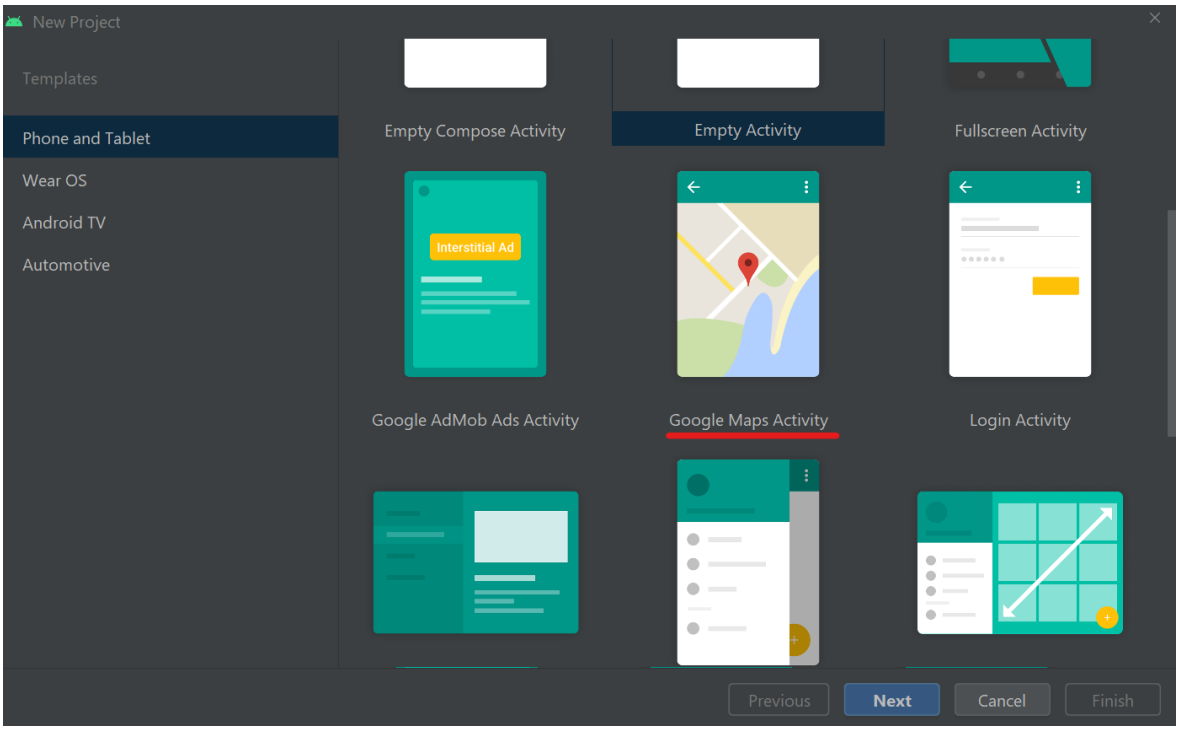
CSE2MAD LAB 7 – GOOGLE MAP ACTIVITY – TRACE MY STEPS

AIMS

To learn how Google Map Fragments can be used in activities and how we can use location detection to update and overlay a map with information.

1.) Create a new project with a “Google Maps Activity”



2.) In the Android Manifest request permission for Fine and Coarse Location

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

3.) Go to the res/values folder and open the google_maps_api.xml file. Follow the instructions to obtain and insert an API key. You will need to use a google account to login to google cloud.

4.) At this point you will notice that we need to have the google services built alongside our code. Go to the ‘Gradle Scripts’ section and insert into the app module dependencies

```
implementation 'com.google.android.gms:play-services-location:18.0.0'
```

Note: This is dependent on your system. Inserting this into our build scripts allows your app to have access to the resources provided by play-services (Maps and Location).

