

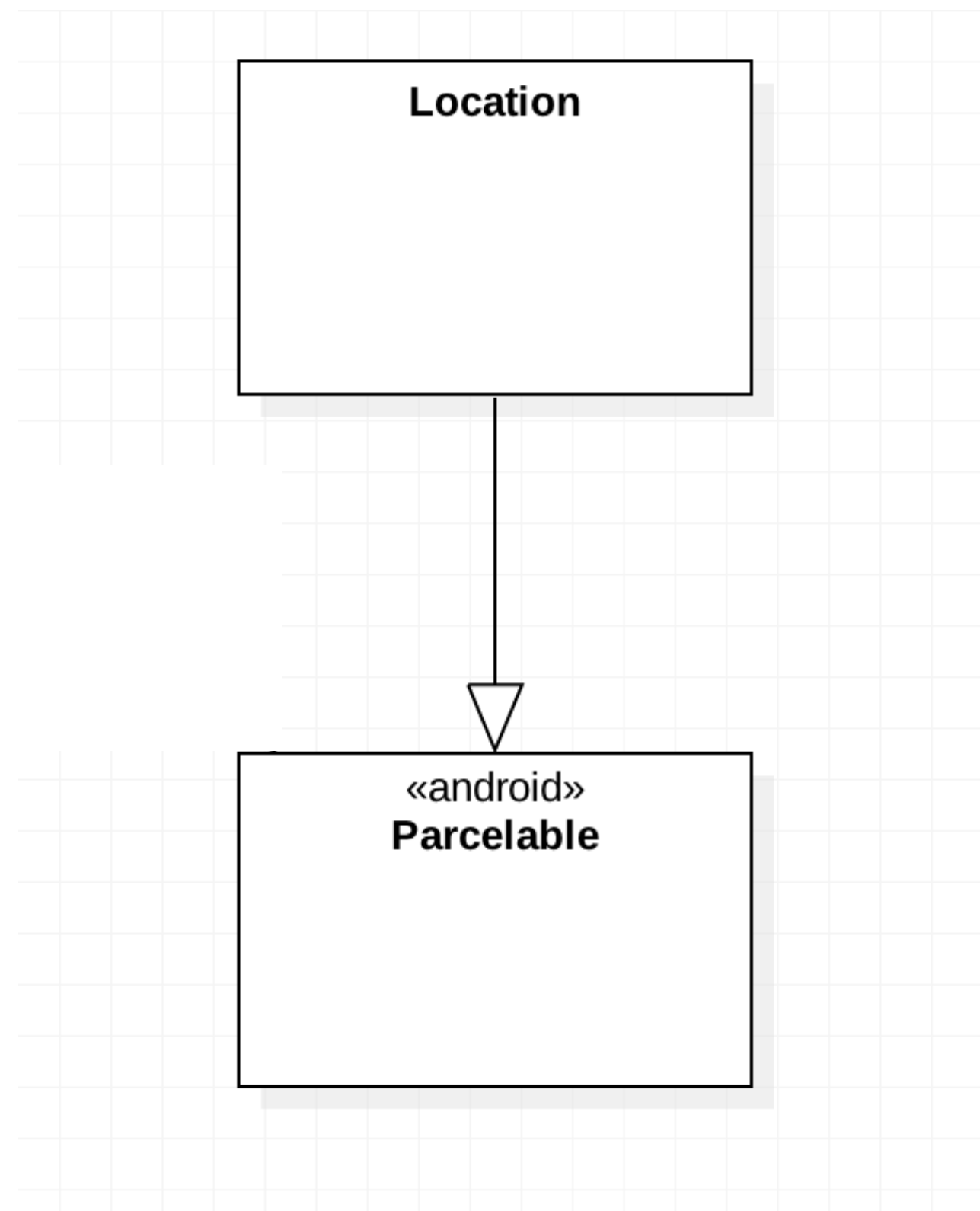
Map Markers

Cameras &
Markers



These abstractions enable
the map view to be
manipulated
programmatically, and
facilitate direct
manipulation by the user.

PlacemarkModel



```
@Parcelize
data class Location(var lat: Double = 0.0,
                    var lng: Double = 0.0,
                    var zoom: Float = 0f) : Parcelable
```


