**모바일 앱 프로그래밍 1팀**

2016114257 김대영

2018114617 박경모

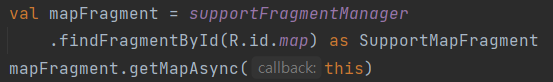
2021110290 신성현

2018110834 채종민

**야운동(야외 운동 기구 위치 찾기) app 설명**

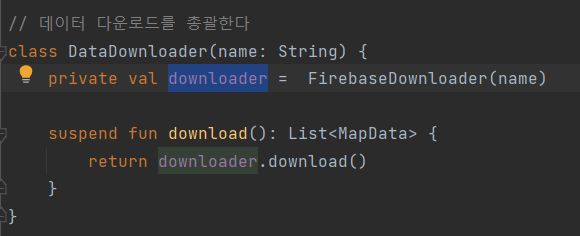
**화면 표시**

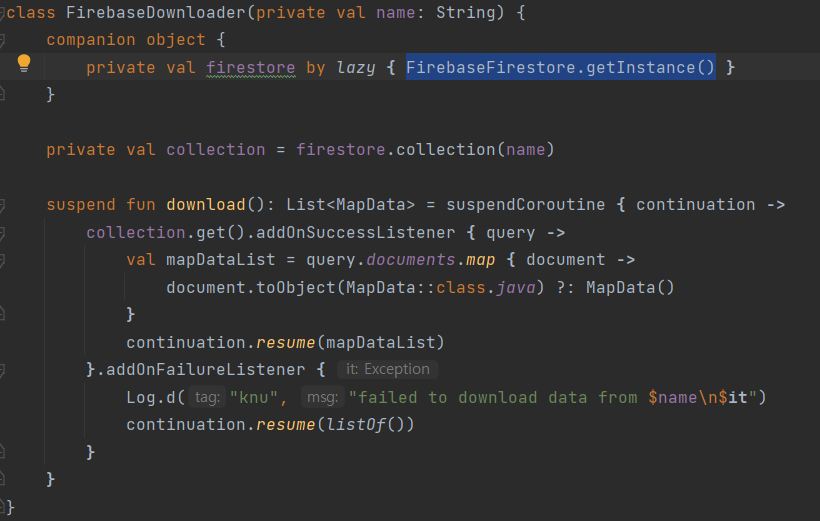
모듈 생성 시 맵 액티비티로 설정하여 구현 하였다.

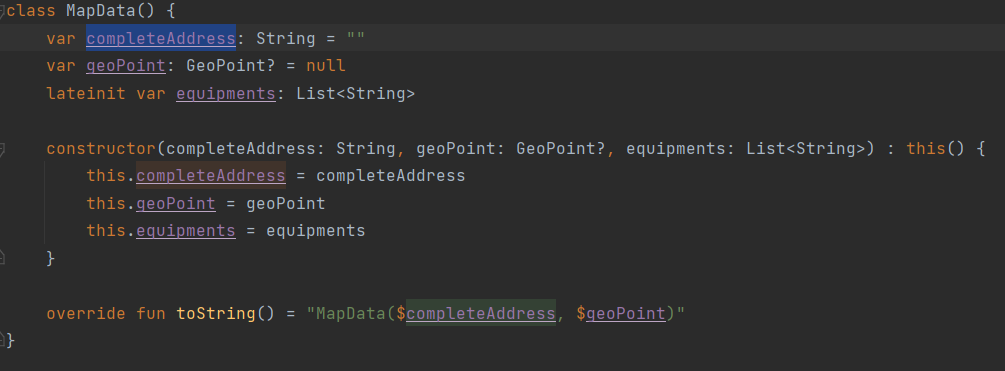


**파이어베이스로부터 데이터 받기**

DataDownloader 클래스의 download() 함수를 통해 원하는 Firebase의 collection 데이터들을 MapData 클래스 형식으로 MapActivity의 onMapReady()에 넘겨준다. 그 후, FirebaseDownloadr 클래스에서 FirebaseFirestore.getInstance()를 통해 파이어베이스 객체를 생성하고, FirebaseDownloadr 클래스의 download() 함수를 통해 원하는 collection의 데이터를 MapData 클래스 형식으로 firebase로부터 받아온다. 마지막으로 MapData 클래스 형식으로 DataDownloader 클래스로 반환한다.

