COM4510/6510

Software Development for Mobile Devices

Lab 8: Mapping, Location and Services

Dr Po Yang
The University of Sheffield
po.yang@sheffield.ac.uk

© Po Yang, University of Sheffield

The University Of Sheffield.

Please make sure to read

the next slide (important for your assignment!!!!)

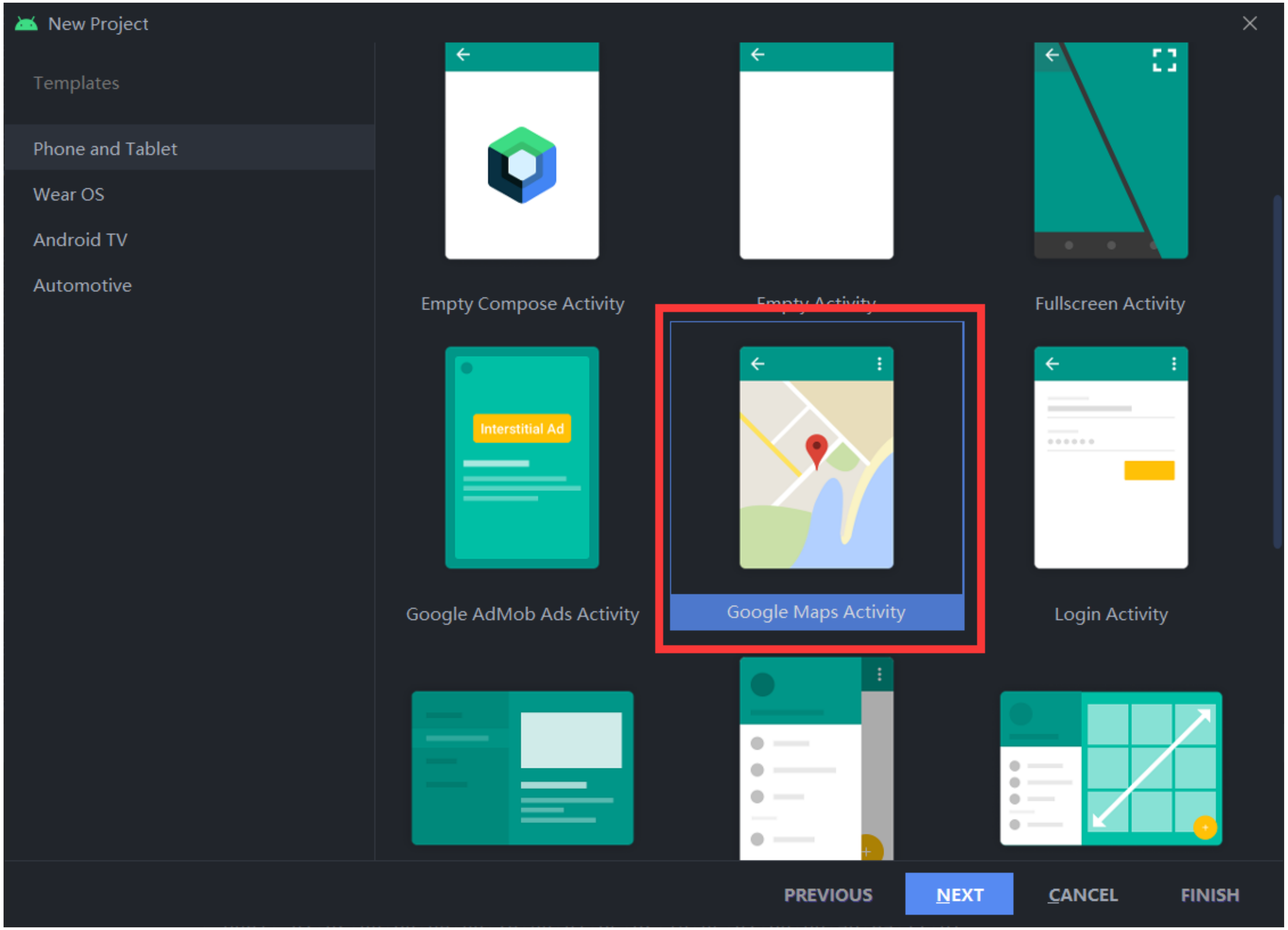
©Po Yang, University of Sheffield

IMPORTANT NOTE

- GIVEN THE RESTRICTIONS ON THE USE OF GOOGLE MAPS
  - it is required that the package is declared in the Android Developer console for every application you develop
    - even for debugging
- So, TODAY AND IN YOUR ASSIGNMENT you MUST either use the package
  - uk.ac.shef.oak.com4510   OR
  - uk.ac.shef.oak.com6510
  - or in case of emergency you can ask me to add your own package (please do not - there are too many of you!)

© Po Yang, University of Sheffield

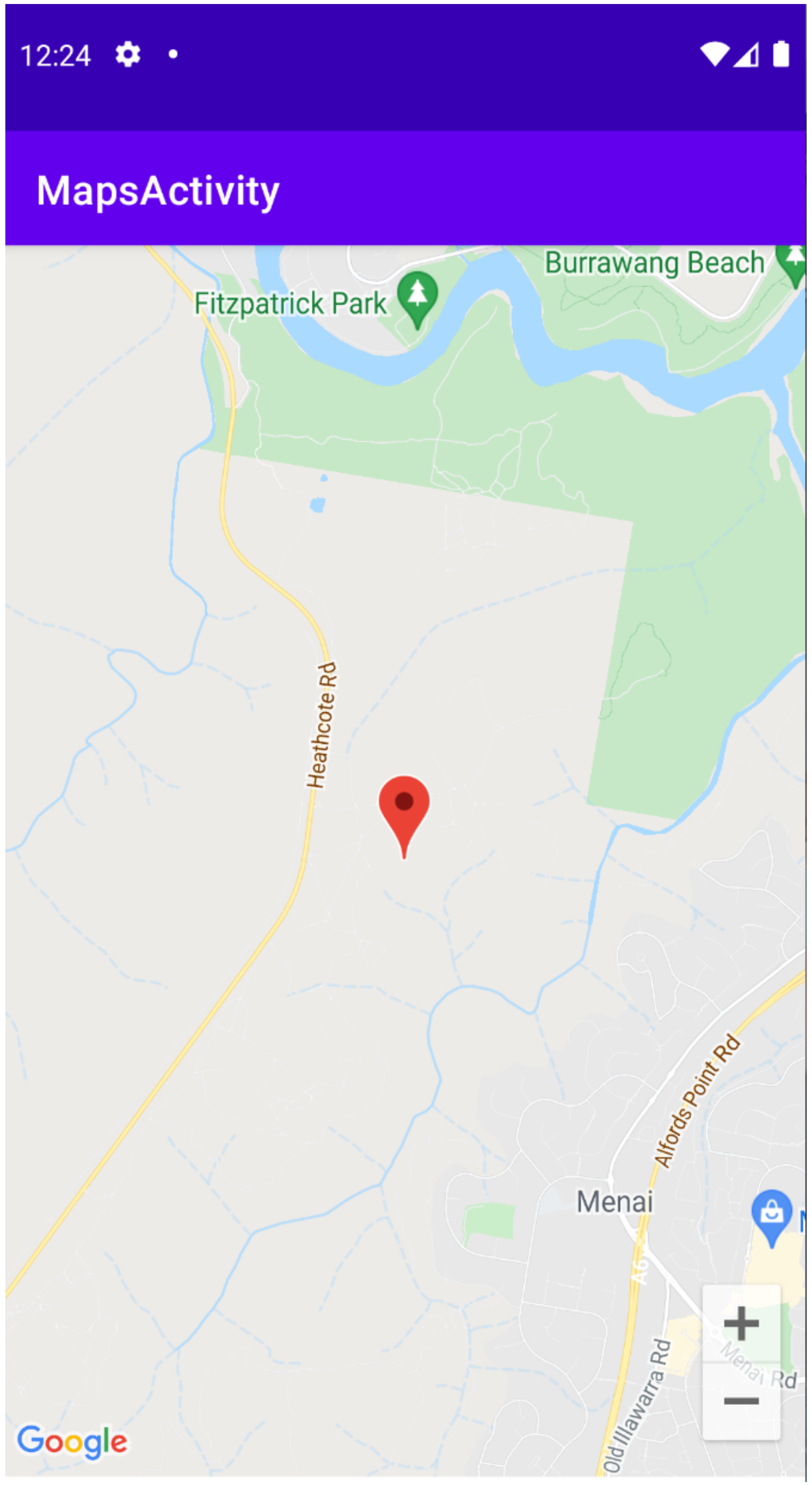
Create a MAP project



© Po Yang, University of Sheffield

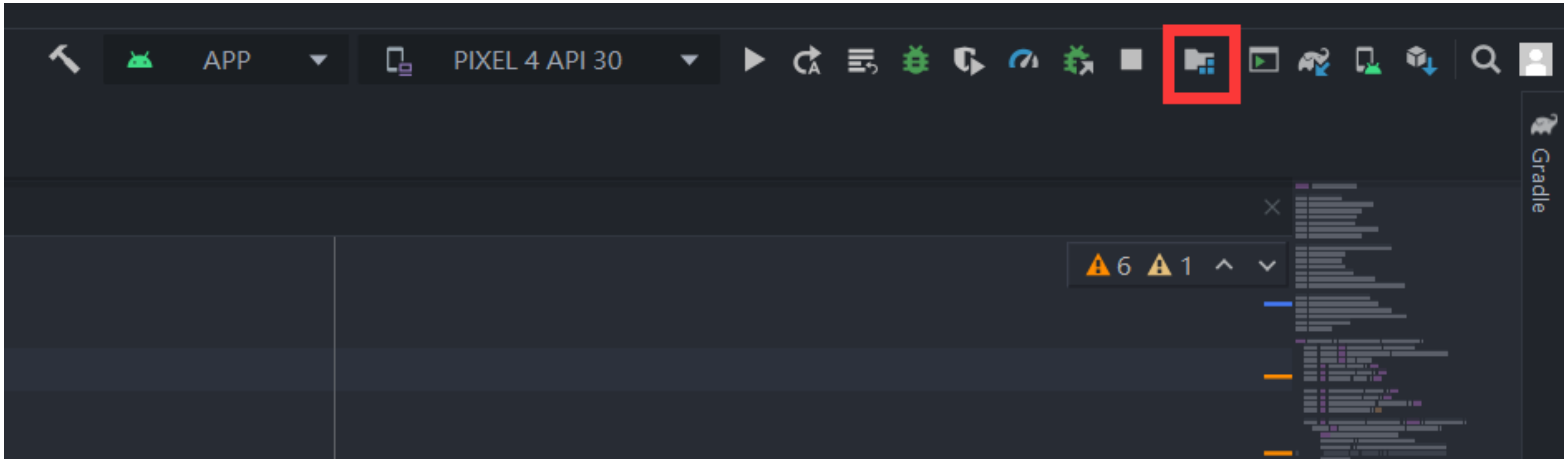
open Maps

- run the project on the emulator
- this is what it should show
  - but it won't
  - we do not have permissions to use Google Maps
  - we have to enable them on the Google Console
  - I have done this for you
    - you must be a registered Android programmer (£50)