New Model to represent a location + zoom level

Placemark CANCEL

Placemark Title

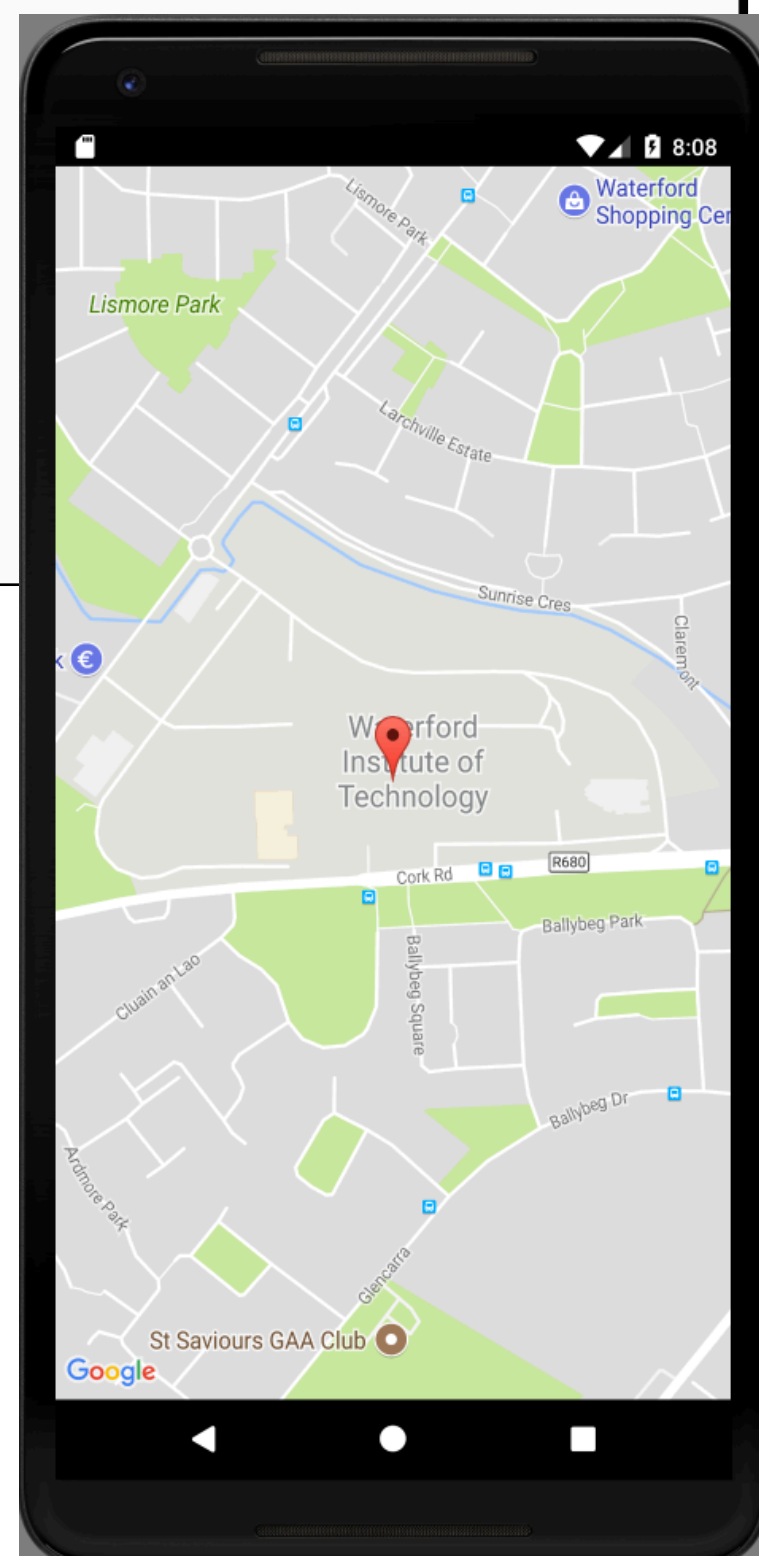
Description

ADD IMAGE



SET LOCATION

ADD PLACEMARK



```
placemarkLocation.setOnClickListener {
    val location = Location(52.245696, -7.139102, 15f)
    startActivity (intentFor<MapsActivity>().putExtra("location", location))
}
```

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

    private lateinit var mMap: GoogleMap
    var location = Location()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        location = intent.extras.getParcelable<Location>("location")
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap
        val loc = LatLng(location.lat, location.lng)
        mMap.addMarker(MarkerOptions().position(loc).title("Default Marker"))
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, location.zoom))
    }
}
```

GoogleMap

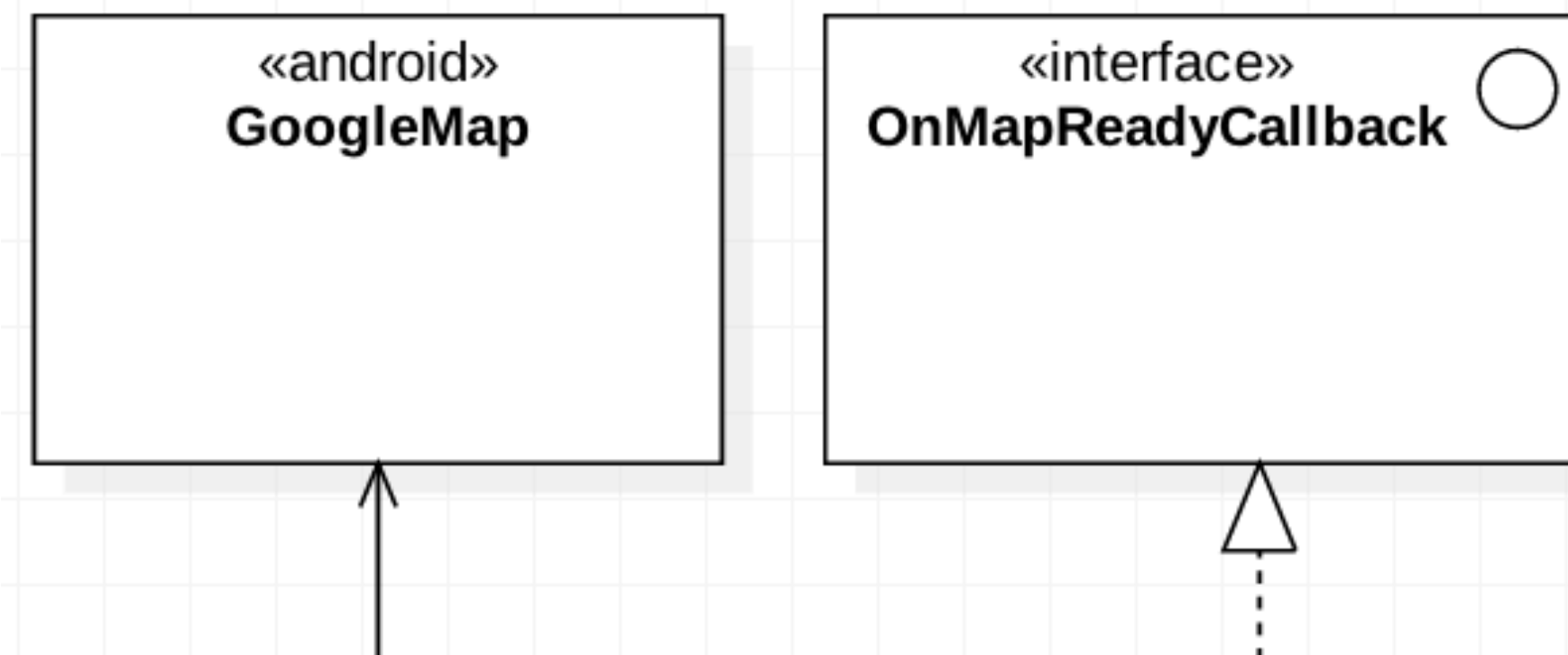
```
private lateinit var map: GoogleMap
```

public final class **GoogleMap** extends **Object**

This is the main class of the Google Maps Android API and is the entry point for all methods related to the map. You cannot instantiate a **GoogleMap** object directly, rather, you must obtain one from the `getMapAsync()` method on a **MapFragment** or **MapView** that you have added to your application.

Note: Similar to a **View** object, a **GoogleMap** can only be read and modified from the Android UI thread. Calling **GoogleMap** methods from another thread will result in an exception.

You can adjust the viewpoint of a map by changing the position of the camera (as opposed to moving the map). You can use the map's camera to set parameters such as location, zoom level, tilt angle, and bearing. For more information, see [Camera and View](#).



OnMapReadyCallback

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

Contents

Public Method Summary

Public Methods

public interface **OnMapReadyCallback**

Callback interface for when the map is ready to be used.

Once an instance of this interface is set on a `MapFragment` or `MapView` object, the `onMapReady(GoogleMap)` method is triggered when the map is ready to be used and provides a non-null instance of `GoogleMap`.

If Google Play services is not installed on the device, the user will be prompted to install it, and the `onMapReady(GoogleMap)` method will only be triggered when the user has installed it and returned to the app.

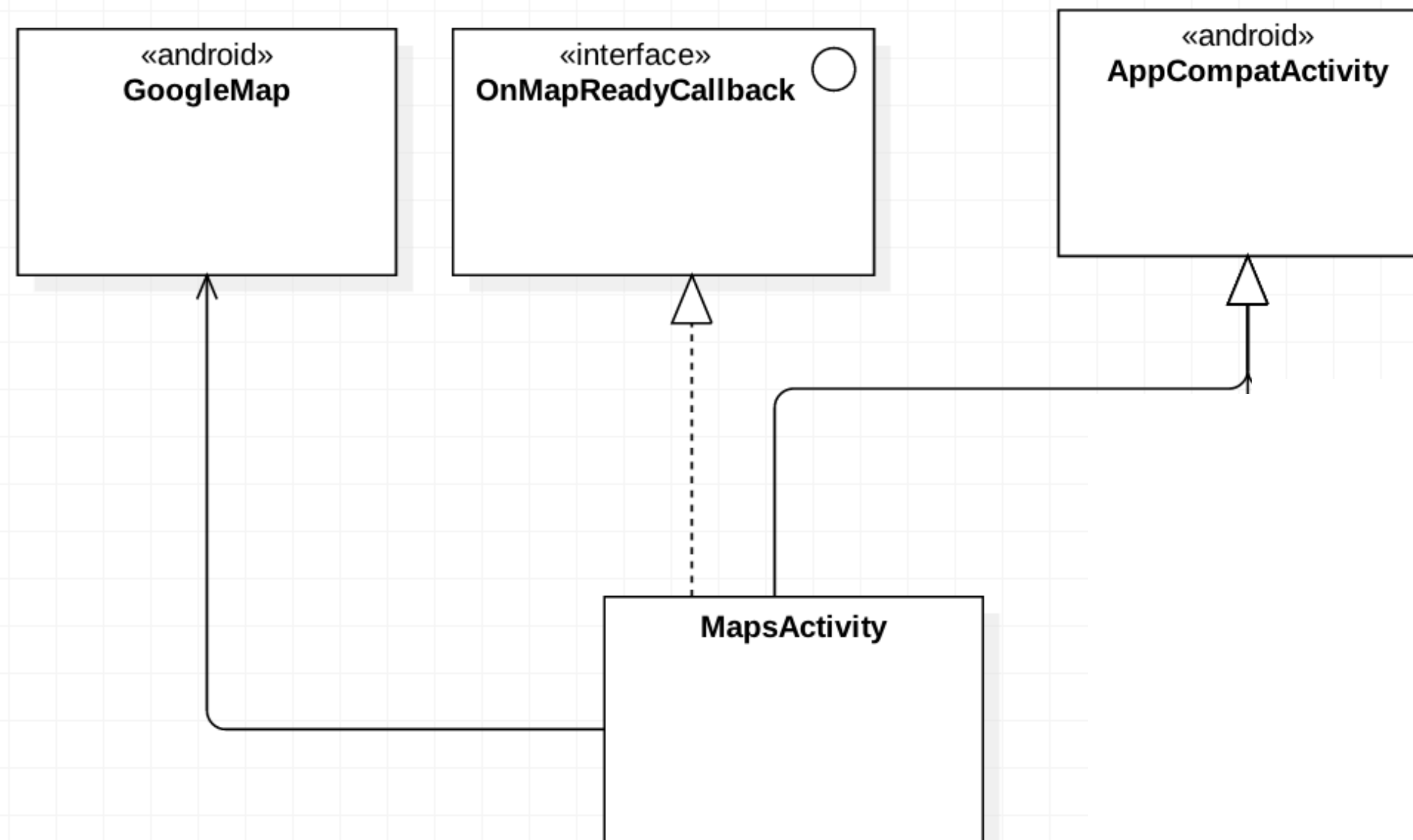
Public Method Summary