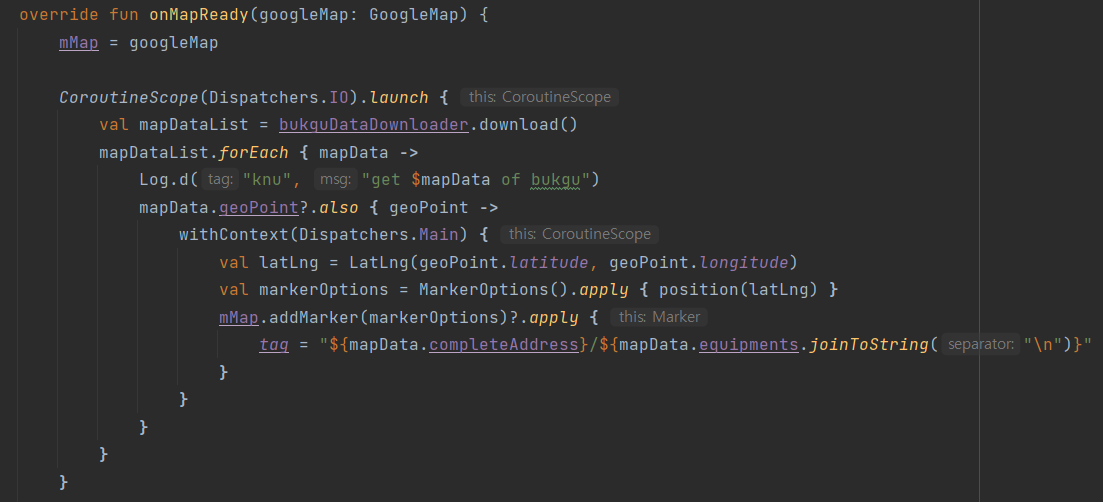






**파이어 베이스에서 받은 정보 지도 화면에 표시**

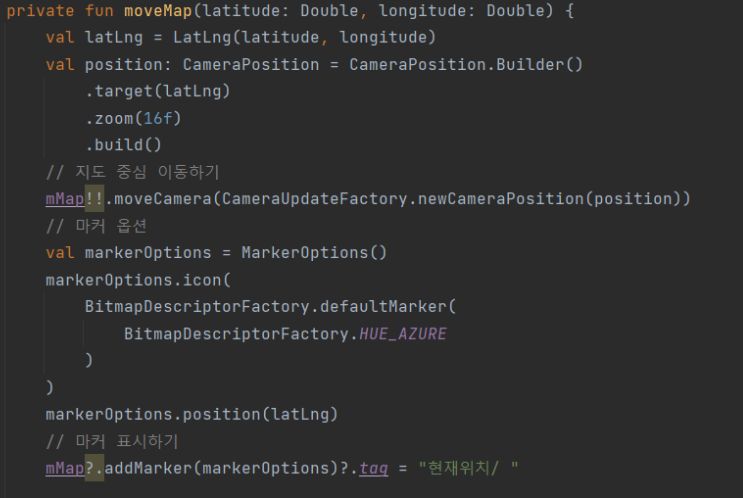
firebase로 부터 받은 MapData 형식의 데이터에서 위도 경도 정보를 받는다. 그 후 MarkerOptions의 position()를 통해 지도에 마커 형태로 표시한다. 또한 addMarker를 통해 firebase로 부터 받은 추가 정보(도로명 주소와 야외 운동 기구 종류 )들을 담는다. 이 때 CoroutineScope의 withContext 안에서 이러한 작업을 하는 이유는 메인 함수 밖의 코루틴 작업 시, 메인 함수의 UI 작업을 하지 못하기 때문에 마커를 화면의 지도에 띄울수 없기 때문이다.



북구 데이터 받는 예시 코드

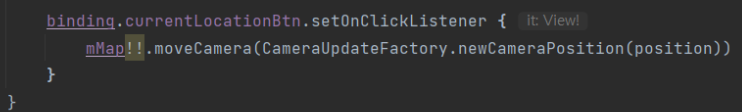
**애플리케이션 실행 시 현재 위치 표시**

onConnected() 함수에서 getLastLocation()를 통해 현재 위도와 경도를 받는다. moveMap() 함수에서 CameraPosition을 통해 받은 위치 정보와 지도 화면 표시에 표시될 배율을 정하고 이를 moveCamera()에 넣어 주면 애플리케이션 실행 시 현재 위치가 화면에 표시된다.



**애플리케이션 사용 중 버튼 클릭 시 현재 위치로 이동**

moveMap() 함수의 binding.currentLocationBtn.setOnClickListener와 위에서 받은 현재 위치 정보를 이용하여 구현 하였다.