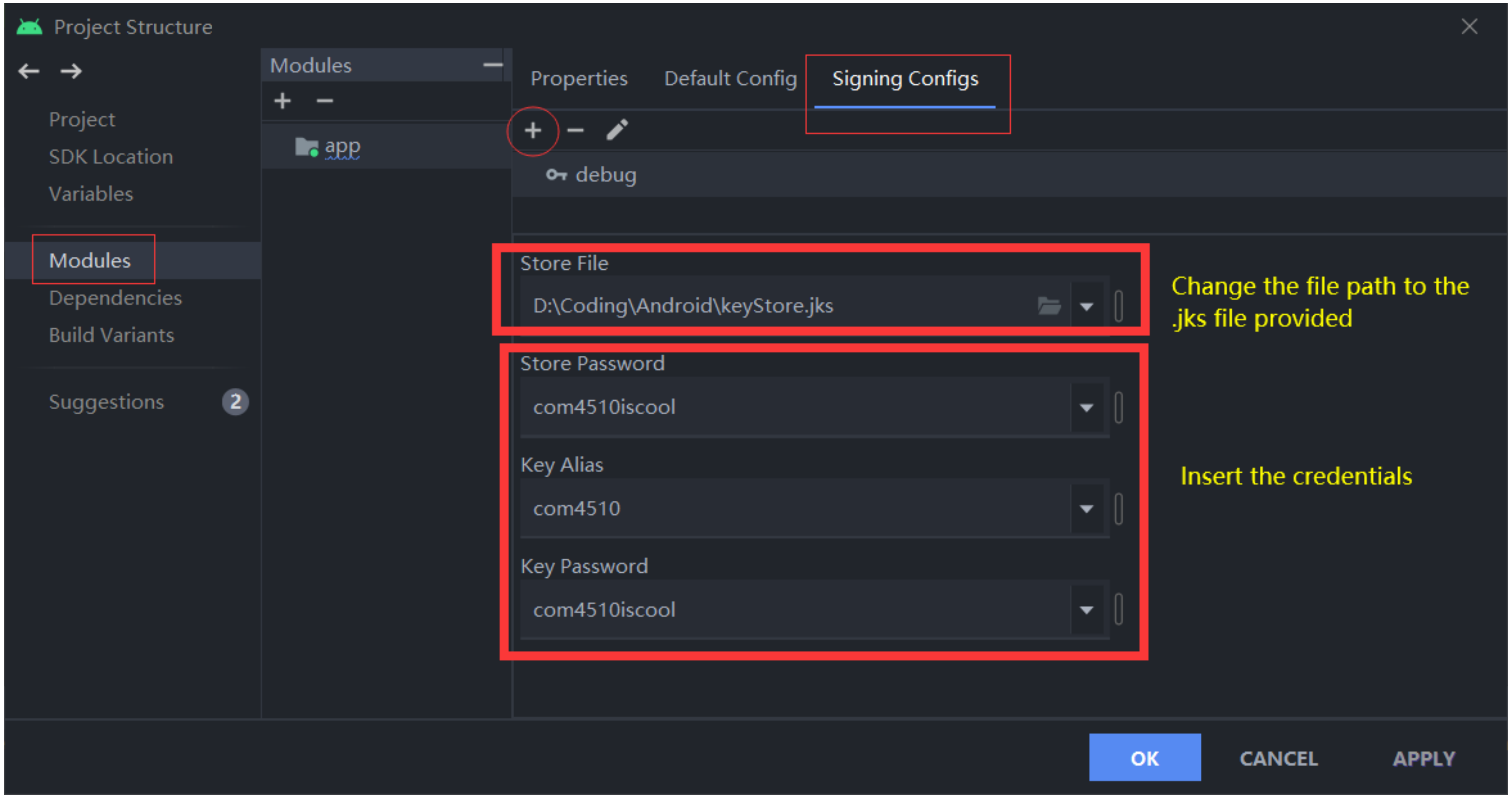
© Po Yang, University of Sheffield

Import the key file



© Po Yang, University of Sheffield

Import the key file



Store File: D:\Coding\Android\keyStore.jks

Store Password: com4510iscool

Key Alias: com4510

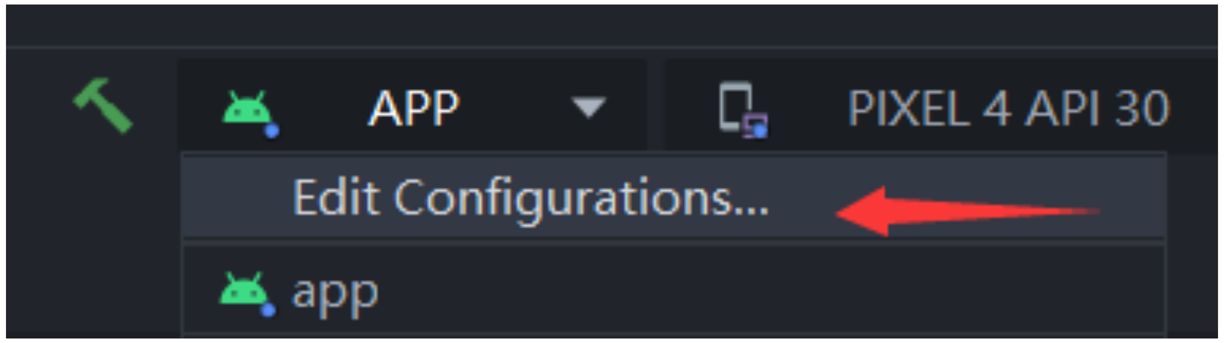
Key Password: com4510iscool

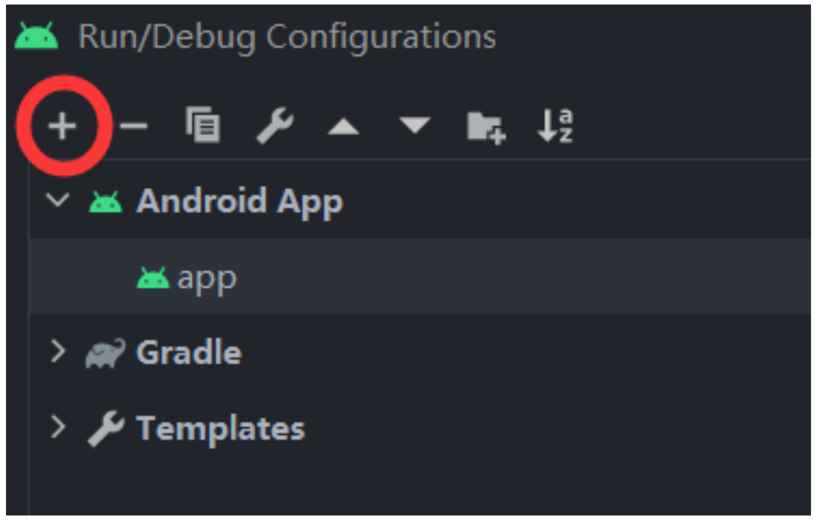
Change the file path to the .jks file provided

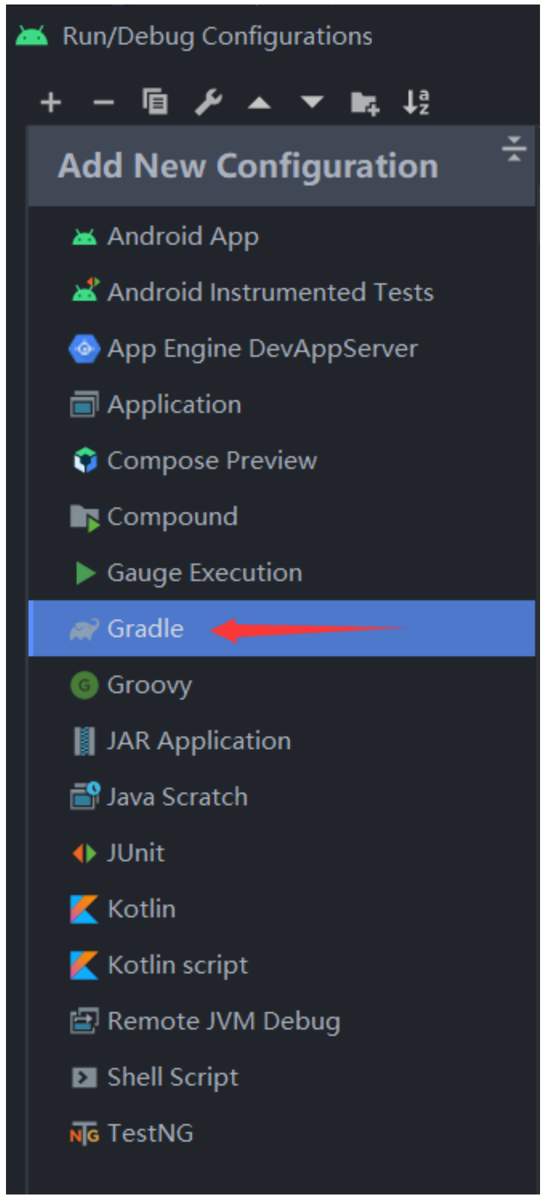
Insert the credentials

© Po Yang, University of Sheffield

To check if it works:


APP
PIXEL 4 API 30
Edit Configurations...
app

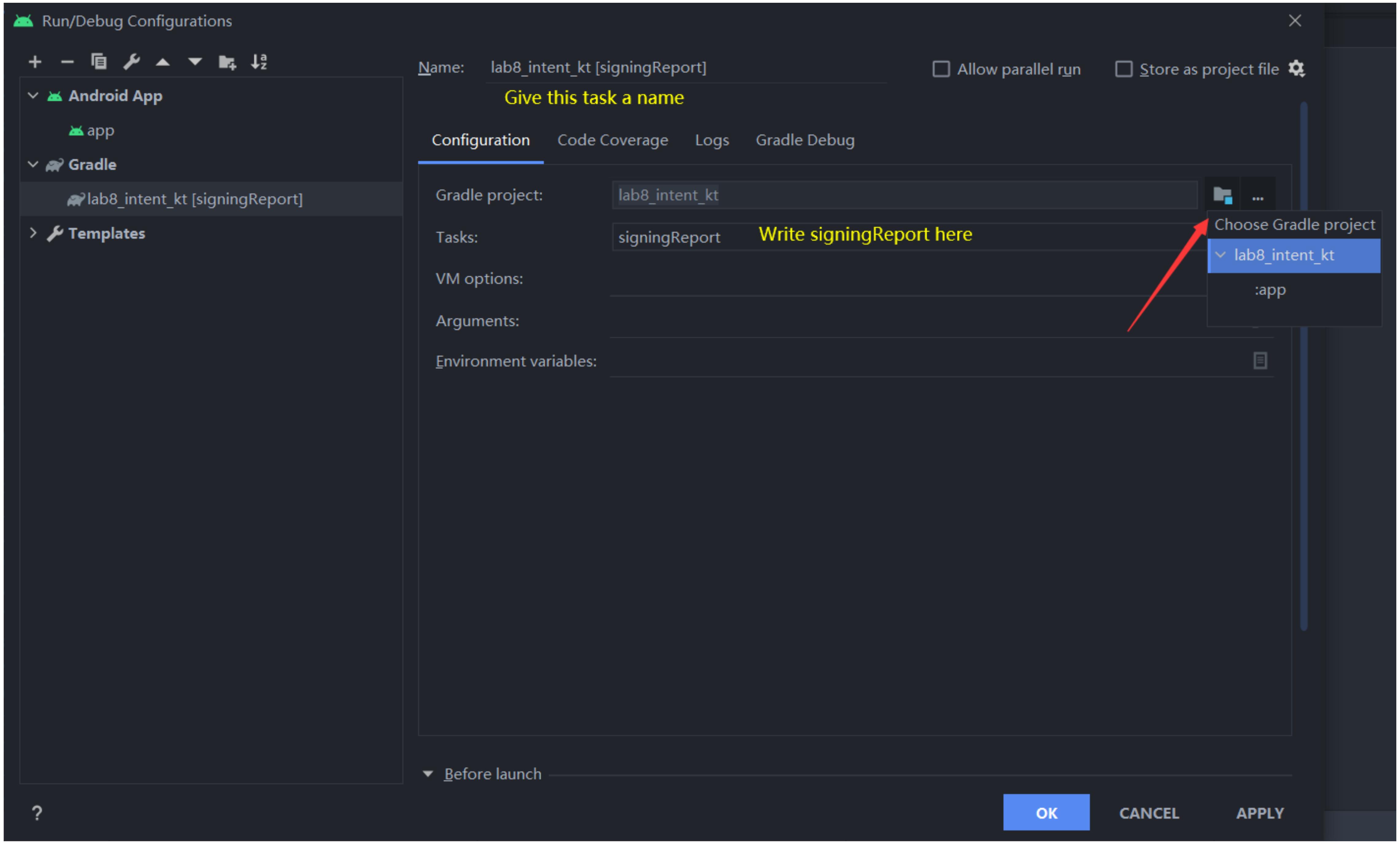

Run/Debug Configurations
Android App
app
Gradle
Templates


Run/Debug Configurations
Add New Configuration
Android App
Android Instrumented Tests
App Engine DevAppServer
Application
Compose Preview
Compound
Gauge Execution
Gradle
Groovy
JAR Application
Java Scratch
JUnit
Kotlin
Kotlin script
Remote JVM Debug
Shell Script
TestNG

© Po Yang, University of Sheffield

The University Of Sheffield.

To check if it works:



Run/Debug Configurations

- Android App
  - app
- Gradle
  - lab8_intent_kt [signingReport]
- Templates

Name: lab8_intent_kt [signingReport]

Allow parallel run ☐ Store as project file ☐

Give this task a name

Configuration | Code Coverage | Logs | Gradle Debug

Gradle project: lab8_intent_kt

Tasks: signingReport

Write signingReport here

VM options:

Arguments:

Environment variables:

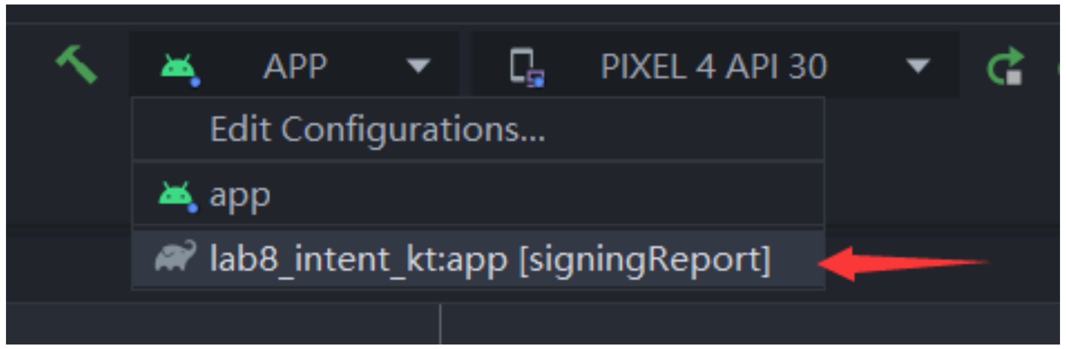
Choose Gradle project
- lab8_intent_kt
  - :app