```
abstract void onMapReady(GoogleMap googleMap)  
Called when the map is ready to be used.
```



MapsActivity Essentials

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {  
    private lateinit var map: GoogleMap  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_maps)  
        val mapFragment = supportFragmentManager  
            .findFragmentById(R.id.map) as SupportMapFragment  
        mapFragment.getMapAsync(this)  
    }  
    override fun onMapReady(googleMap: GoogleMap) {  
        map = googleMap  
    }  
}
```



Marker

An icon placed at a particular point on the map's surface. A marker icon is drawn oriented against the device's screen rather than the map's surface; i.e., it will not necessarily change orientation due to map rotations, tilting, or zooming.

A marker has the following properties:

Alpha

Sets the opacity of the marker. Defaults to 1.0.

Anchor

The point on the image that will be placed at the `LatLng` position of the marker. This defaults to 50% from the left of the image and at the bottom of the image.

Position

The `LatLng` value for the marker's position on the map. You can change this value at any time if you want to move the marker.

Title

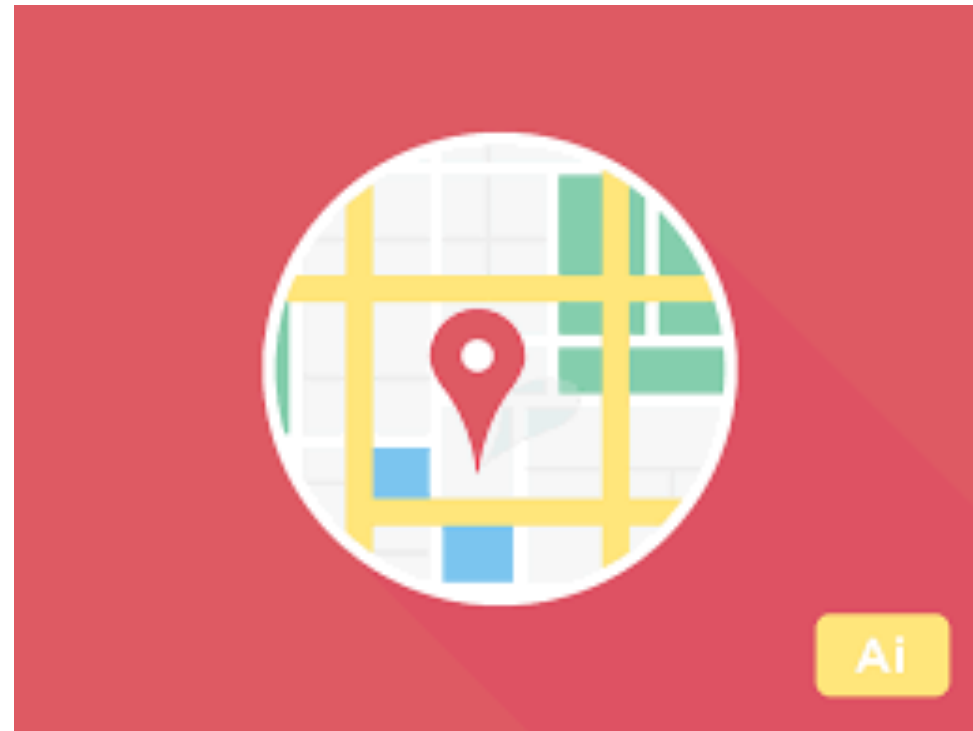
A text string that's displayed in an info window when the user taps the marker. You can change this value at any time.

Snippet

Additional text that's displayed below the title. You can change this value at any time.



Marker



Icon

A bitmap that's displayed for the marker. If the icon is left unset, a default icon is displayed. You can specify an alternative coloring of the default icon using `defaultMarker(float)`.

Drag Status

If you want to allow the user to drag the marker, set this property to `true`. You can change this value at any time. The default is `false`.

