다양한 메서드 제공

▷ addPolyLine() : 선의 집합으로 지도에 경로와 노선을 표시

▷ addCircle() : 원을 표시

▷ addPolygon() : 영역을 표시

### ▶ 구현 순서

1. 이동 경로 그리기
2. 화면 유지하기
3. 에뮬레이터에서 테스트하기

# 지도와 GPS

## ▶ 아래 코드를 추가

### ▶ 멤버 추가

- ▶ polylineOptions()로 객체를 생성하고 선을 이루는 좌표들과 선의 굵기, 색상 등을 설정, 여기서는 굵기 5f, 색상 빨강으로 설정

```
private val polylineOptions = PolylineOptions().width(5f).color(Color.RED)
```

### ▶ MyLocationCallBack 클래스에 선그리기

- ▶ 위치 정보가 갱신되면 해당 좌표를 polylineOptions 객체에 추가
- ▶ 지도에 polylineOptions 객체를 추가

```
location?.run {  
    val latLng = LatLng(latitude, longitude)  
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 17f))  
    Log.d("MapsActivity", "위도: $latitude, 경도: $longitude")  
    //polyline에 좌표(객체) 추가  
    polylineOptions.add(latLng)  
    //선 그리기  
    mMap.addPolyline(polylineOptions)  
}
```

### ▶ 실행 확인

# 지도와 GPS

## ▶ 화면 유지하기

- ▶ 지도를 테스트할 때 화면이 돌아가거나 자동으로 꺼지면 테스트하기 어려우므로 화면을 고정하고 꺼지지 않도록 설정

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    // 화면이 꺼지지 않게 하기  
    window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)  
    // 세로 모드로 화면 고정  
    requestedOrientation = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT  
    setContentView(R.layout.activity_maps)  
  
    // SupportMapFragment를 가져와서 지도가 준비되면 알림을 받습니다 2  
    val mapFragment = supportFragmentManager  
        .findFragmentById(R.id.map) as SupportMapFragment  
    mapFragment.getMapAsync(this)  
  
    locationInit()  
}
```

# 지도와 GPS

## ▶ 에뮬레이터에서 테스트하기

- ▶ 위치 정보를 테스트하는 데 에뮬레이터를 사용
- ▶ 에뮬레이터를 실행하고 에뮬레이터 메뉴에서 Extended controls 화면을 열기
- ▶ Location 탭을 클릭하면 GPS를 가상으로 테스트하는 화면 나옴
- ▶ 오픈 스트리트 맵에서 공개된 GPS 이동 경로 내려 받기
  - ▷ <https://www.openstreetmap.org/>
  - ▷ 접속하여 GPS 궤적을 클릭

GPS 궤적 사용자 일기 저작권 도움말



# 지도와 GPS

## ▶ 에뮬레이터에서 테스트 하기

### ▶ 공개 GPS 궤적 목록 표시

### ▶ 점 개수가 많을수록 용량이 커지므로 점 개수가 1,000개 이하인 적당한 파일을 내려 받음

#### ▷ 에뮬레이터에서 사용하려면 gpx확장자 파일이 좋음

### ▶ 테스트 화면에서 [LOAD GPX/KML]을 클릭하여 내려 받은 파일을 선택하면 정보 데이터가 표시

### ▶ 플레이를 클릭하면 좌표를 순서대로 에뮬레이터에 전송

### ▶ 좌표 간격을 빠르게 하려면 [speed]를 클릭하여 속도를 5배까지 조절 가능

---

**Q & A**

---