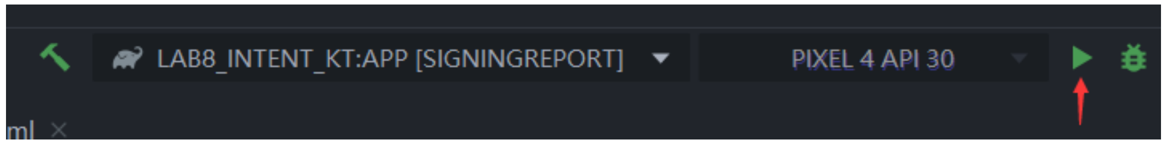
Before launch

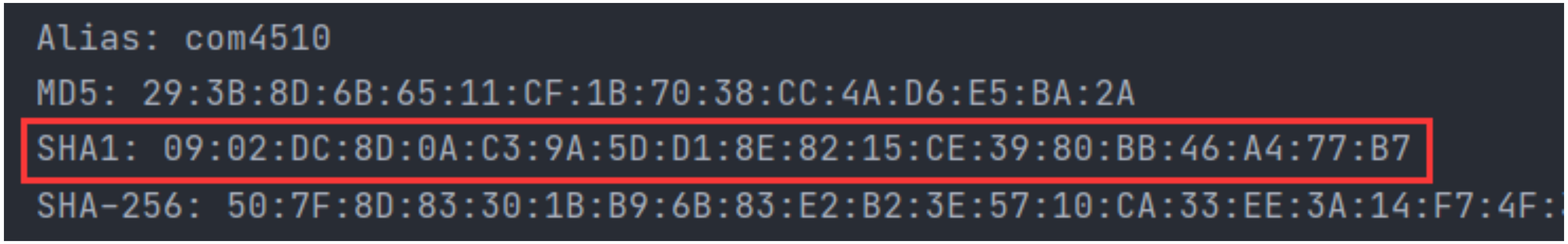
OK CANCEL APPLY

© Po Yang, University of Sheffield

Run the task and check if it is identical





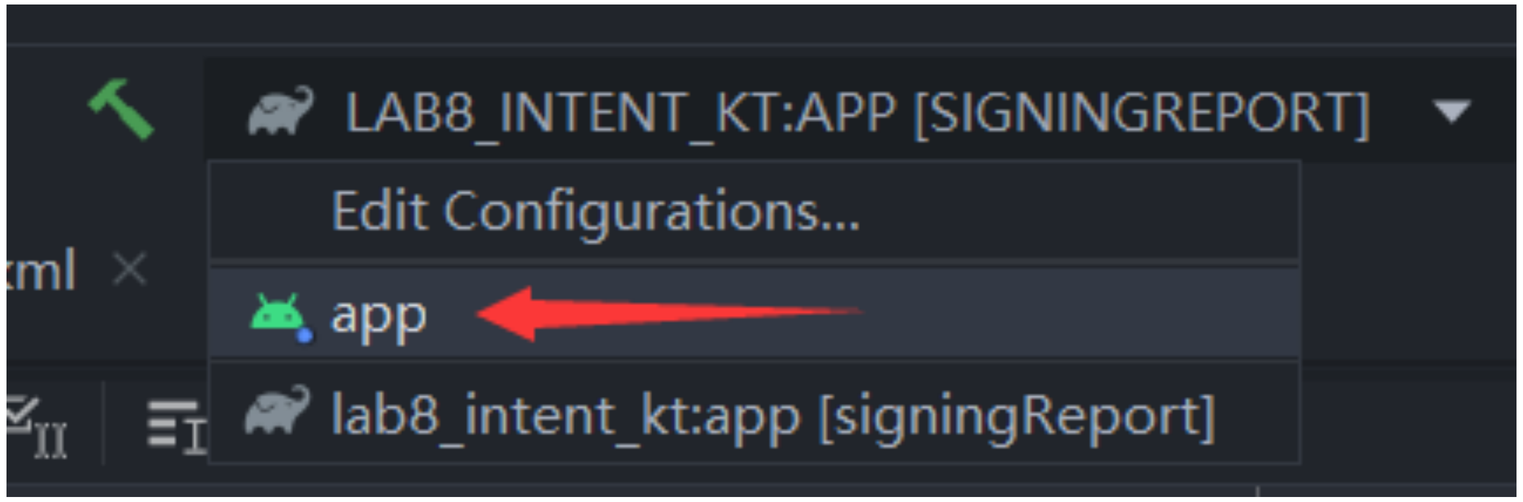


```
Alias: com4510
MD5: 29:3B:8D:6B:65:11:CF:1B:70:38:CC:4A:D6:E5:BA:2A
SHA1: 09:02:DC:8D:0A:C3:9A:5D:D1:8E:82:15:CE:39:80:BB:46:A4:77:B7
SHA-256: 50:7F:8D:83:30:1B:B9:6B:83:E2:B2:3E:57:10:CA:33:EE:3A:14:F7:4F:
```

© Po Yang, University of Sheffield

WARNING

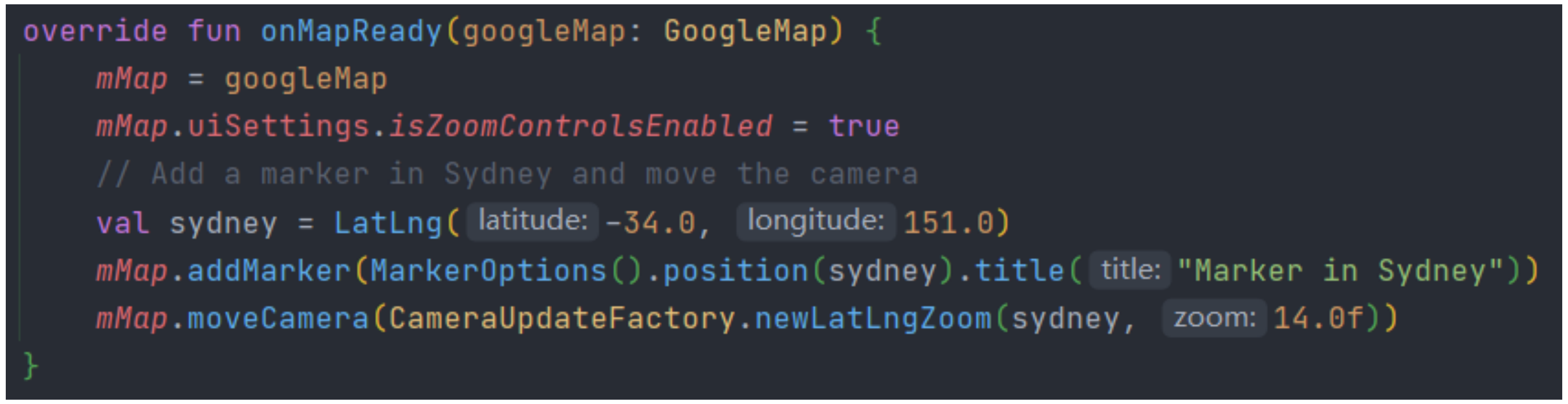
- After this, every time you run the compilation process, Android Studio will no longer run your app. It will run this same signing in process
- Change it back to app


LAB8_INTENT_KT:APP [SIGNINGREPORT]
Edit Configurations...
app
lab8_intent_kt:app [signingReport]
ml

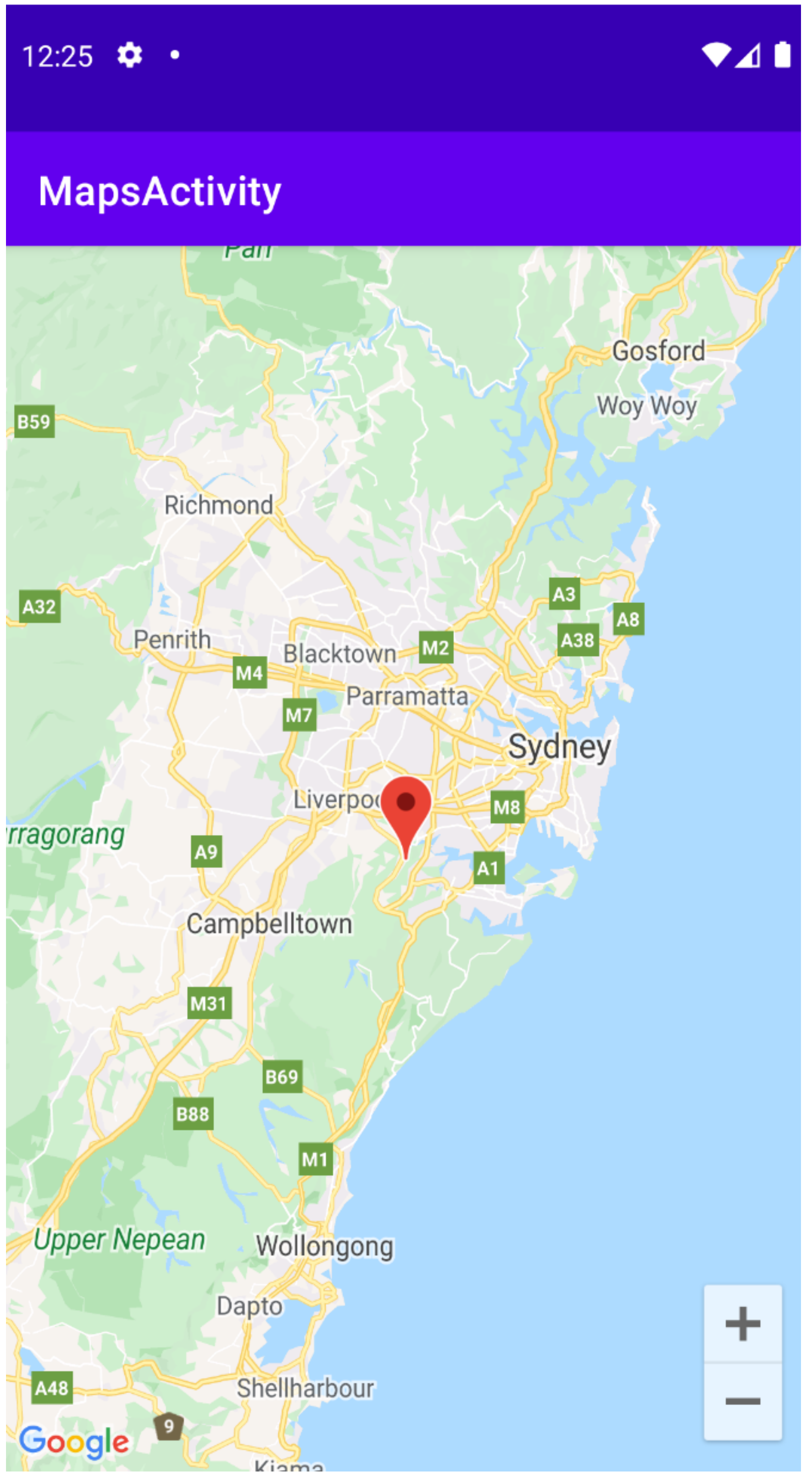
© Po Yang, University of Sheffield

Now it should run

- run the project on the emulator
- it should show you in Sydney Australia
- that is because:

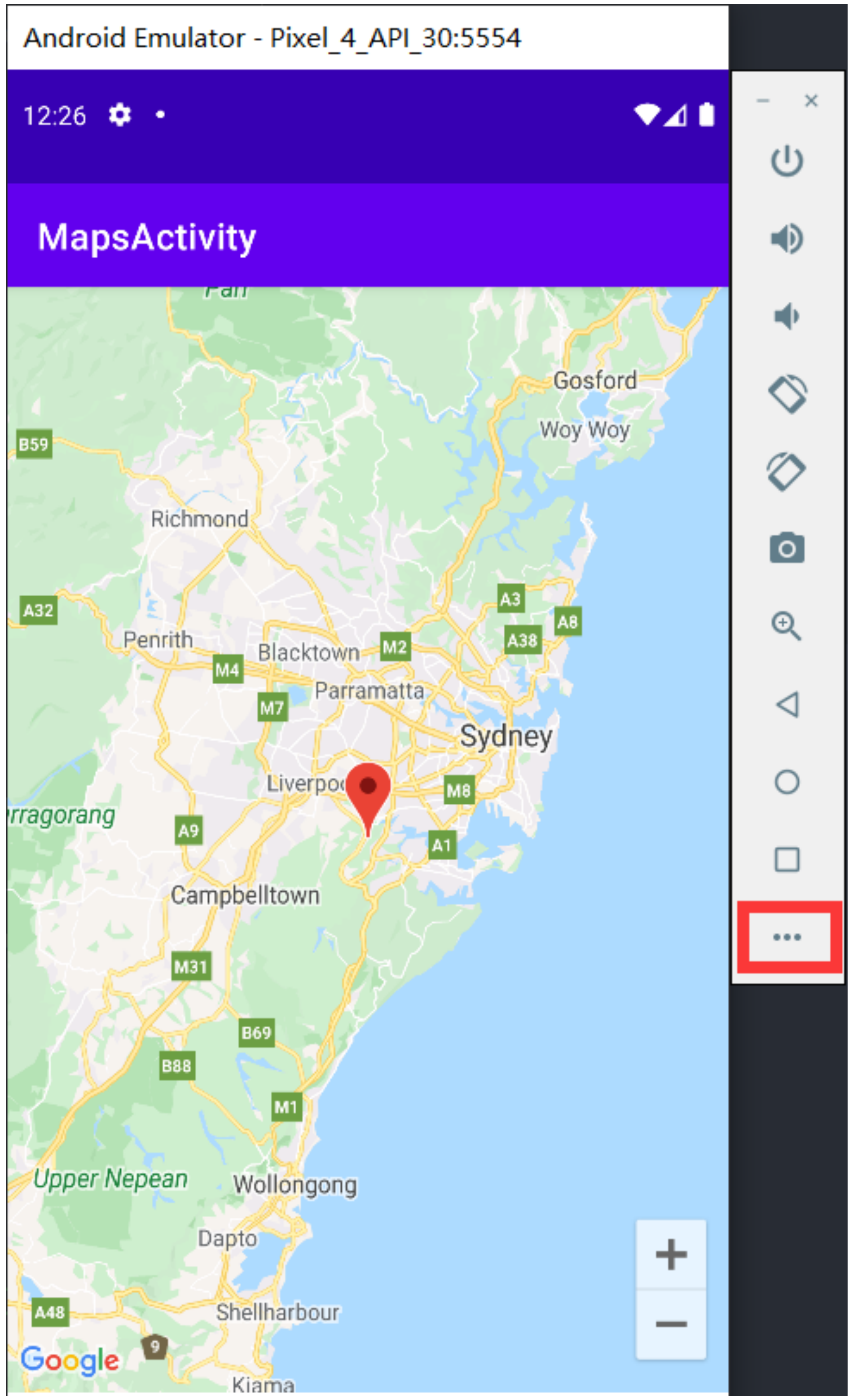


```
override fun onMapReady(googleMap: GoogleMap) {
    mMap = googleMap
    mMap.uiSettings.isZoomControlsEnabled = true
    // Add a marker in Sydney and move the camera
    val sydney = LatLng( latitude: -34.0,  longitude: 151.0)
    mMap.addMarker(MarkerOptions().position(sydney).title( title: "Marker in Sydney"))
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(sydney,  zoom: 14.0f))
}
```