Visibility

By default, the marker is visible. To make the marker invisible, set this property to `false`. You can change this value at any time.

Flat or Billboard

If the marker is flat against the map, it will remain stuck to the map as the camera rotates and tilts but will still remain the same size as the camera zooms, unlike a `GroundOverlay`. If the marker is a billboard, it will always be drawn facing the camera and will rotate and tilt with the camera. The default is a billboard (`false`)

Rotation

The rotation of the marker in degrees clockwise about the marker's anchor point. The axis of rotation is perpendicular to the marker. A rotation of 0 corresponds to the default position of the marker. When the marker is flat on the map, the default position is North aligned and the rotation is such that the marker always remains flat on the map. When the marker is a billboard, the default position is pointing up and the rotation is such that the marker is always facing the camera. The default value is 0.

zIndex

The draw order for the marker. The markers are drawn in order of the `zIndex`, with the highest `zIndex` marker drawn on top. By setting the `zIndex` property for each marker, you can control which tap target your user is most likely to hit. The default value is 0.

Tag

An `Object` associated with the marker. For example, the `Object` can contain data about what the marker represents. This is easier than storing a separate `Map<Marker, Object>`. As another example, you can associate a `String` ID corresponding to the ID from a data set. Google Maps Android API neither reads nor writes this property.

MarkerOptions



MarkerOptions	<code>alpha</code> (float alpha) Sets the alpha (opacity) of the marker.
MarkerOptions	<code>anchor</code> (float u, float v) Specifies the anchor to be at a particular point in the marker image.
MarkerOptions	<code>draggable</code> (boolean draggable) Sets the draggability for the marker.
MarkerOptions	<code>flat</code> (boolean flat) Sets whether this marker should be flat against the map true or a billboard facing the camera false .
MarkerOptions	<code>position</code> (LatLng latlng) Sets the location for the marker.
MarkerOptions	<code>rotation</code> (float rotation) Sets the rotation of the marker in degrees clockwise about the marker's anchor point.
MarkerOptions	<code>snippet</code> (String snippet) Sets the snippet for the marker.
MarkerOptions	<code>title</code> (String title) Sets the title for the marker.
MarkerOptions	<code>visible</code> (boolean visible) Sets the visibility for the marker.
void	<code>writeToParcel</code> (Parcel out, int flags)
MarkerOptions	<code>zIndex</code> (float zIndex) Sets the zIndex for the marker.

LatLng

Field Summary

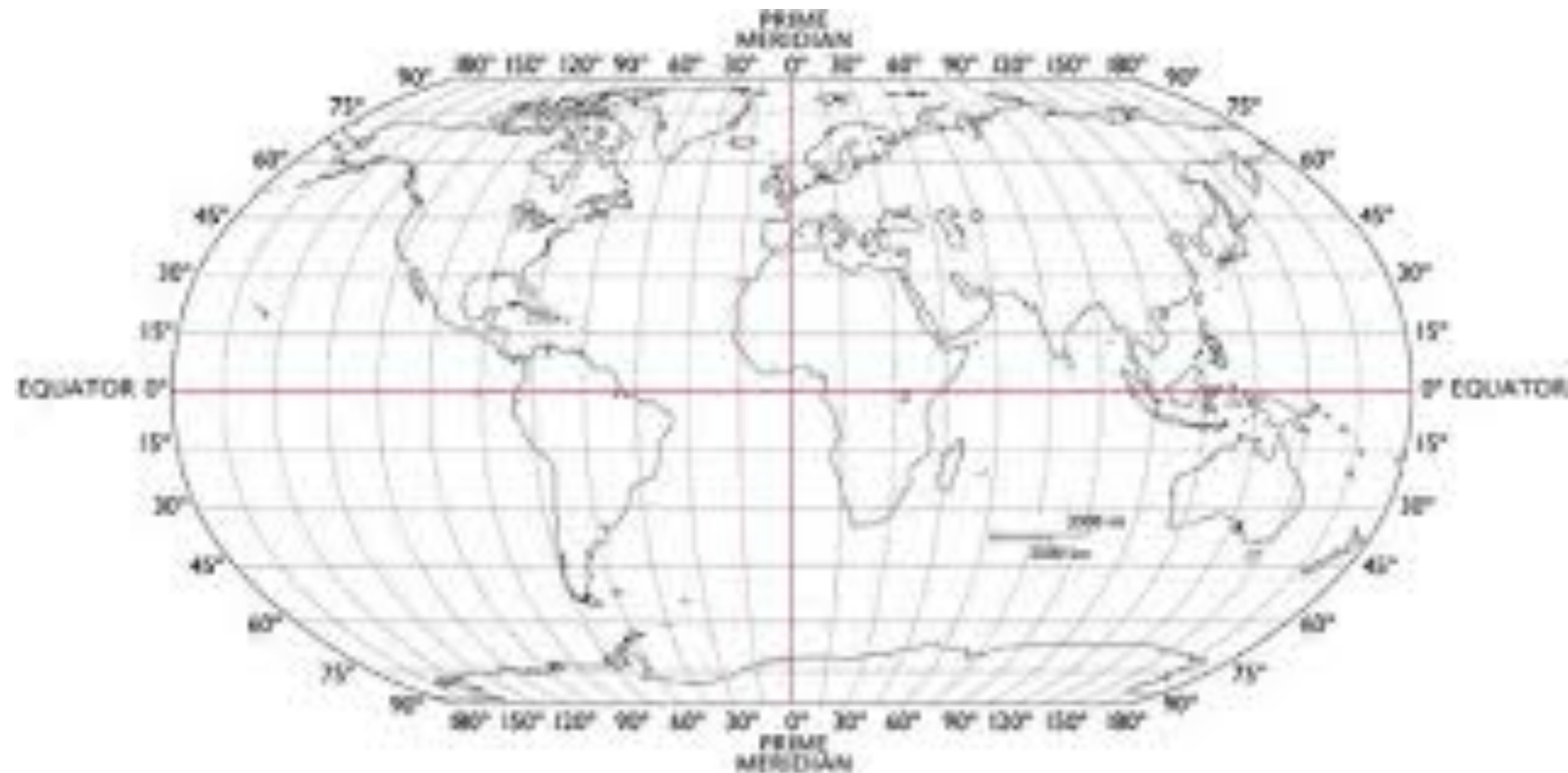
public final double [latitude](#) Latitude, in degrees.

public final double [longitude](#) Longitude, in degrees.

Public Constructor Summary

[LatLng](#)(double latitude, double longitude)

Constructs a LatLng with the given latitude and longitude, measured in degrees.



CameraUpdateFactory

public final class **CameraUpdateFactory** extends [Object](#)

A class containing methods for creating [CameraUpdate](#) objects that change a map's camera. To modify the map's camera, call [animateCamera\(CameraUpdate\)](#), [animateCamera\(CameraUpdate, GoogleMap.CancelableCallback\)](#) or [moveCamera\(CameraUpdate\)](#), using a [CameraUpdate](#) object created with this class.

