Integration and Switching Between Google Maps SDK and ODRD Consumer SDK

Overview

This documentation explains how to integrate both Google Maps SDK and ODRD Consumer SDK into the same Java project, allowing you to switch between the two map providers as needed. The implementation involves creating wrapper classes for each map SDK and a common interface to ensure both map implementations can be used interchangeably.

Components

Interface: MapWrapper

The MapWrapper interface defines the methods that both map implementations must provide (includes Map and Map UI functionalities)

Abstract Class: MapsGeneralFunctions

This class will contain the common functionalities required for Consumer SDK and Google Maps SDK. As a result, this class won't contain any instances of Google Maps or Consumer Maps.

Class: GoogleMapWrapper

This class wraps the Google Maps SDK functionality, implementing the MapWrapper interface and inheriting MapsGeneralFunctions.

Class: ConsumerGMapWrapper

This class wraps the ODRD Consumer SDK functionality, implementing the MapWrapper interface and inheriting MapsGeneralFunctions.

Integration

1. Create an interface that will contain all the common functions required for Google Maps and Consumer Map

```
public interface MapWrapper {

    void setMapUI();
    void animateCamera(@NonNull CameraUpdate var1, @NonNull GoogleMap.CancelableCallback var2);
    void setOnMarkerClickListener(@Nullable GoogleMap.OnMarkerClickListener var1);
    void setOnMapClickListener(@Nullable GoogleMap.OnMapClickListener var1);

    ...
    ...
}
```

2. Implement the MapWrapper interface in GoogleMapWrapper and ConsumerMapWrapper classes.

    a. GoogleMapWrapper class

```
public class GoogleMapWrapper extends MapsGeneralFunctions implements MapWrapper {

    private final GoogleMap googleMap;
    MainActivity mainActivity;
    static GeneralFunctions generalFunc;
    SupportMapFragment gMapFragment;

    public GoogleMapWrapper(GoogleMap googleMap, MainActivity mainActivity, SupportMapFragment gMapFragment) {
        this.googleMap = googleMap;
        this.mainActivity = mainActivity;
        this.gMapFragment = gMapFragment;
        generalFunc = mainActivity.generalFunc;
    }
}
```

The constructor takes three arguments:

googleMap: The GoogleMap object that this wrapper will manage.

mainActivity: A reference to the MainActivity instance.

gMapFragment: A reference to the SupportMapFragment instance.

b. ConsumerWrapper class

```
public class ConsumerGMapWrapper extends MapsGeneralFunctions implements MapWrapper {

    private final ConsumerGoogleMap consumerGMap;
    MainActivity mainActivity;
    ConsumerMapView consumerMapView;
    static GeneralFunctions generalFunc;

    public ConsumerGMapWrapper(ConsumerGoogleMap consumerGMap, MainActivity mainActivity, ConsumerMapView consumerMapView) {
        this.consumerGMap = consumerGMap;
        this.mainActivity = mainActivity;
        this.consumerMapView = consumerMapView;
        generalFunc = mainActivity.generalFunc;
    }
}
```

The constructor takes three arguments:

consumerMap: The GoogleMap object that this wrapper will manage.

mainActivity: A reference to the MainActivity instance.

consimerMapView: A reference to the ConsumerMapView instance.

3. Add a condition to check whether to load Google Map SDK or ODRD SDK

Note: Replace “isODRDMap” with your condition.

```
public class MainActivity extends BaseActivity{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (isODRDMap) {
            initializeODRDSdk();
        } else {
            loadMapDynamicAsync();
        }
    }
}
```

4. Initialize the mapWrapper. If the map to be loaded is:

    a. GoogleMap: - When the onMapReady function is called, initialize the mapWrapper with the instance of GoogleMapWrapper class. Also, the google map view is initialized here.

```
private void loadMapDynamicAsync() {
    gMapFragment = (SupportMapFragment) getSupportFragmentManager
        ().findFragmentById(R.id.mapV2);
    gMapFragment.getMapAsync(MainActivity.this);
    gMapFragment.getMapAsync(new OnMapReadyCallback() {
            @Override
            public void onMapReady(@NonNull GoogleMap googleMap) {
                gMap = googleMap;
                if (mapWrapper == null){
                    MainActivity.this.mapWrapper = new GoogleMapWrapper(googleMap
                        , MainActivity.this, gMapFragment);
                }
                staticMapImg.setVisibility(View.GONE);
            }
        });
}
```

b. ConsumserMap: - When the onConsumerMapReady function is called, initialize the mapWrapper with the instance of ConsumerMapWrapper class. Also the consumer map view is initialized here.

```
private void initializeODRDSdk() {
    consumerMapView = findViewById(R.id.consumer_map_view);
    consumerMapView.getConsumerGoogleMapAsync(
        new ConsumerGoogleMap.ConsumerMapReadyCallback() {

        @Override
        public void onConsumerMapReady(@NonNull ConsumerGoogleMap consumerGoogleMap) {
            mapWrapper = new ConsumerGMapWrapper(consumerGoogleMap, MainActivity.this, consumerMapView);
        }
    }
```

5. Replace the gMap, getMap(), and map instances with mapWrapper.

Before

```
1  //gMap and getMap() is an instance of GoogleMap
2  gMap.setOnCameraMoveStartListener();
3  getMap().setPadding(0, 0, 0, 0, 0));
4
5  //map is an instance of SupportMapFragment
6  map.setVisibility(View.VISIBLE);
7
8
```

After

```
1  //mapWrapper is an instance of GoogleMapWrapper class or
2  //ConsumerMapWrapper class depending upon the satisfied condition
3  mapWrapper.setOnCameraMoveStartListener();
4  mapWrapper.setPadding(0, 0, 0, 0, 0));
5
6  mapWrapper.setVisibility(View.VISIBLE);
7
8
```

6. How to use MapWrapper in Fragments and other classes?

```
11 usages  new *
public MapWrapper getMapWrapper(){
    return this.mapWrapper;
}
```

a. A function called getMapWrapper is created in the MainActivity which will return the instance of MapWrapper.
b. A reference of MainActivity has already been used throughout the Fragments and other functions
c. So we can call the getMapWrapper function like this;

```
mainAct.getMapWrapper().moveCamera(CameraUpdateFactory.newLatLngBounds(builder.build(),  padding: 10));
```

7. MapsGeneralFunctions

MapsGeneralFunctions is an abstract class that contains all the common functions required for both maps. This class will be inherited by GoogleMapWrapper and ConsumerGMapWrapper classes.

The End