© Po Yang, University of Sheffield

Mock Locations



- It is possible to pass mock locations to the emulator so to simulate moving around
- Click on the three dots

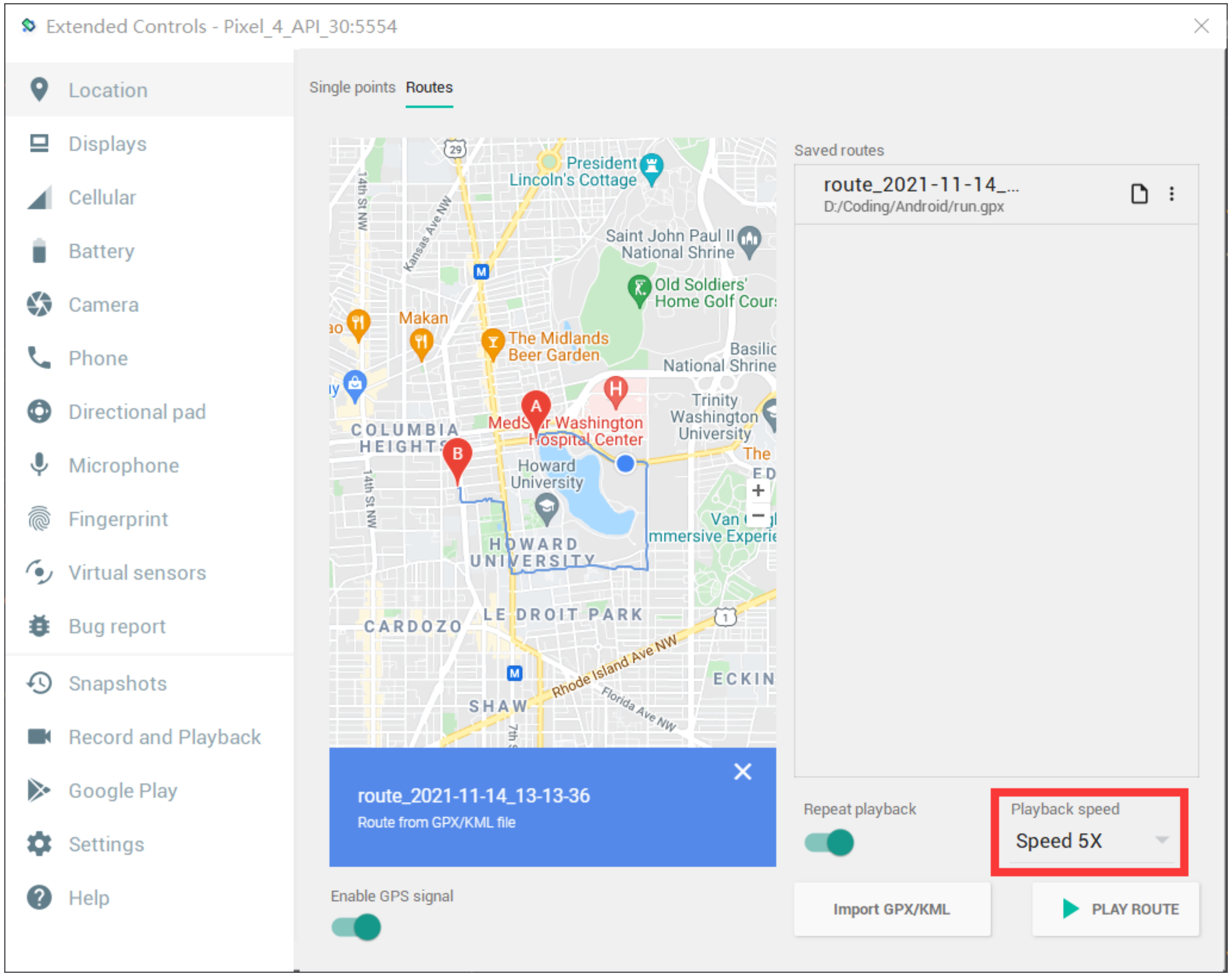
© Po Yang, University of Sheffield

Import GPX file

- Select Location
- you can provide the location manually or you can provide a GPX file to play
  - it simulates a path
  - take one GPX file from - for example -
    - http://www.mountainbikerides.co.uk/downloads/category/3-peak-district-gpx-routes.html

© Po Yang, University of Sheffield

Choose higher speed



route_2021-11-14_...
D:/Coding/Android/run.gpx

route_2021-11-14_13-13-36
Route from GPX/KML file

Enable GPS signal

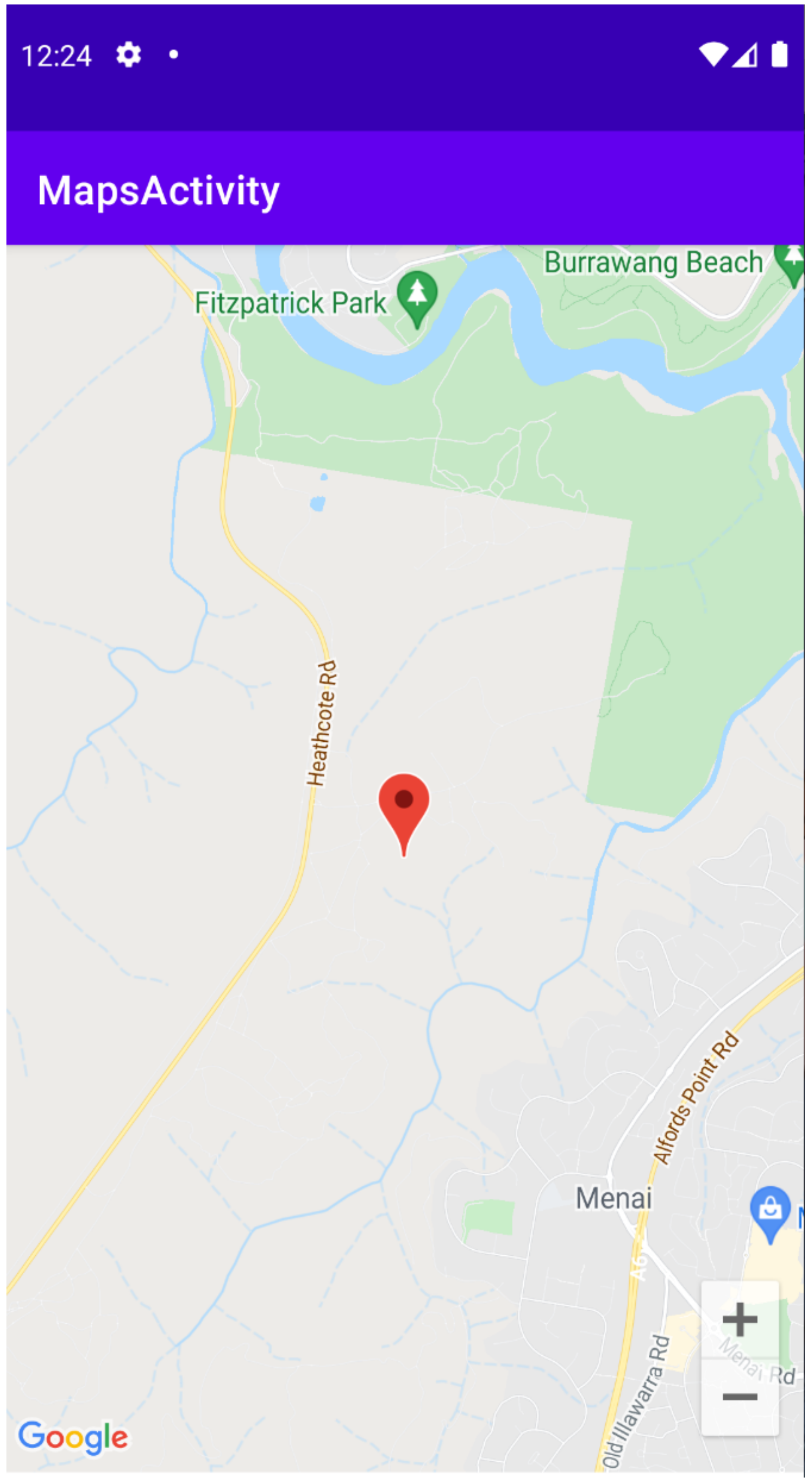
Repeat playback

Playback speed: Speed 5X

© Po Yang, University of Sheffield

open Maps

- run the project on the emulator
- In the new version there is no way to see the location change unless you centre the map manually every time a location is detected
- next exercise will be about doing it automatically



© Po Yang, University of Sheffield

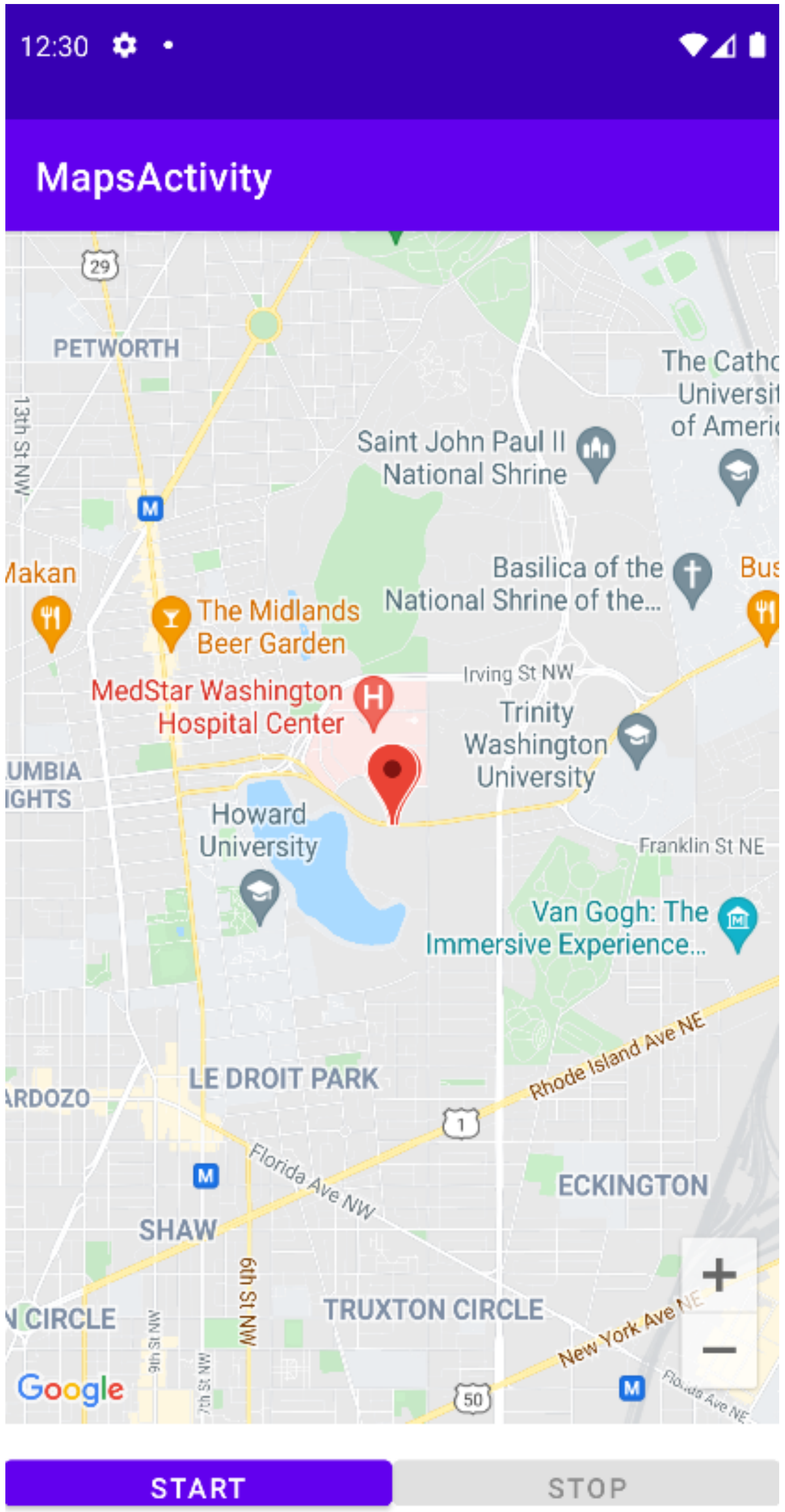
Creating a service tracking location

https://developer.android.com/guide/components/services.html

©Po Yang, University of Sheffield

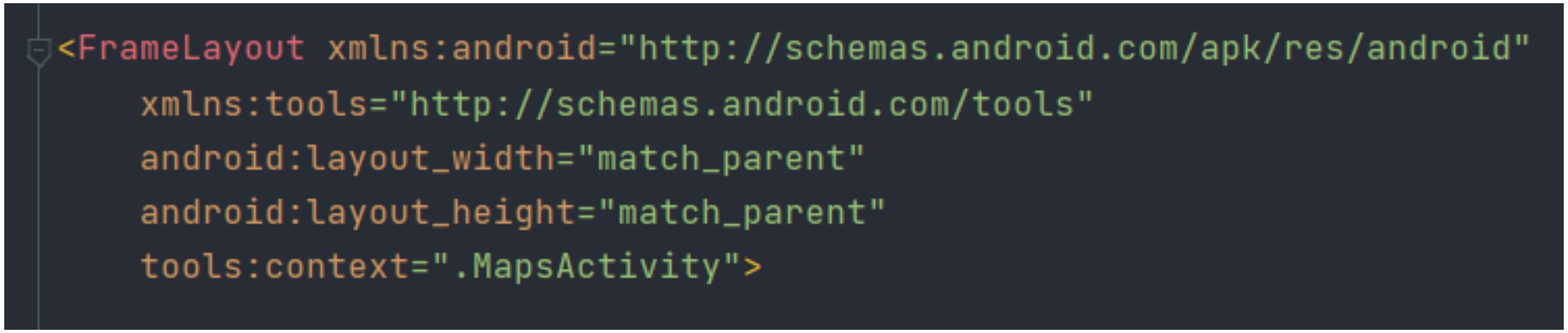
Start/Stop location tracking

- Add two buttons to the activity
- Start button starting location updates
- Stop button stopping location updates
- in your onCreate, get hold of the buttons and define two empty callbacks for clicking