For example, to zoom in on a map, you can use the following code:

```
GoogleMap map = ...;  
map.animateCamera(CameraUpdateFactory.zoomIn());
```

Prior to using any methods from this class, you must do one of the following to ensure that this class is initialized:

- Wait for a [GoogleMap](#) to become available from a [MapFragment](#) or [MapView](#) that you have added to your application. You can obtain the [GoogleMap](#) instance by calling [getMapAsync\(\)](#) and waiting for the [onMapReady\(GoogleMap map\)](#) callback.
- Call [initialize\(Context\)](#). As long as a [GooglePlayServicesNotAvailableException](#) isn't thrown, this class will be correctly initialized.



CameraUpdateFactory

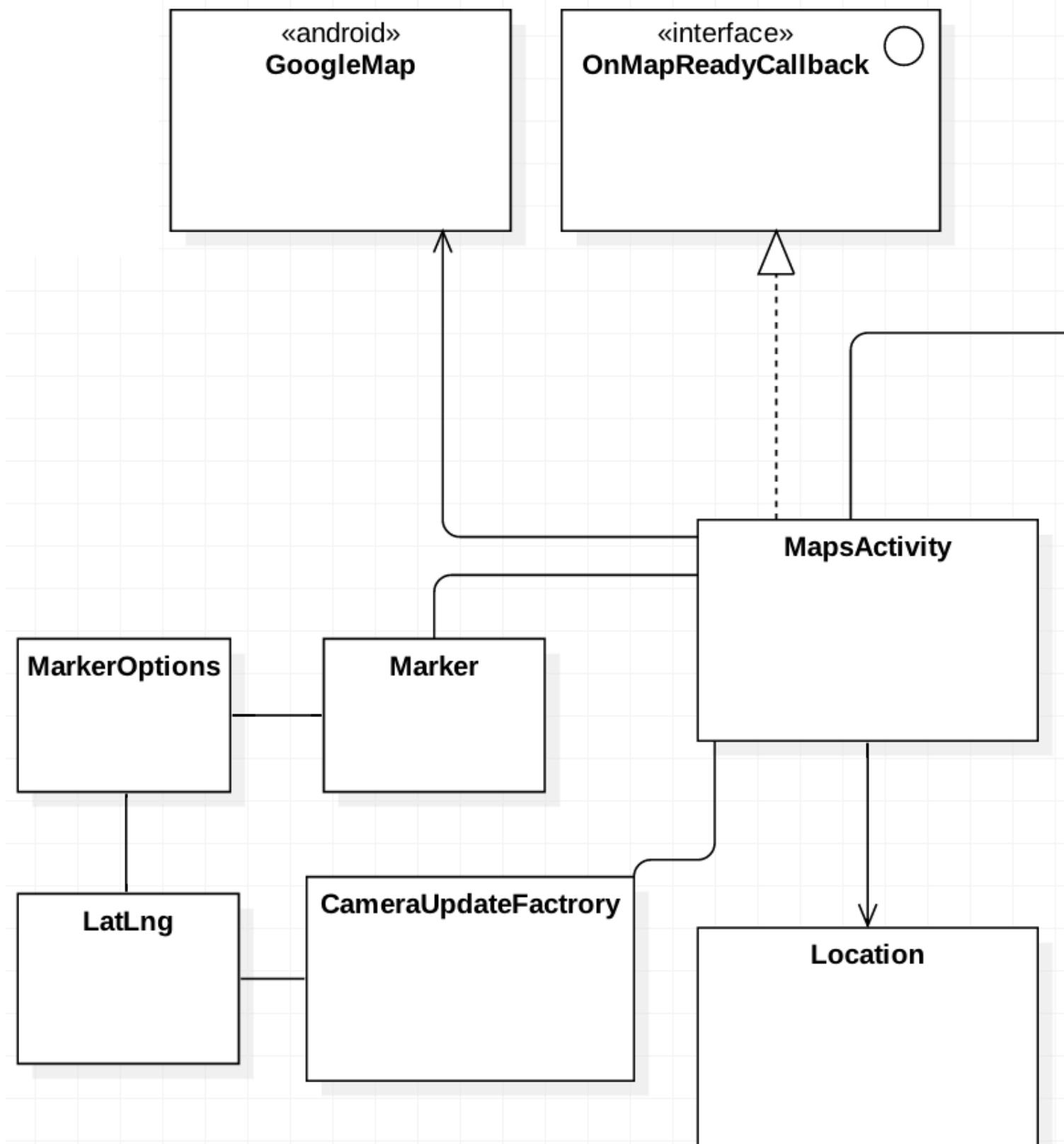


static CameraUpdate	newCameraPosition (CameraPosition cameraPosition) Returns a CameraUpdate that moves the camera to a specified CameraPosition .
static CameraUpdate	newLatLng (LatLng latLng) Returns a CameraUpdate that moves the center of the screen to a latitude and longitude specified by a LatLng object.
static CameraUpdate	newLatLngBounds (LatLngBounds bounds, int padding) Returns a CameraUpdate that transforms the camera such that the specified latitude/longitude bounds are centered on screen at the greatest possible zoom level.
static CameraUpdate	newLatLngBounds (LatLngBounds bounds, int width, int height, int padding) Returns a CameraUpdate that transforms the camera such that the specified latitude/longitude bounds are centered on screen within a bounding box of specified dimensions at the greatest possible zoom level.
static CameraUpdate	newLatLngZoom (LatLng latLng, float zoom) Returns a CameraUpdate that moves the center of the screen to a latitude and longitude specified by a LatLng object, and moves to the given zoom level.
static CameraUpdate	scrollBy (float xPixel, float yPixel) Returns a CameraUpdate that scrolls the camera over the map, shifting the center of view by the specified number of pixels in the x and y directions.
static CameraUpdate	zoomBy (float amount, Point focus) Returns a CameraUpdate that shifts the zoom level of the current camera viewpoint.
static CameraUpdate	zoomBy (float amount) Returns a CameraUpdate that shifts the zoom level of the current camera viewpoint.
static CameraUpdate	zoomIn () Returns a CameraUpdate that zooms in on the map by moving the viewpoint's height closer to the Earth's surface.
static CameraUpdate	zoomOut () Returns a CameraUpdate that zooms out on the map by moving the viewpoint's height farther away from the Earth's surface.
static CameraUpdate	zoomTo (float zoom) Returns a CameraUpdate that moves the camera viewpoint to a particular zoom level.

```

override fun onMapReady(googleMap: GoogleMap) {
    map = googleMap
    val loc = LatLng(location.lat, location.lng)
    val options = MarkerOptions()
        .title("Placemark")
        .snippet("GPS : " + loc.toString())
        .draggable(true)
        .position(loc)
    map.addMarker(options)
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, location.zoom))
}