5.) Let's declare some member variables, mMap and binding should already be there.

```
//Google maps variables
private GoogleMap mMap;
private ActivityMapsBinding binding;

//Required permissions array
final String[] PERMISSIONS = {Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION};

//Variable for debugging in logcat
String TAG = "MapAct";

//Location variables
private FusedLocationProviderClient mFusedLocationClient;
private LocationCallback locationCallback;
private LocationRequest mLocationRequest;
private ArrayList<LatLng> points;

//Draw on map
Polyline line;
private static final float SMALLEST_DISPLACEMENT = 0.5F; //half a meter
```

6.) The activity has just like any other, an onCreate method, we have to initialize our FusedLocationClient and create a location request. Add this to the code already there. Here we are doing typical onCreate type actions. Initialising the locationClient, obtaining an initial location, moving the map to that location, initialising an array of LatLng points to store our path as we move, setting up a callback for location updates. Also note the permissions request, we must check before attempting to obtain any information data that the user has allowed us to access their location. This is the same as in Lab3, where we also required multiple real time permissions.

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);

   binding = ActivityMapsBinding.inflate(getLayoutInflater());
   setContentView(binding.getRoot());

   // Obtain the SupportMapFragment and get notified when the map is ready to be used.
   SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
         .findFragmentById(R.id.map);
   mapFragment.getMapAsync(this);

   //initalise fused location client
   mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

   //get the initial location
   if (hasPermission()) {
      Log.d(TAG, "App has required permissions");
```

```
        getLastLocation(); //get start location
        createLocationRequest(); // set up the location tracking

    } else {
        Log.d(TAG, "App does not have required permissions, asking now");
        askPermissions();
    }

    // to hold our location readings
    points = new ArrayList<LatLng>();

    // This is run EVERYTIME a location update is recieved
    locationCallback = new LocationCallback() {
        @Override
        public void onLocationResult(LocationResult locationResult) {
            if (locationResult == null) {
                return;
            }
            for (Location location : locationResult.getLocations()) {
                if (location != null) {
                    double latitude = location.getLatitude();
                    double longitude = location.getLongitude();
                    LatLng latLng = new LatLng(latitude, longitude);
                    Log.d(TAG, "Adding location to points ArrayList");
                    points.add(latLng);
                    //redrawing the line with the new location
                    redrawLine();
                    //moving the map to the new location
                    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
                }
            }
        }
    };
}
```

7.) Now we must implement the missing methods OUTSIDE the onCreate, firstly the createLocationRequest that will trigger the location callbacks. Here, we set the request to be every 2 seconds, no shorter than 1seconds, of 0.5m minimum displacement and of high accuracy.

```
//for more info https://developer.android.com/training/location/request-updates
protected void createLocationRequest() {
    mLocationRequest = LocationRequest.create()
            .setInterval(2000)
            .setFastestInterval(1000)
            .setSmallestDisplacement(SMALLEST_DISPLACEMENT)
            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
            .addLocationRequest(mLocationRequest);
}
```

8.) Next we will need the permission methods we discussed previously mentioned. The hasPermissions() method checks if permissions are already granted, the askPermissions() method as the name may suggests triggers permissions requests by launching the multiplePermissionActivityResultLauncher.

```
//helper function to check permission status
private boolean hasPermissions() {
    boolean permissionStatus = true;
    for (String permission : PERMISSIONS) {
        if (ActivityCompat.checkSelfPermission(this, permission) == PackageManager.PERMISSION_GRANTED)
{
            Log.d(TAG, "Permission is granted: " + permission);
        } else {
            Log.d(TAG, "Permission is not granted: " + permission);
            permissionStatus = false;
        }
    }
    return permissionStatus;
}

//helper function to ask user permissions
private void askPermissions() {
    if (!hasPermissions()) {
        Log.d(TAG, "Launching multiple contract permission launcher for ALL required permissions");
        multiplePermissionActivityResultLauncher.launch(PERMISSIONS);
    } else {
        Log.d(TAG, "All permissions are already granted");
    }
}