**구글 지도와 연동**

MarkerOptions()의 addMarker를 통해 마커 생성 시 마커를 클릭하면 오른쪽 상단에 구글 지도와 연동할 수 있는 아이콘이 생성 된다.

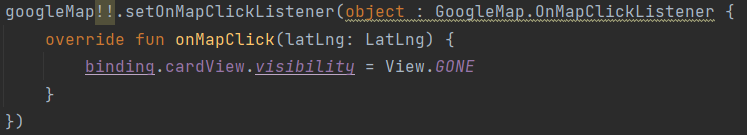
**위치 마커 클릭 시 상세 정보 표시**

setOnMarkerClickListener의 onMarkerClick를 통해 cardView를 화면에 보이게 설정하고, marker.tag에 저장되었던 추가 정보를 cardView에 적는다.

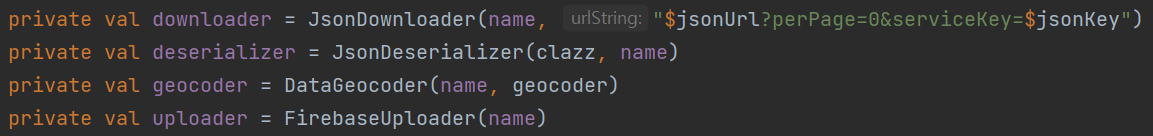


**지도 화면 클릭 시 상세 정보 사라짐**

setOnMapClickListenerd의 onMapClick를 통해 cardView를 안보이게 한다.

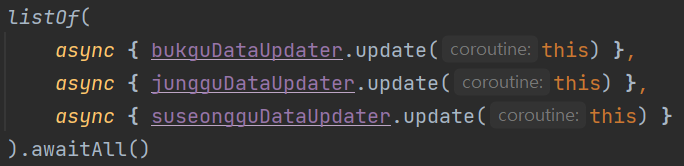


**야운동 서버 app 동작 설명**



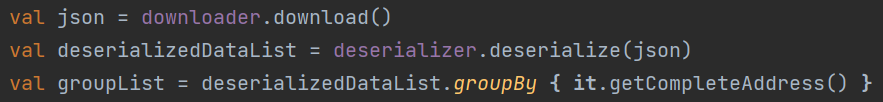
**초기화**

1. 업데이트를 처리하는 DataUpdater 클래스의 객체를 생성한다.
2. DataUpdater는 하위 클래스들의 객체를 포함하고 있으며 초기화될 때 각 객체들을 생성해서 초기화한다.
   * JsonDownloader, JsonDeserializer, DataGeocoder, FirebaseUploader…



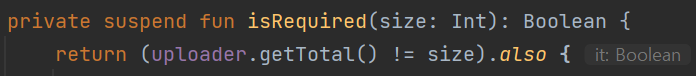
**업데이트 시작**

1. **launch of CoroutineScope**: 업데이트가 오래 걸리기 때문에 main thread에서는 진행할 수가 없다. 업데이트를 비동기적으로 수행하기 위해서 coroutine을 사용한다.
2. **async, awaitAll() of CoroutineScope**: 각 지역이 서로 순서대로 업데이트될 필요는 없으므로 동시에 병렬적으로 처리한다.



**공공데이터 다운로드**

1. **download() of JsonDownloader**: 공공데이터포털 API를 URL로 접근해서 필요한 데이터를 JSON string으로 받아온다.
2. **deserialize() of JsonDeserializer**: JSON string을 Deserialize(역직렬화)한다. 그러면 이후부터는 DeserializedData class의 객체의 list로 사용할 수 있다.
3. **groupBy() of deserializedDataList**: 주소를 키 값으로 해서 DeserializedData를 group-by한다.



**업데이트 확인**

1. **isRequired() of DataUpdater**: 업데이트가 오래 걸리기 때문에 groupData의 개수와 파이어베이스에 저장된 데이터의 개수가 다를 경우에만 진행한다.
2. **getTotal() of FirebaseUploader**: 파이어베이스에 저장된 데이터의 개수를 확인한다. 이 작업은 기본적으로 비동기적으로 수행되므로 동기화를 하기 위해 suspendCoroutine을 사용한다.



**데이터 가공**

1. **geocode() of DataGeocoder**: groupData의 주소를 Android에서 제공하는 Geocoder로 보내서 위도, 경도를 구한다. 이 작업이 가장 오래 걸린다.
2. **map() of groupList**: 필요한 정보만 뽑아서 MapData라는 새로운 class 객체의 list로 변환한다.