12:30
MapsActivity
PETWORTH
Saint John Paul II National Shrine
Basilica of the National Shrine of the...
The Midlands Beer Garden
MedStar Washington Hospital Center
Trinity Washington University
Howard University
Van Gogh: The Immersive Experience...
LE DROIT PARK
ECKINGTON
SHAW
TRUXTON CIRCLE
Google
START
STOP

© Po Yang, University of Sheffield

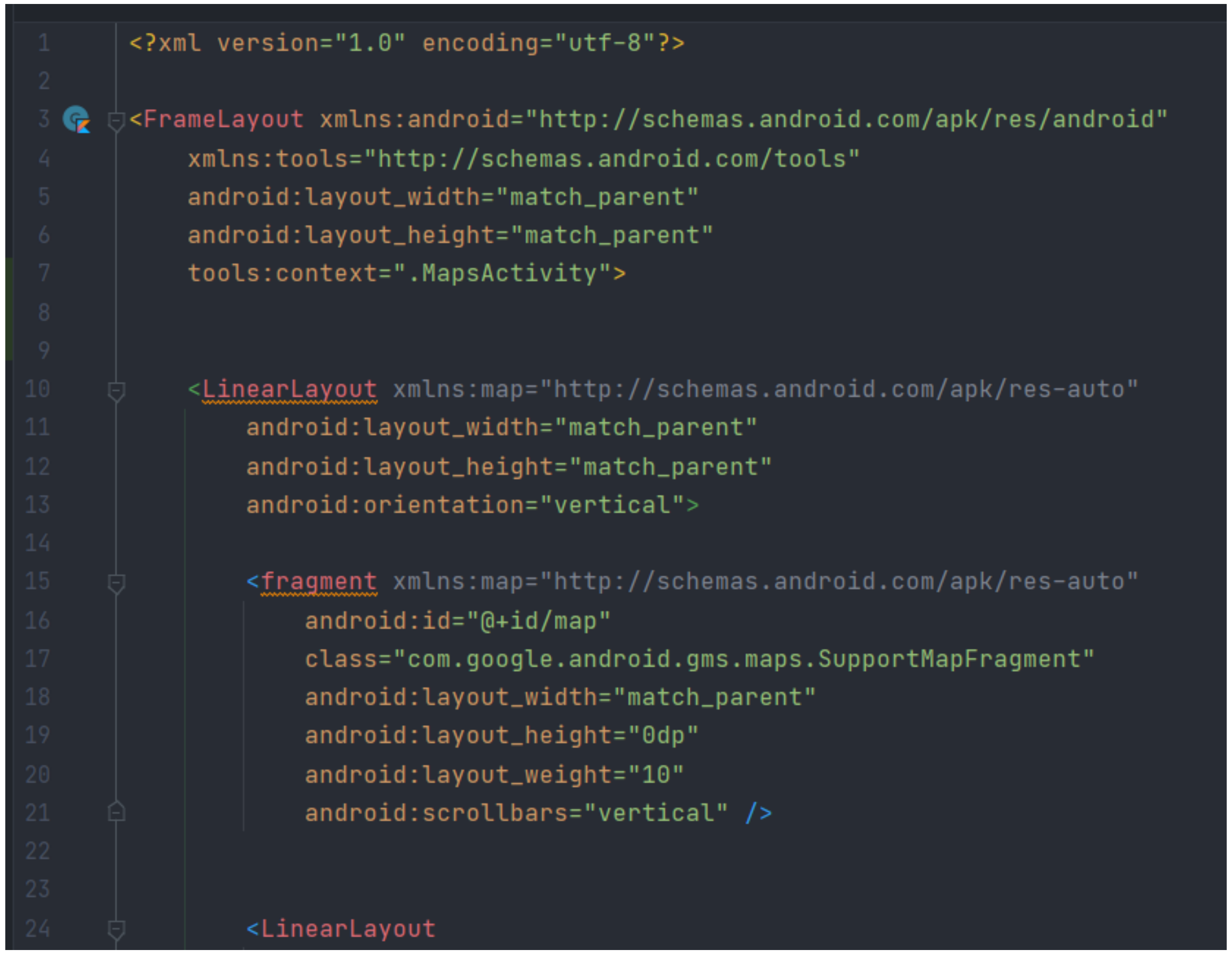
Initial layout



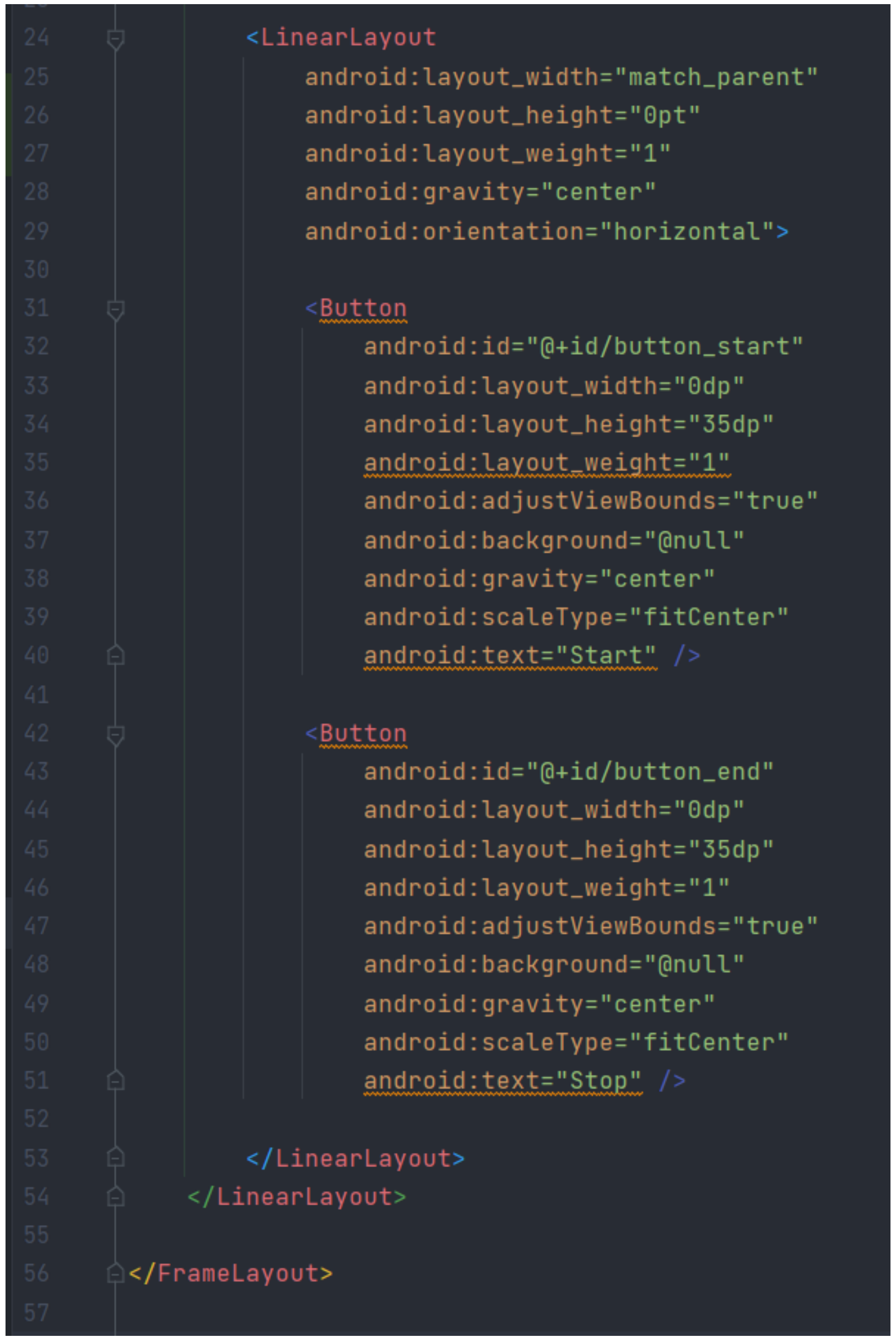
```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity">
```

© Po Yang, University of Sheffield

Final Layout



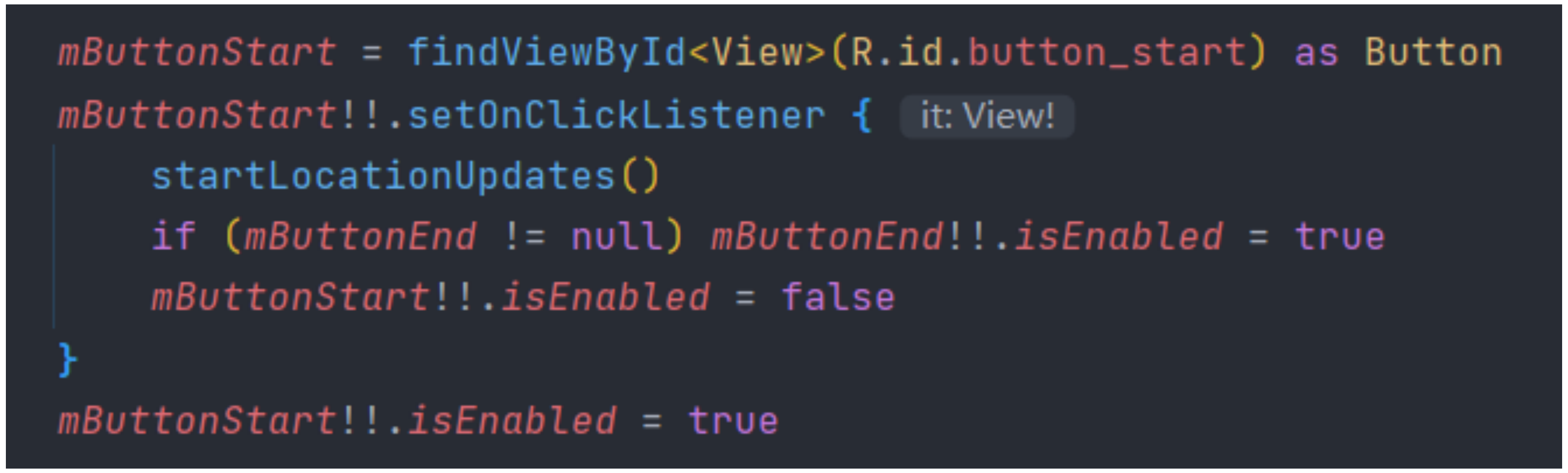
```xml
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity">


    <LinearLayout xmlns:map="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <fragment xmlns:map="http://schemas.android.com/apk/res-auto"
            android:id="@+id/map"
            class="com.google.android.gms.maps.SupportMapFragment"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="10"
            android:scrollbars="vertical" />


        <LinearLayout
```



```xml
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0pt"
            android:layout_weight="1"
            android:gravity="center"
            android:orientation="horizontal">

            <Button
                android:id="@+id/button_start"
                android:layout_width="0dp"
                android:layout_height="35dp"
                android:layout_weight="1"
                android:adjustViewBounds="true"
                android:background="@null"
                android:gravity="center"
                android:scaleType="fitCenter"
                android:text="Start" />

            <Button
                android:id="@+id/button_end"
                android:layout_width="0dp"
                android:layout_height="35dp"
                android:layout_weight="1"
                android:adjustViewBounds="true"
                android:background="@null"
                android:gravity="center"
                android:scaleType="fitCenter"
                android:text="Stop" />

        </LinearLayout>
    </LinearLayout>

</FrameLayout>