//Result launcher for permissions
private final ActivityResultLauncher<String[]> multiplePermissionActivityResultLauncher =
        registerForActivityResult(new ActivityResultContracts.RequestMultiplePermissions(), isGranted -> {
            Log.d(TAG, "Launcher result: " + isGranted.toString());
            //permissions are granted lets get to work!
```

```
        getLastLocation(); //get start location
        createLocationRequest(); // set up the location tracking
        if (isGranted.containsValue(false)) {
            Log.d(TAG, "At least one of the permissions was not granted, please enable permissions to ensure
app functionality");
        }
    });
```

9. Now that all requisite permissions have been covered we need to insert the function to het the last known location. If you are wondering about the SuppressLint annotation, we have already checked permissions so this is supressing a warning to check permissions.

```
@SuppressLint("MissingPermission")
private void getLastLocation() {
    mFusedLocationClient.getLastLocation()
        .addOnSuccessListener(this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Got last known location. In some rare situations this can be null.
                if (location != null) {
                    Log.d(TAG, "Location detected " + location.getLatitude() + " " + location.getLongitude());
                    LatLng loc = new LatLng(location.getLatitude(), location.getLongitude());
                    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, 19));
                    createLocationRequest();
                } else {
                    Toast.makeText(getApplicationContext(), "no_location_detected",
Toast.LENGTH_SHORT).show();
                    Log.d(TAG, "Location obj is null");
                }
            }
        });
}
```

10. Next we create a method that will draw the line that will be tracing your path using the Arraylist of LatLang objects.

```
private void redrawLine() {
    mMap.clear(); //clears all overlays
    PolylineOptions options = new
        PolylineOptions().width(5).color(Color.BLUE).geodesic(true);
    for (int i = 0; i < points.size(); i++) {
        LatLng point = points.get(i);
        options.add(point);
    }
    Log.d(TAG, "Adding polyline");
    line = mMap.addPolyline(options); //adds Polyline }
}
```

11. As location services use a large amount of power we must also add in methods with the Android Activity Lifecycle in mind. These ensure location requests are only running while the app is open and being used.

```
@Override
protected void onResume() {
    super.onResume();
    startLocationUpdates();
}

@Override
protected void onPause() {
    super.onPause();
    stopLocationUpdates();
}

@Override
protected void onStop() {
    super.onStop();
    stopLocationUpdates();
}


@SuppressLint("MissingPermission")
private void startLocationUpdates() {
    mFusedLocationClient.requestLocationUpdates(mLocationRequest,
            locationCallback,
            Looper.getMainLooper());
}