```




Recover Location from Activity

Placemark CANCEL

Placemark Title

Description

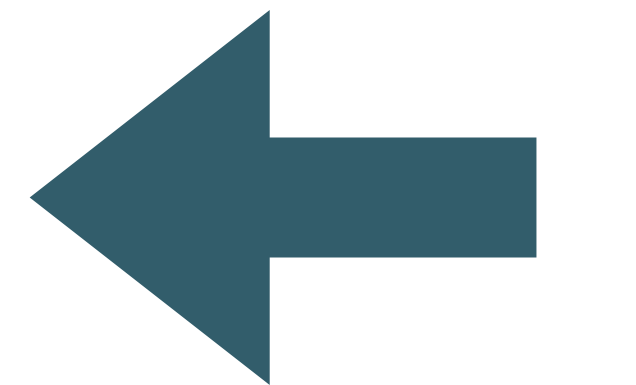
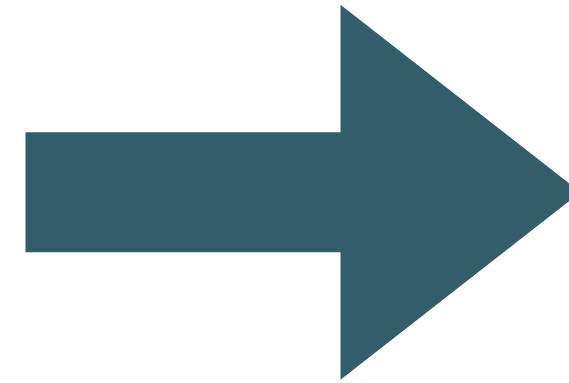
ADD IMAGE