```

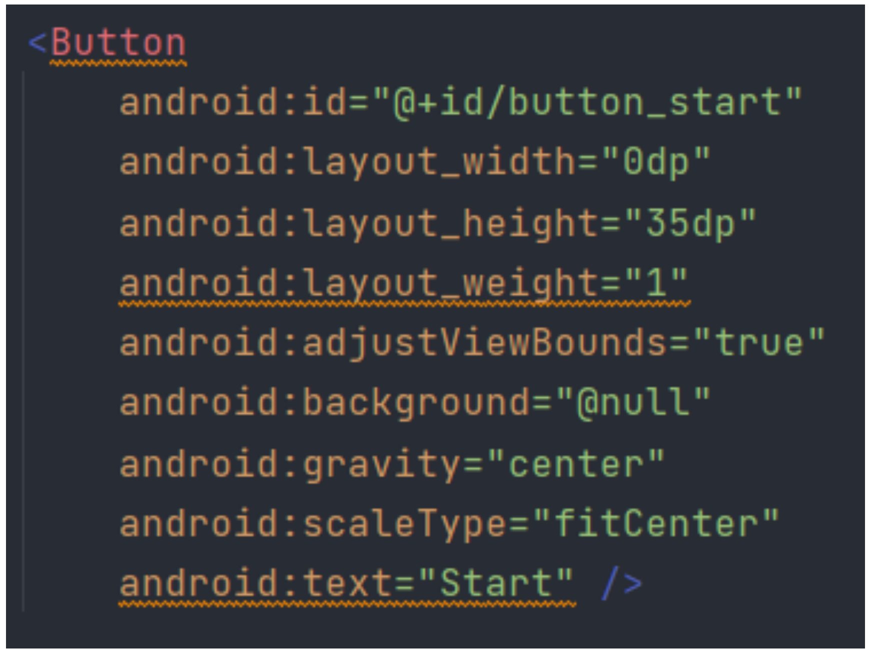
© Po Yang, University of Sheffield

Button



```kotlin
mButtonStart = findViewById<View>(R.id.button_start) as Button
mButtonStart!!.setOnClickListener {  it: View!
    startLocationUpdates()
    if (mButtonEnd != null) mButtonEnd!!.isEnabled = true
    mButtonStart!!.isEnabled = false
}
mButtonStart!!.isEnabled = true
```

(in your xml layout file define something like:



```xml
<Button
    android:id="@+id/button_start"
    android:layout_width="0dp"
    android:layout_height="35dp"
    android:layout_weight="1"
    android:adjustViewBounds="true"
    android:background="@null"
    android:gravity="center"
    android:scaleType="fitCenter"
    android:text="Start" />
```

Do the same for the stop button

© Po Yang, University of Sheffield

OnclickListener

- You should call the two methods for start and stop of the location tracking
  - this should be straightforward

© Po Yang, University of Sheffield

We will see two ways of doing this

- On the UI thread
  - useful only if the user is constantly looking at the map
- in a separate process
  - to track the user in the background

© Po Yang, University of Sheffield

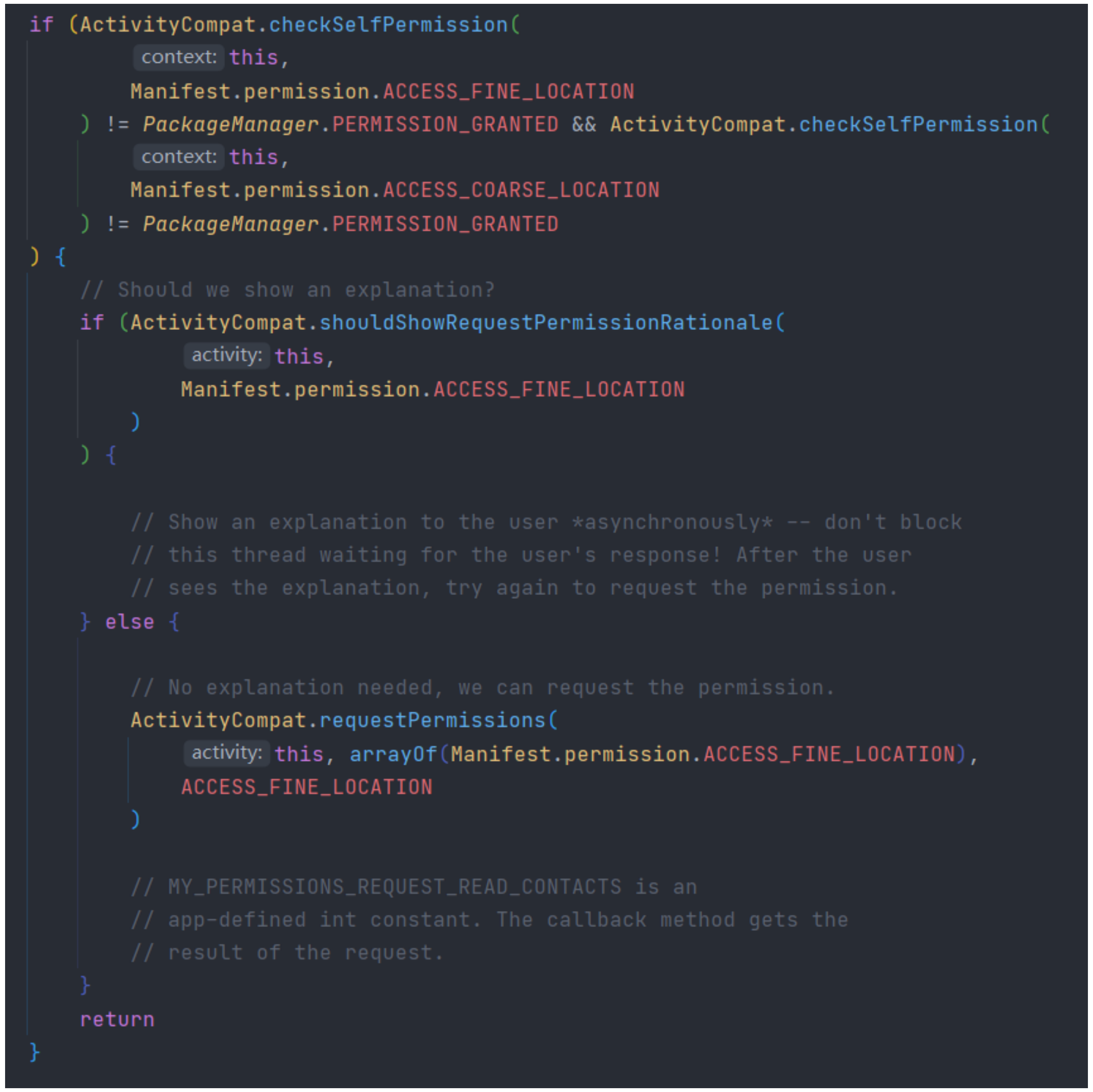
On UI Thread

See last week's lab class

- We need two variables

```
private LocationRequest mLocationRequest;
private FusedLocationProviderClient mFusedLocationClient;
```

- A way to ask
- for permissions

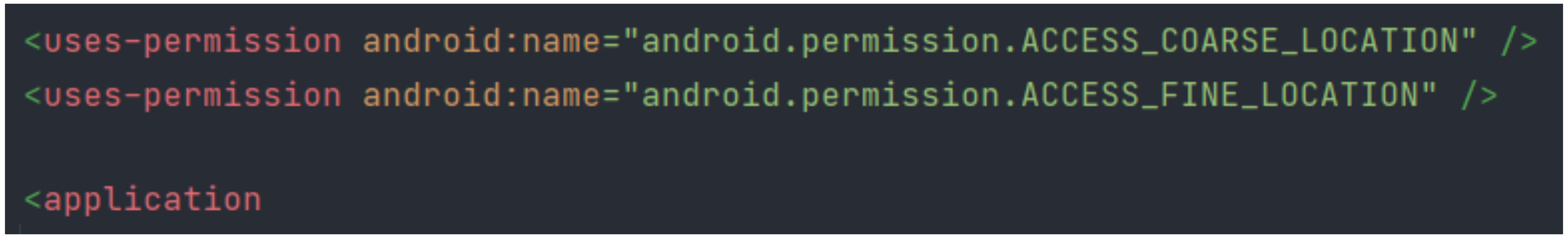


```
if (ActivityCompat.checkSelfPermission(
        context: this,
        Manifest.permission.ACCESS_FINE_LOCATION
    ) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
        context: this,
        Manifest.permission.ACCESS_COARSE_LOCATION
    ) != PackageManager.PERMISSION_GRANTED
) {
    // Should we show an explanation?
    if (ActivityCompat.shouldShowRequestPermissionRationale(
            activity: this,
            Manifest.permission.ACCESS_FINE_LOCATION
        )
    ) {

        // Show an explanation to the user *asynchronously* -- don't block
        // this thread waiting for the user's response! After the user
        // sees the explanation, try again to request the permission.
    } else {

        // No explanation needed, we can request the permission.
        ActivityCompat.requestPermissions(
            activity: this, arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
            ACCESS_FINE_LOCATION
        )

        // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
        // app-defined int constant. The callback method gets the
        // result of the request.
    }
    return
}
```

© Po Yang, University of Sheffield

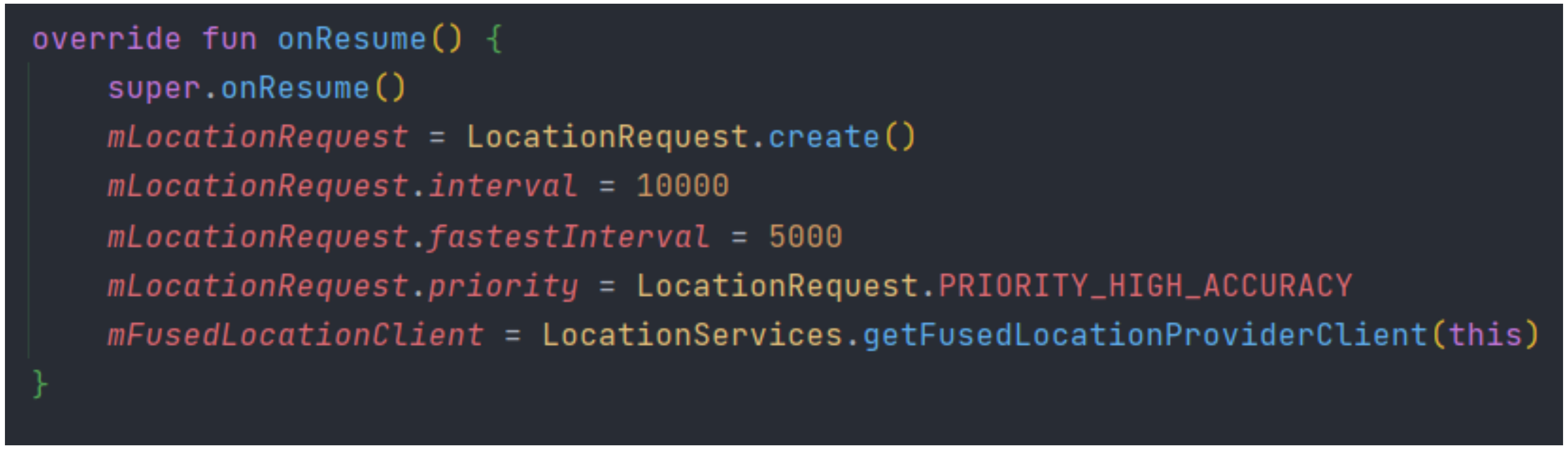
- Remember to add the permissions in Manifest file

–>



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

<application
```

- In the onResume we need to establish the tracker parameters

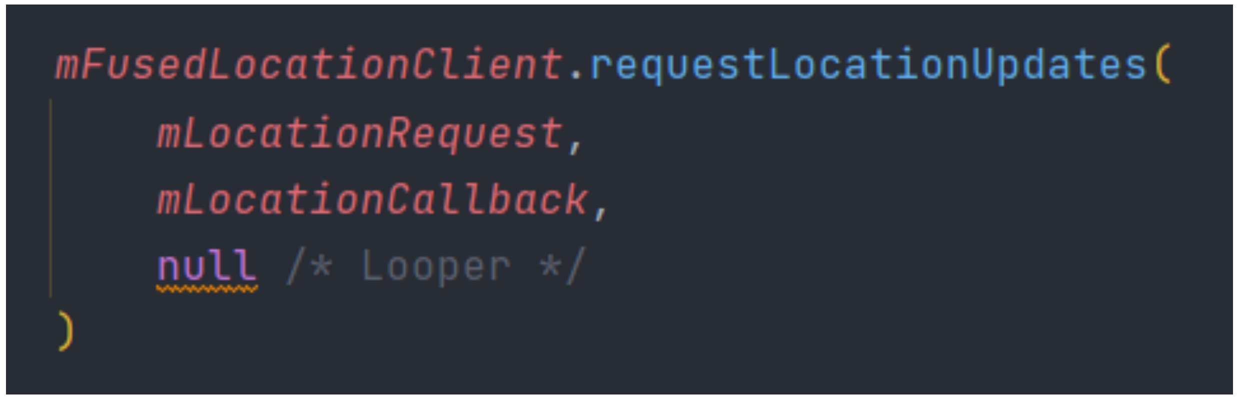