private void stopLocationUpdates() {
    mFusedLocationClient.removeLocationUpdates(locationCallback);
}
```

12. And finally we add the code below to the onMapReady() method, creating the tracking location request as soon as the map is ready. Ensure your onMapReady() methods looks like this, else you will get a marker to Sydney

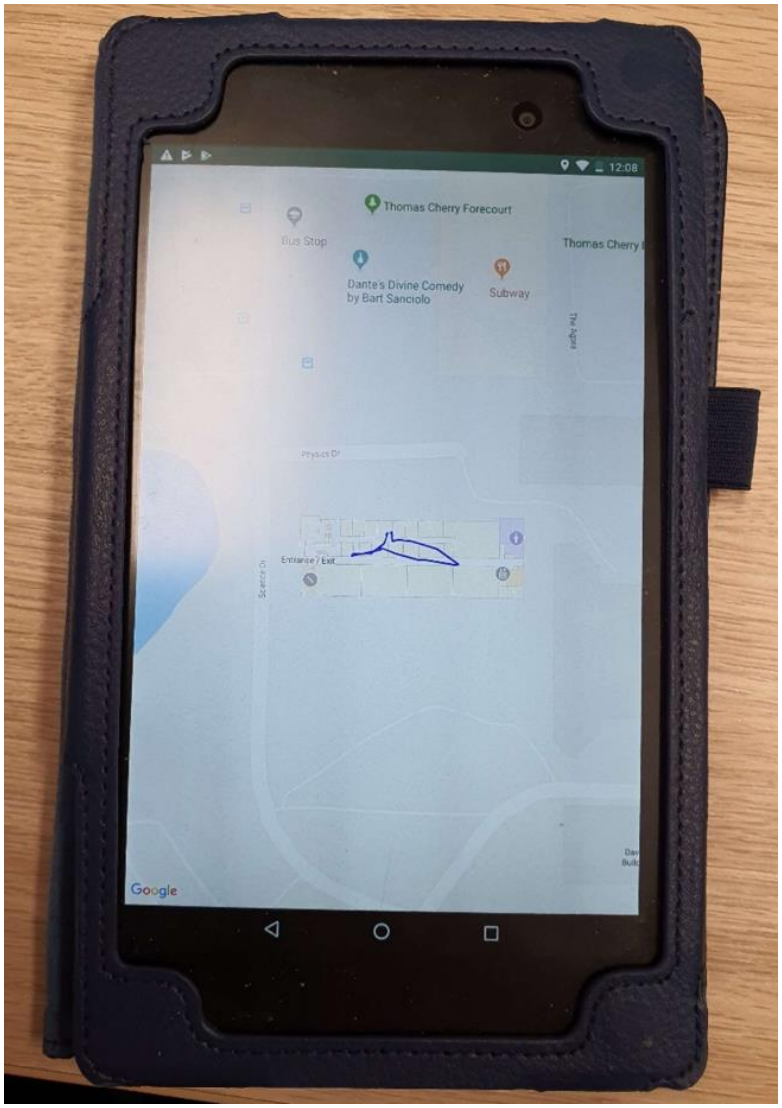
```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    if (!mFusedLocationClient.getLastLocation().isSuccessful()) {
        Log.d(TAG, "Setting up location tracking");
        createLocationRequest(); // set up the location tracking
    }
}
```

Problems?

If you run the app and don't see a map, it could be any of the following;

- the API key is invalid
- you have no network connectivity
- you have not given the app permissions.
- If your lastlocation object is null you can open the onboard maps app to generate a last location. Although there is another way, how do you think you could use your location request if the getlastlocation failed?

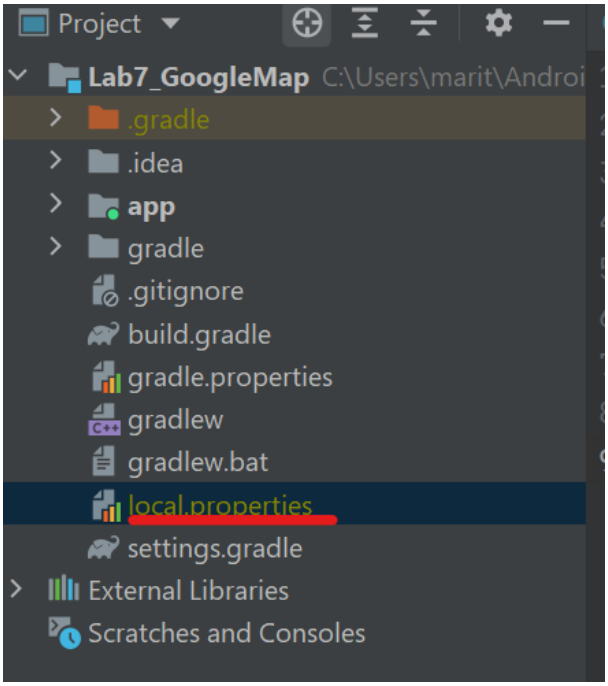


CHALLENGE TASK

As many of you will be using Github or other Version control systems it is important you never upload your API key as it will be viewable by the public. There is a simple solution, using the gradle secrets plugin.

Follow the instructions in the repository below using the Groovy Gradle code to protect your API key by storing it in your local.properties file.

https://github.com/google/secrets-gradle-plugin


Project
Lab7_GoogleMap C:\Users\marit\Androi
.gradle
.idea
app
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries
Scratches and Consoles