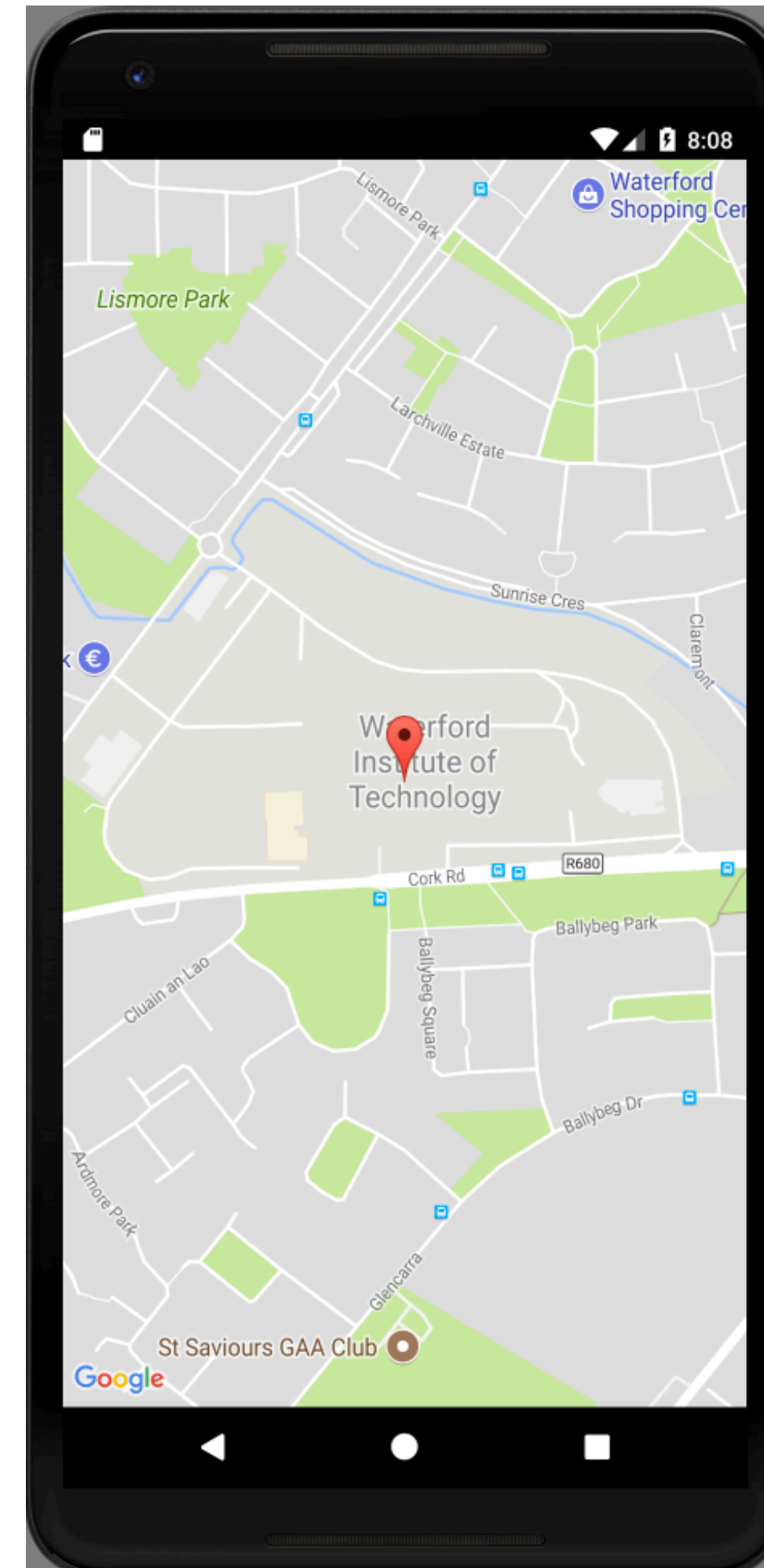
SET LOCATION

ADD PLACEMARK

Initial
Location



Changed
Location if
user
dragged
Marker



PlacemarkActivity

Placemark CANCEL

Placemark Title

Description

ADD IMAGE

SET LOCATION

ADD PLACEMARK

Initial Location + Request Code

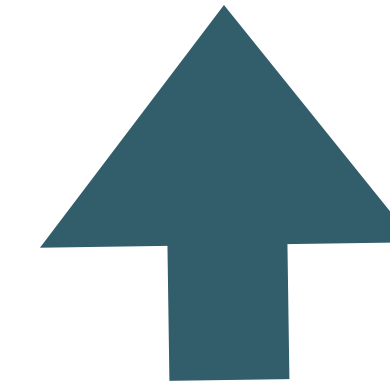
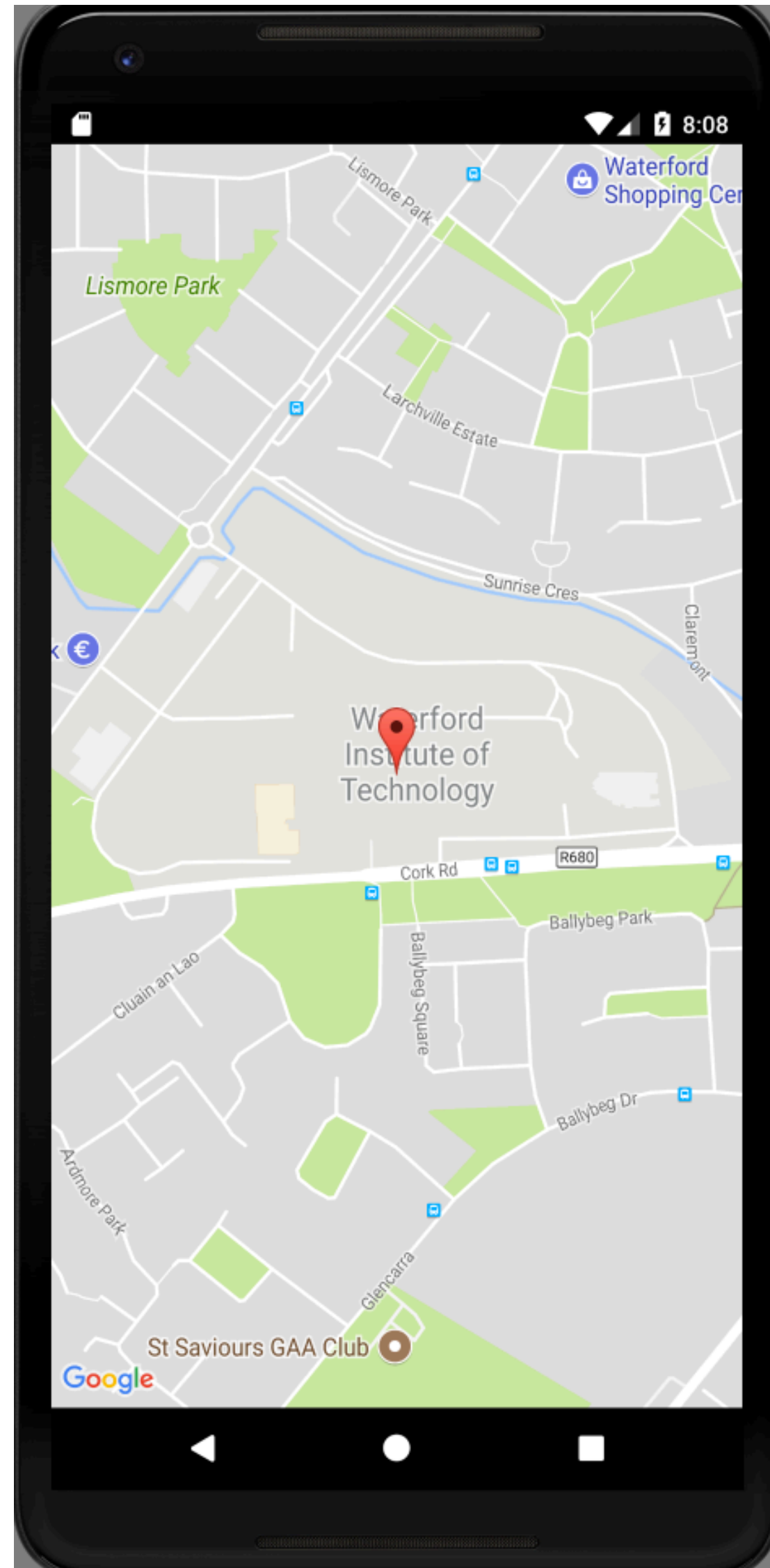
```
val LOCATION_REQUEST = 2  
var location = Location(52.245696, -7.139102, 15f)
```

Send location to MapsActivity + await result

```
placemarkLocation.setOnClickListener {  
    startActivityForResult(intentFor<MapsActivity>().putExtra("location", location), LOCATION_REQUEST)  
}
```

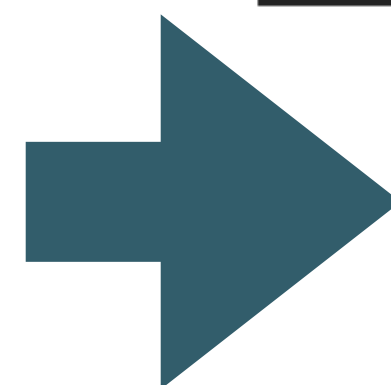
MapsActivity

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback, GoogleMap.OnMarkerDragListener {
```



Implement listener for marker drag events

```
override fun onMarkerDragStart(marker: Marker) {  
}  
  
override fun onMarkerDrag(marker: Marker) {  
}  
  
override fun onMarkerDragEnd(marker: Marker) {  
    location.lat = marker.position.latitude  
    location.lng = marker.position.longitude  
    location.zoom = map.cameraPosition.zoom  
}
```

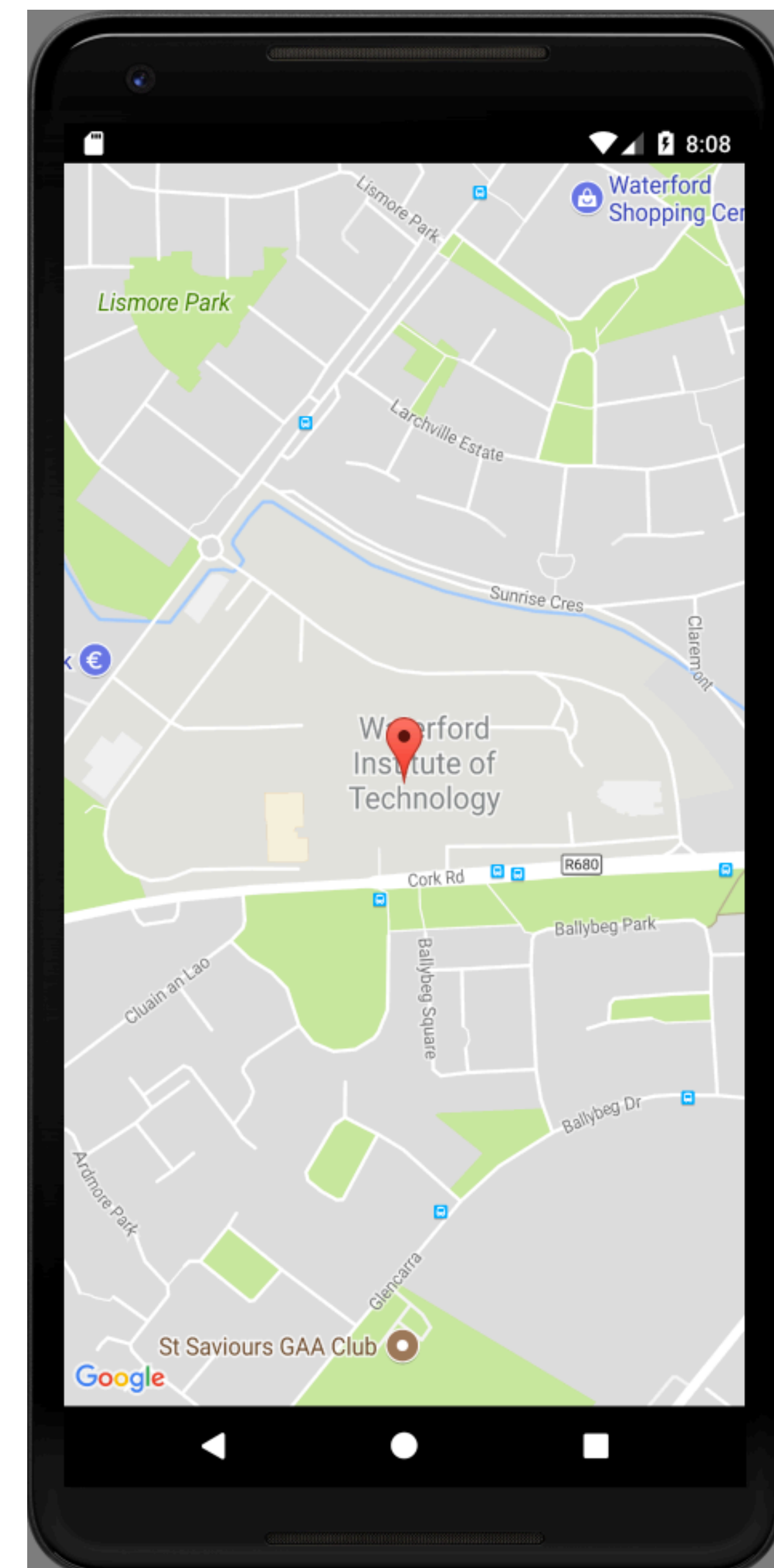
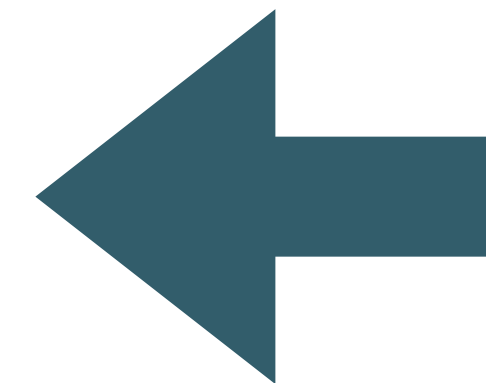


Store new location + zoom level in location object

MapsActivity

Send location object back to
parent view