```kotlin
override fun onResume() {
    super.onResume()
    mLocationRequest = LocationRequest.create()
    mLocationRequest.interval = 10000
    mLocationRequest.fastestInterval = 5000
    mLocationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY
    mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
}
```

© Po Yang, University of Sheffield

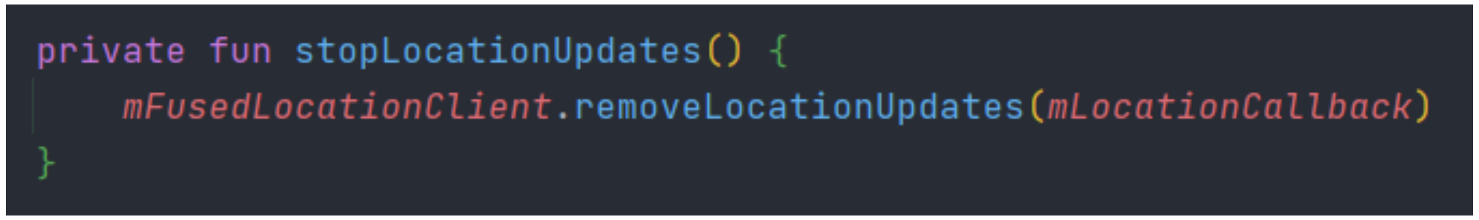
Start/Stop

- We need to start location tracking



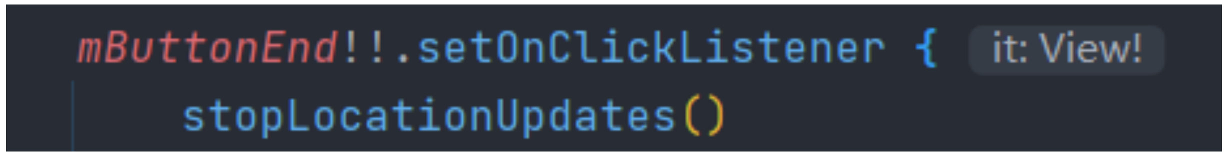
```
mFusedLocationClient.requestLocationUpdates(
    mLocationRequest,
    mLocationCallback,
    null /* Looper */
)
```

- And a way to stop them



```
private fun stopLocationUpdates() {
    mFusedLocationClient.removeLocationUpdates(mLocationCallback)
}
```

- Connect these to the buttons



```
mButtonEnd!!.setOnClickListener {  it: View!
    stopLocationUpdates()
```

Location in the background

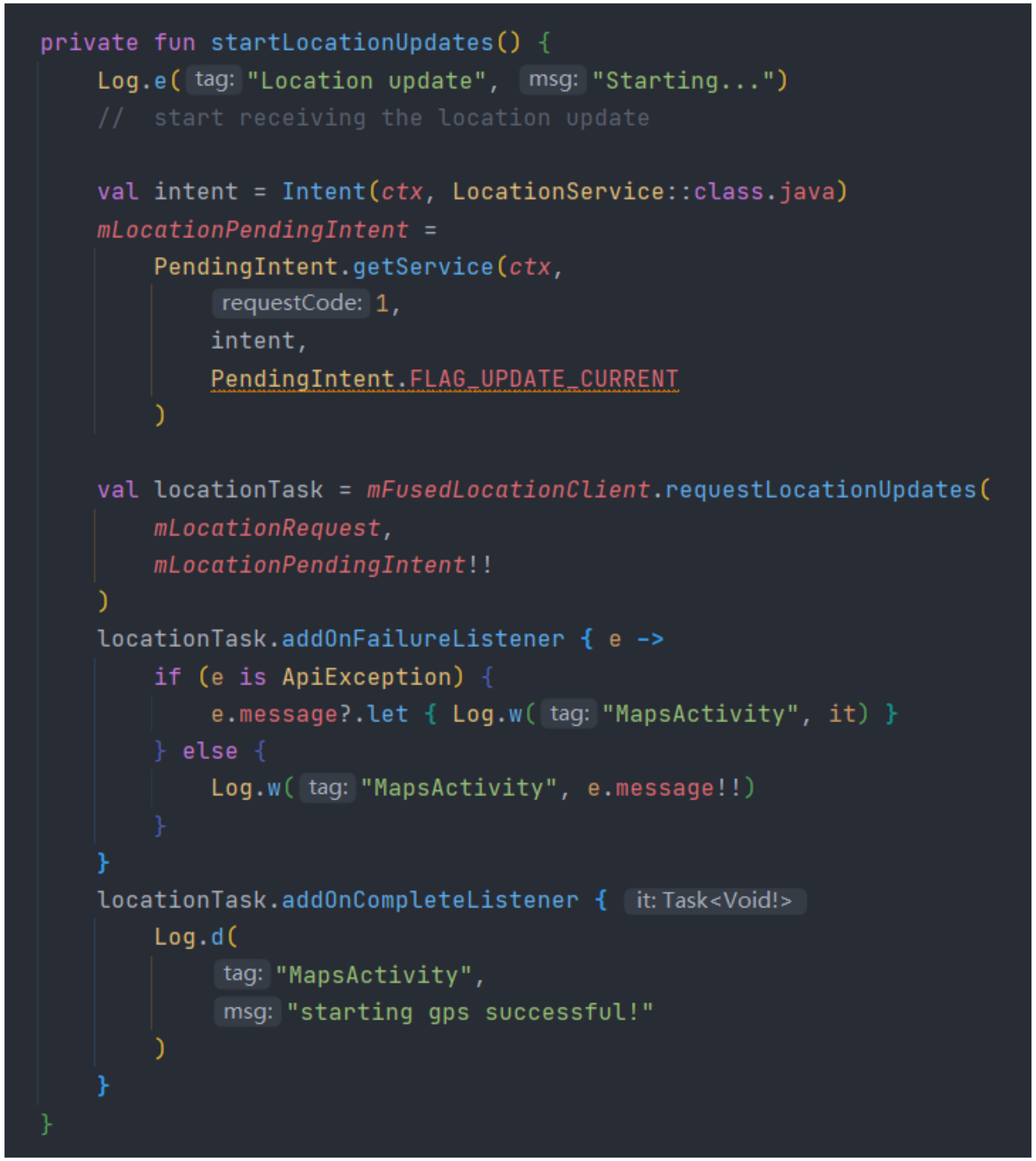
©Po Yang, University of Sheffield

Location in background

- Tracking locations in the background requires an IntentService
  - create an Service (LocationIntent)
  - create a location provider, the FusedLocationClient
    - define what to do when the creation of the task is successful
    - define what to do when unsuccessful
  - Define an onStartCommand method to capture the identification of the location

© Po Yang, University of Sheffield

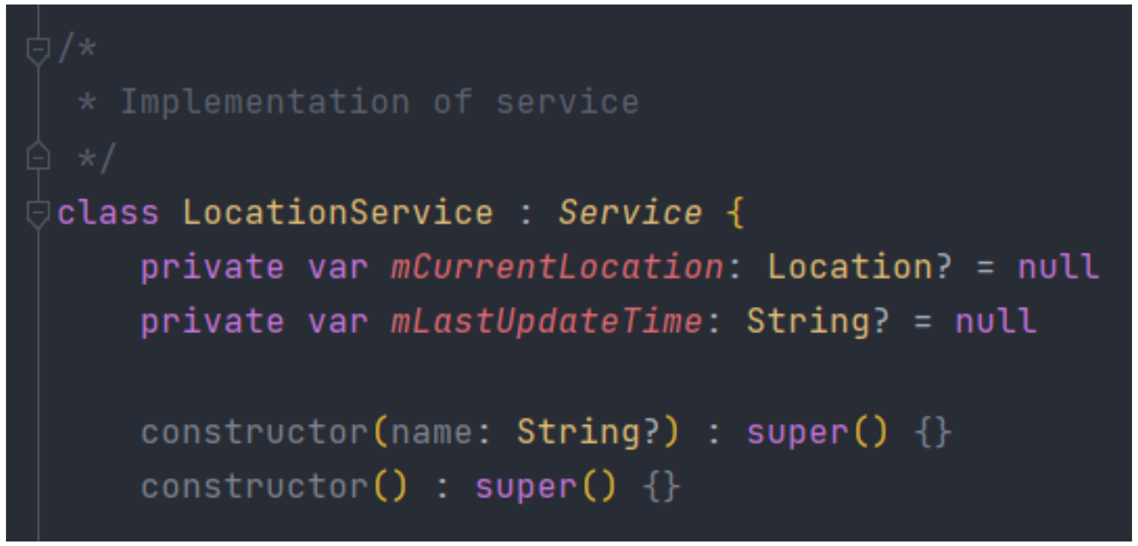
in the onclicklistener for start



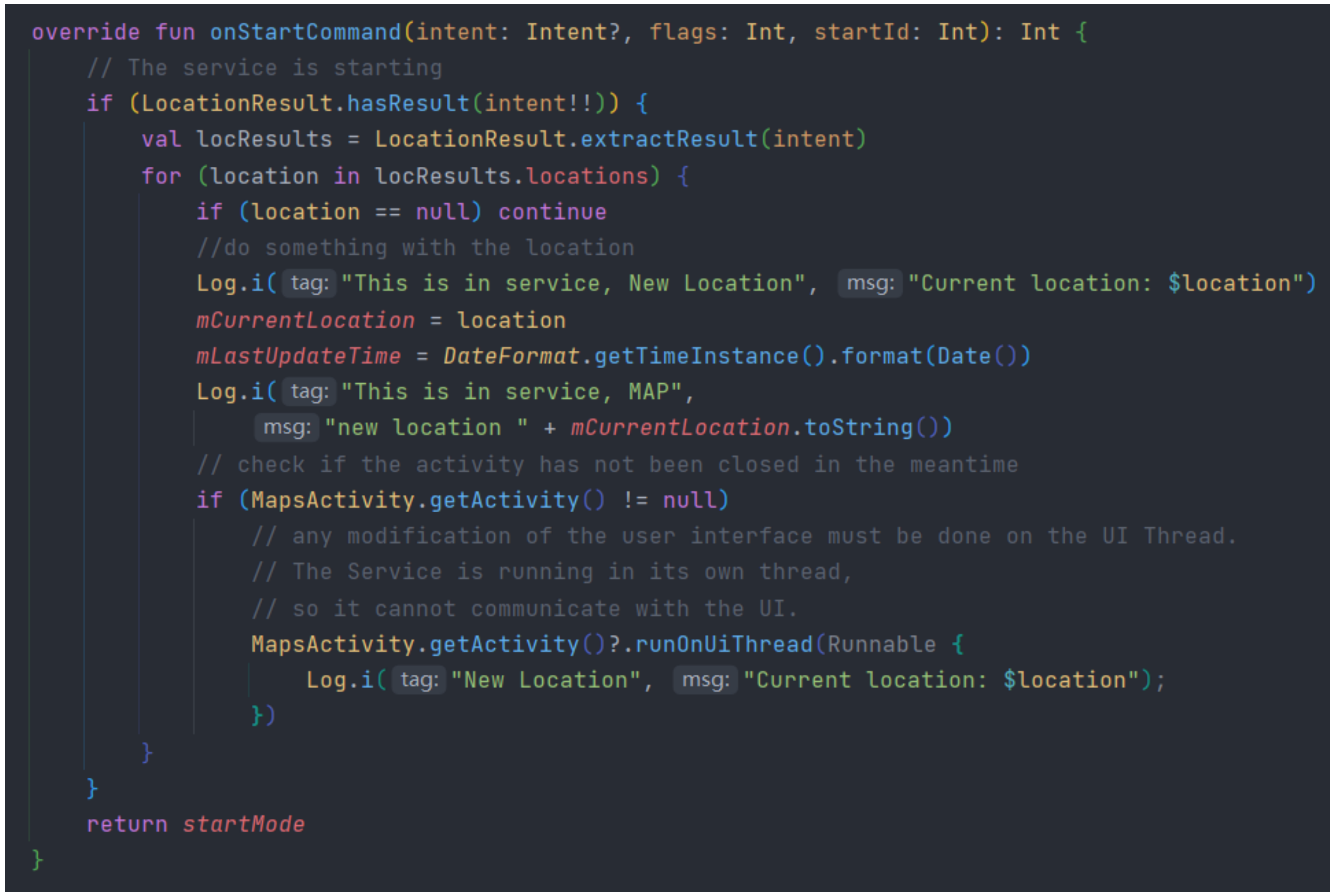
```
private fun startLocationUpdates() {
    Log.e( tag: "Location update",  msg: "Starting...")
    //  start receiving the location update

    val intent = Intent(ctx, LocationService::class.java)
    mLocationPendingIntent =
        PendingIntent.getService(ctx,
            requestCode: 1,
            intent,
            PendingIntent.FLAG_UPDATE_CURRENT
        )

    val locationTask = mFusedLocationClient.requestLocationUpdates(
        mLocationRequest,
        mLocationPendingIntent!!
    )
    locationTask.addOnFailureListener { e ->
        if (e is ApiException) {
            e.message?.let { Log.w( tag: "MapsActivity", it) }
        } else {
            Log.w( tag: "MapsActivity", e.message!!)
        }
    }
    locationTask.addOnCompleteListener {  it: Task<Void!>
        Log.d(
            tag: "MapsActivity",
            msg: "starting gps successful!"
        )
    }
}
```

© Po Yang, University of Sheffield

Location Service



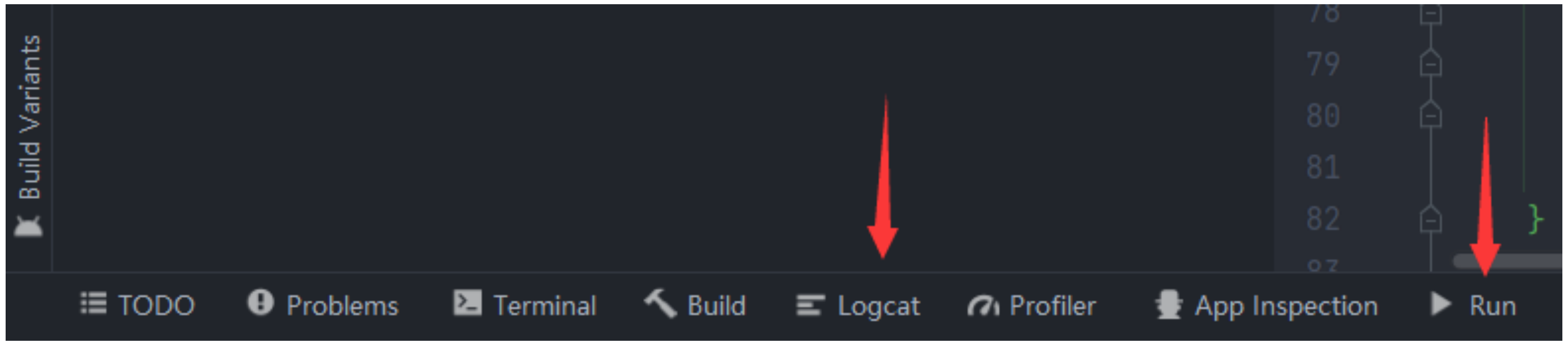
```kotlin
/*
 * Implementation of service
 */
class LocationService : Service {
    private var mCurrentLocation: Location? = null
    private var mLastUpdateTime: String? = null