**파이어베이스 업로드**

1. **upload() in FirebaseUploade**r: MapData를 파이어베이스로 보낸다. 이 작업은 기본적으로 비동기적으로 수행되므로 동기화를 하기 위해 suspendCoroutine을 사용한다.

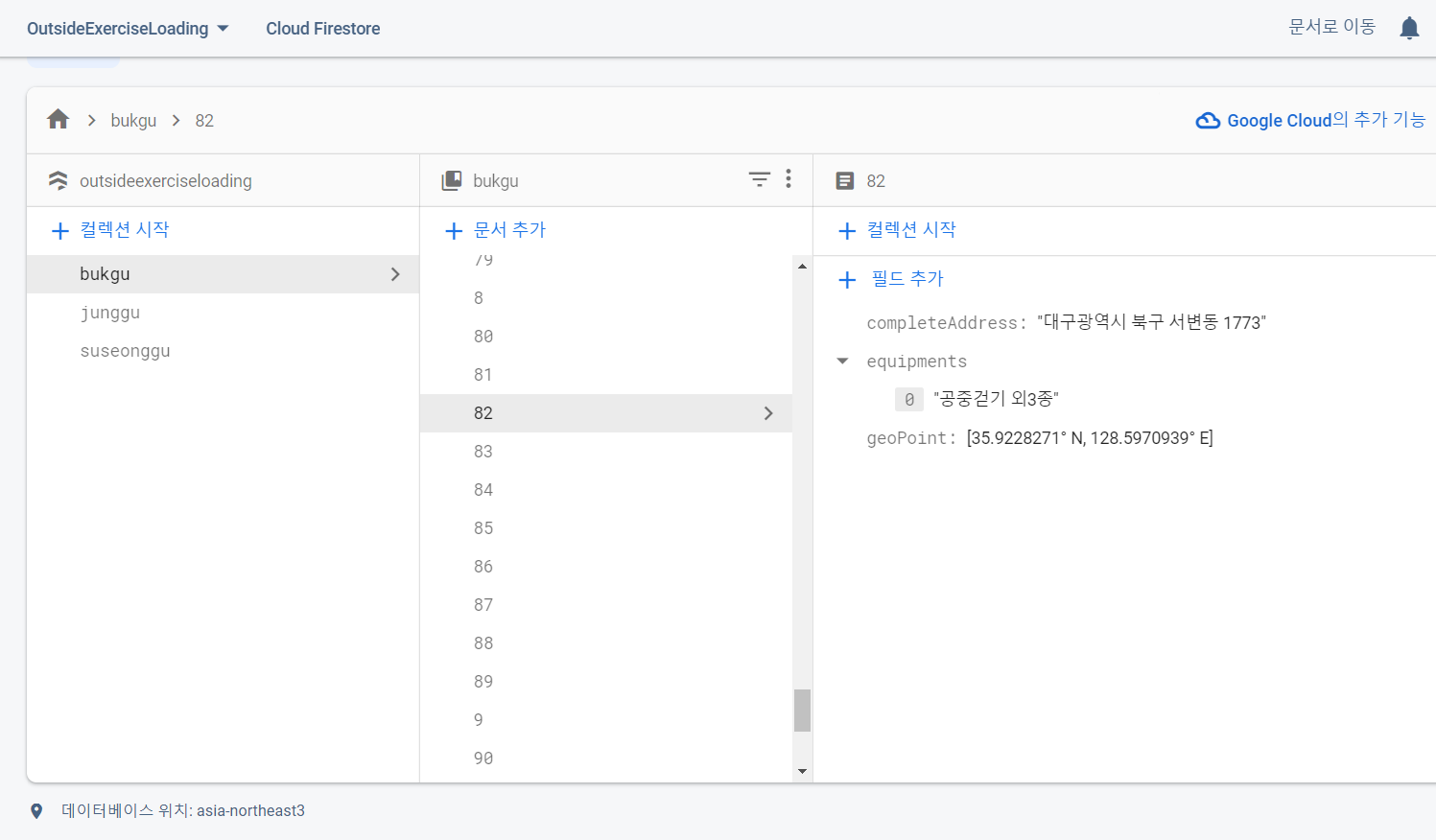
**Firebase**

**데이터**

**컬렉션** - bukgu, junggu, suseonggu

**데이터 형식**

1. completeAddress : String (도로명 주소), equipments : List<String> (운동 기구 종류), geoPoint : GeoPoint (위도 경도)
2. toal : Int (컬렉션 별 총 데이터 개수)



**사용 스택**

안드로이드 스튜디오, 인텔리제이, 코틀린, 파이어베이스, 피그마