```
override fun onBackPressed() {  
    val resultIntent = Intent()  
    resultIntent.putExtra("location", location)  
    setResult(Activity.RESULT_OK, resultIntent)  
    finish()  
    super.onBackPressed()  
}
```



Intercept back button press

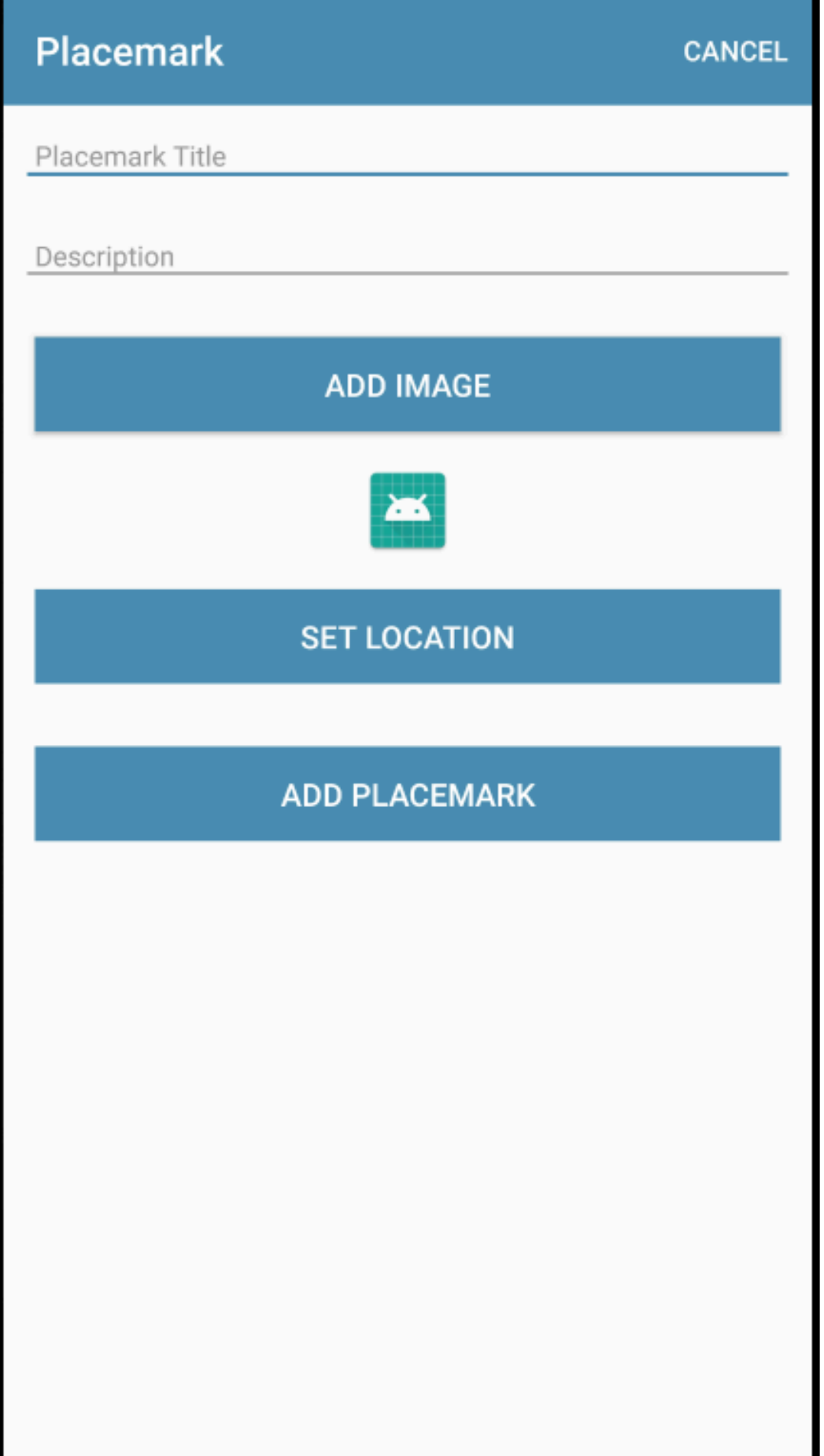
PlacemarkActivity

Send location to MapsActivity + await result

```
placemarkLocation.setOnClickListener {  
    startActivityForResult(intentFor<MapsActivity>().putExtra("location", location), LOCATION_REQUEST)  
}
```

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
    super.onActivityResult(requestCode, resultCode, data)  
    when (requestCode) {  
        IMAGE_REQUEST -> {  
            if (data != null) {  
                placemark.image = data.getData().toString()  
                placemarkImage.setImageBitmap(readImage(this, resultCode, data))  
                chooseImage.setText(R.string.change_placemark_image)  
            }  
        }  
        LOCATION_REQUEST -> {  
            if (data != null) {  
                location = data.extras.getParcelable<Location>("location")  
            }  
        }  
    }  
}
```

If LOCATION_REQUEST
detected, recover location object



Placemark CANCEL

Placemark Title

Description

ADD IMAGE

SET LOCATION

ADD PLACEMARK