    constructor(name: String?) : super() {}
    constructor() : super() {}
```



```kotlin
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
    // The service is starting
    if (LocationResult.hasResult(intent!!)) {
        val locResults = LocationResult.extractResult(intent)
        for (location in locResults.locations) {
            if (location == null) continue
            //do something with the location
            Log.i( tag: "This is in service, New Location",  msg: "Current location: $location")
            mCurrentLocation = location
            mLastUpdateTime = DateFormat.getTimeInstance().format(Date())
            Log.i( tag: "This is in service, MAP",
                 msg: "new location " + mCurrentLocation.toString())
            // check if the activity has not been closed in the meantime
            if (MapsActivity.getActivity() != null)
                // any modification of the user interface must be done on the UI Thread.
                // The Service is running in its own thread,
                // so it cannot communicate with the UI.
                MapsActivity.getActivity()?.runOnUiThread(Runnable {
                    Log.i( tag: "New Location",  msg: "Current location: $location");
                })
        }
    }
    return startMode
}
```

© Po Yang, University of Sheffield

Now test

- Use the GPX file to send locations to the emulator
- Open the logical window to check that locations are recognised
- 

- (or set up a debugger break)

  - TRY TO RUN AND THEN SEE NEXT SLIDE

© Po Yang, University of Sheffield

Visualised the tracked locations on a Map

©Po Yang, University of Sheffield

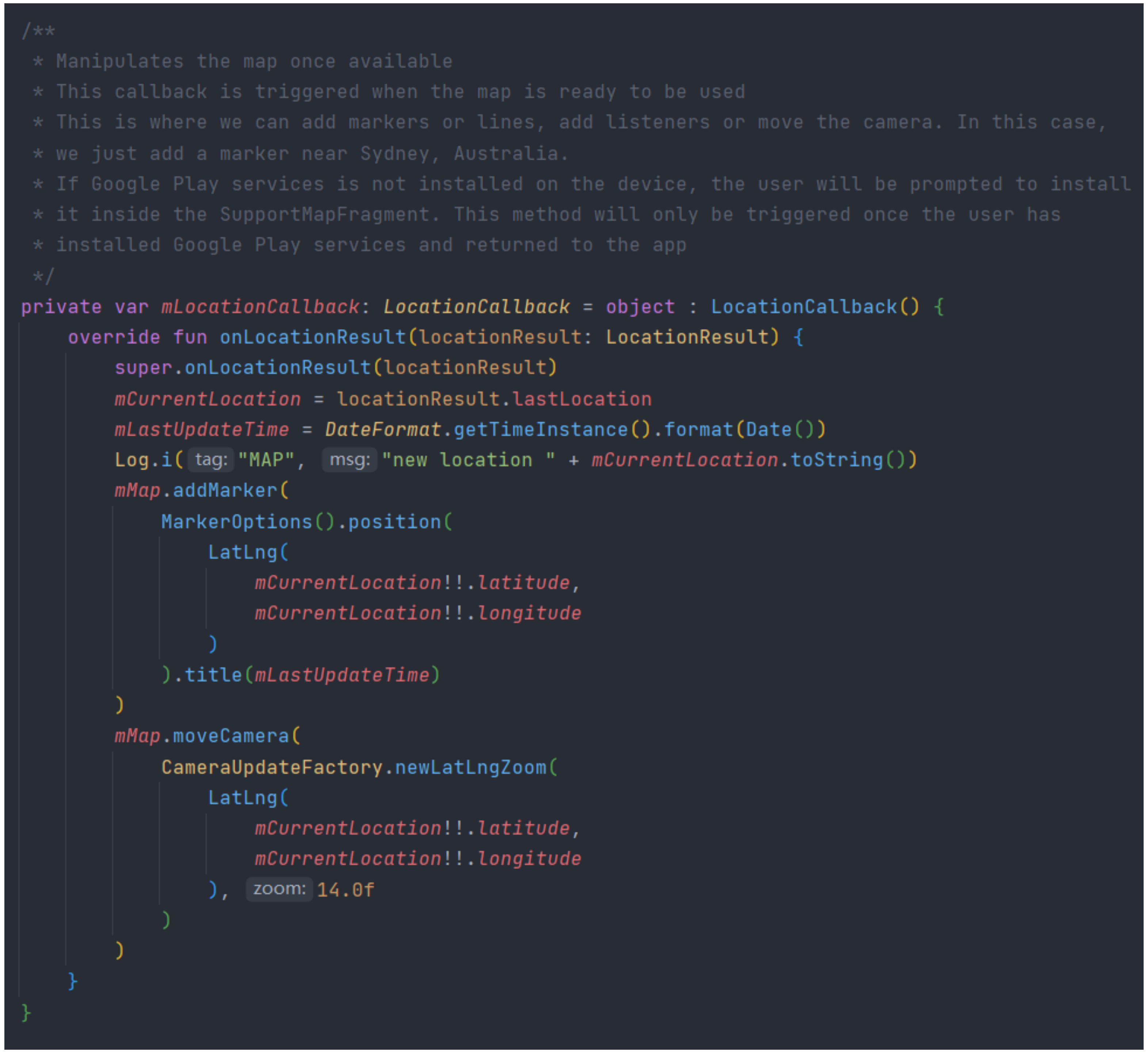
Showing location

- Your map will be positioned on Sydney, Australia
- Now we want to make visualise a location every time it is available

your map will be

created when

this is called



```
/**
 * Manipulates the map once available
 * This callback is triggered when the map is ready to be used
 * This is where we can add markers or lines, add listeners or move the camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app
 */
private var mLocationCallback: LocationCallback = object : LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult) {
        super.onLocationResult(locationResult)
        mCurrentLocation = locationResult.lastLocation
        mLastUpdateTime = DateFormat.getTimeInstance().format(Date())
        Log.i( tag: "MAP",  msg: "new location " + mCurrentLocation.toString())
        mMap.addMarker(
            MarkerOptions().position(
                LatLng(
                    mCurrentLocation!!.latitude,
                    mCurrentLocation!!.longitude
                )
            ).title(mLastUpdateTime)
        )
        mMap.moveCamera(
            CameraUpdateFactory.newLatLngZoom(
                LatLng(
                    mCurrentLocation!!.latitude,
                    mCurrentLocation!!.longitude
                ),  zoom: 14.0f
            )
        )
    }
}
```

© Po Yang, University of Sheffield

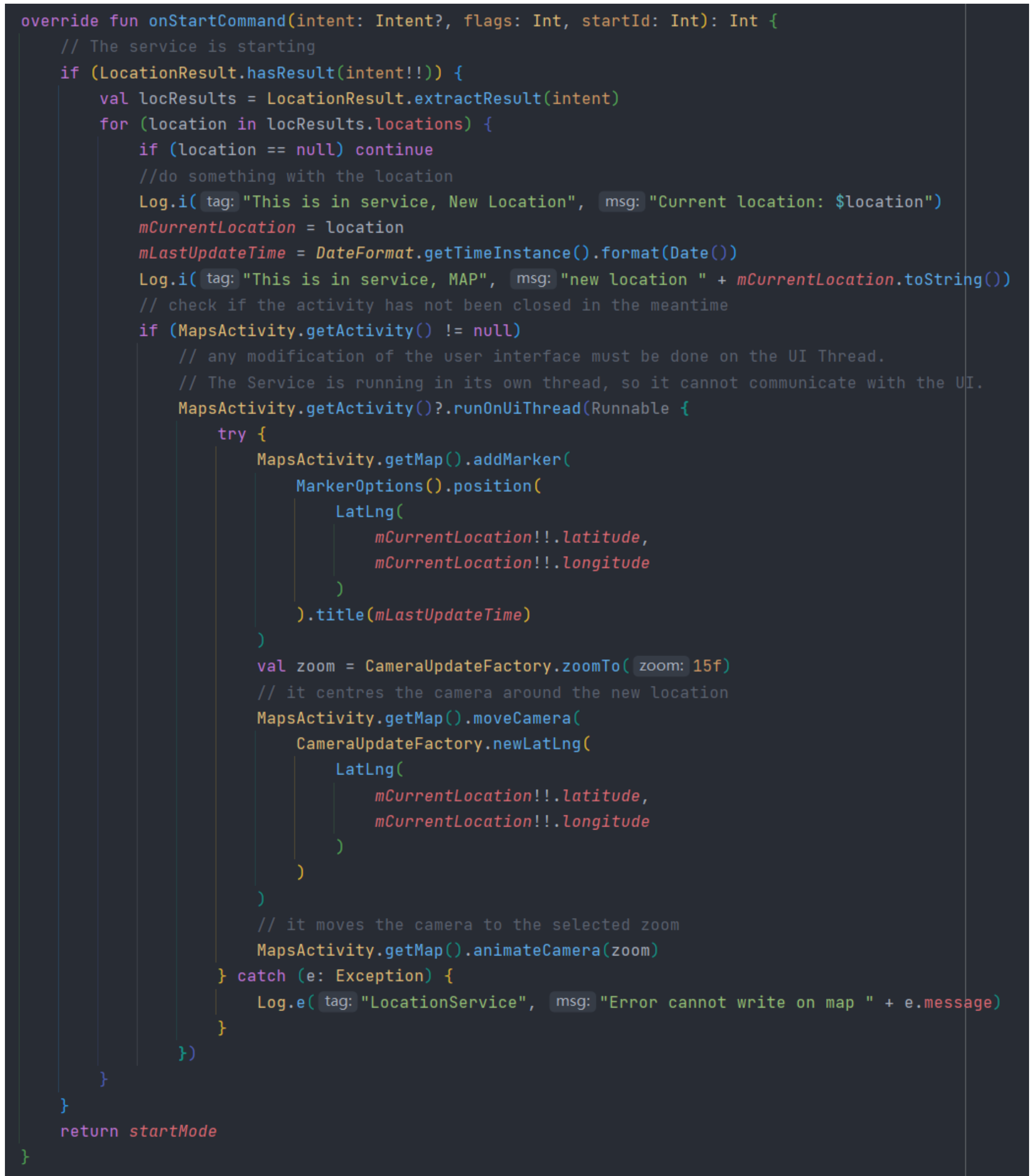
Add marker to the map

- Currently when a new location is received, we just print it in the logical
- Let's add a new marker on the map instead

© Po Yang, University of Sheffield

Visualising the position

change the onStartCommand method defined in the subclass Service



```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
    // The service is starting
    if (LocationResult.hasResult(intent!!)) {
        val locResults = LocationResult.extractResult(intent)
        for (location in locResults.locations) {
            if (location == null) continue
            //do something with the location
            Log.i( tag: "This is in service, New Location",  msg: "Current location: $location")
            mCurrentLocation = location
            mLastUpdateTime = DateFormat.getTimeInstance().format(Date())
            Log.i( tag: "This is in service, MAP",  msg: "new location " + mCurrentLocation.toString())
            // check if the activity has not been closed in the meantime
            if (MapsActivity.getActivity() != null)
                // any modification of the user interface must be done on the UI Thread.
                // The Service is running in its own thread, so it cannot communicate with the UI.
                MapsActivity.getActivity()?.runOnUiThread(Runnable {
                    try {
                        MapsActivity.getMap().addMarker(
                            MarkerOptions().position(
                                LatLng(
                                    mCurrentLocation!!.latitude,
                                    mCurrentLocation!!.longitude
                                )
                            ).title(mLastUpdateTime)
                        )
                        val zoom = CameraUpdateFactory.zoomTo( zoom: 15f)
                        // it centres the camera around the new location
                        MapsActivity.getMap().moveCamera(
                            CameraUpdateFactory.newLatLng(
                                LatLng(
                                    mCurrentLocation!!.latitude,
                                    mCurrentLocation!!.longitude
                                )
                            )
                        )
                        // it moves the camera to the selected zoom
                        MapsActivity.getMap().animateCamera(zoom)
                    } catch (e: Exception) {
                        Log.e( tag: "LocationService",  msg: "Error cannot write on map " + e.message)
                    }
                })
        }
    }
    return startMode
}
```

© Po Yang, University of Sheffield

- where *getActivity*() refers to

```
static MapsActivity activity;
```

- which is:
  - defined in the main class
  - and set in the on resume as

```
setActivity(this);
```

```
And getMaps returns nMaps which also becomes static
```

© Po Yang, University of Sheffield

Or MUCH better

- Use live Data!!!

© Po Yang, University of Sheffield