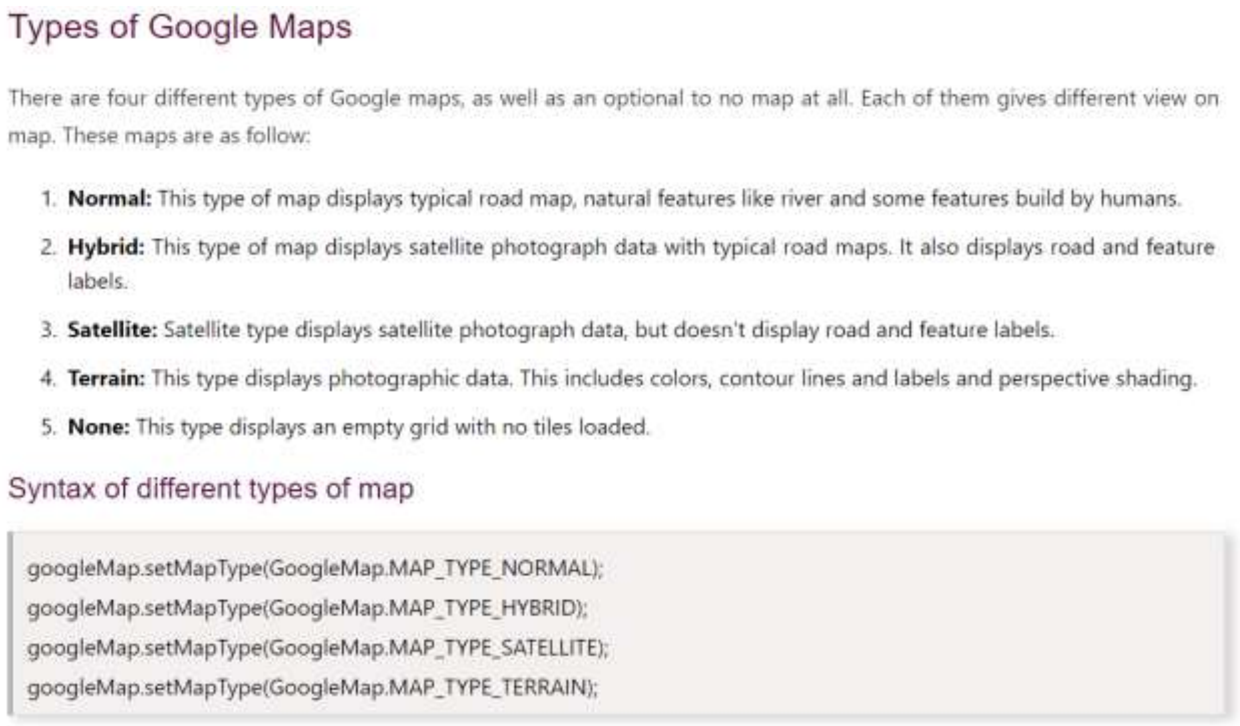
Practical No: 31

Practical Related Questions

1. **List the names of map type and write the syntax to change it.**



Types of Google Maps

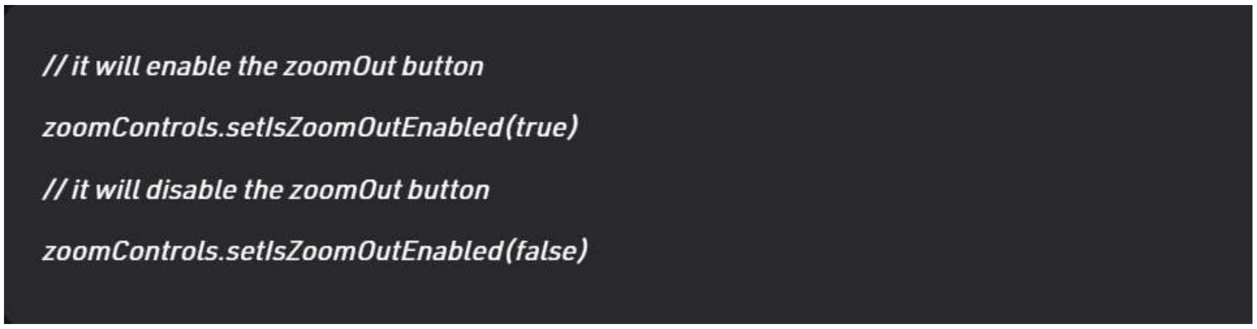
There are four different types of Google maps, as well as an optional to no map at all. Each of them gives different view on map. These maps are as follow:

1. **Normal:** This type of map displays typical road map, natural features like river and some features build by humans.
2. **Hybrid:** This type of map displays satellite photograph data with typical road maps. It also displays road and feature labels.
3. **Satellite:** Satellite type displays satellite photograph data, but doesn't display road and feature labels.
4. **Terrain:** This type displays photographic data. This includes colors, contour lines and labels and perspective shading.
5. **None:** This type displays an empty grid with no tiles loaded.

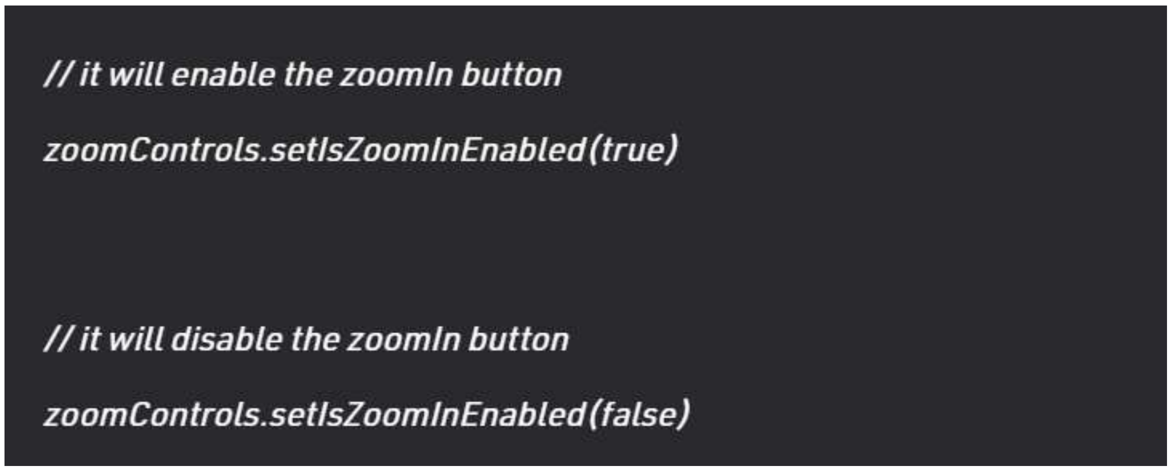
Syntax of different types of map

```
googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

2. **Name the methods used to enable and disable zoom feature.**



```
// it will enable the zoomOut button
zoomControls.setIsZoomOutEnabled(true)
// it will disable the zoomOut button
zoomControls.setIsZoomOutEnabled(false)
```



```
// it will enable the zoomIn button
zoomControls.setIsZoomInEnabled(true)



// it will disable the zoomIn button
zoomControls.setIsZoomInEnabled(false)
```

Practical No: 31

Exercise

MapsActivity.java

```
package com.example.mylocation;

import androidx.fragment.app.FragmentActivity;

import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
LocationListener {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be
used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync((OnMapReadyCallback) this);

    }
    @Override
    public void onLocationChanged(Location location) {
        TextView locationTv = (TextView) findViewById(R.id.latlongLocation);
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        LatLng latLng = new LatLng(latitude, longitude);
        mMap.addMarker(new MarkerOptions().position(latLng));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
        locationTv.setText("Latitude:" + latitude + ", Longitude:" + longitude);
    }

    @Override
    public void onProviderDisabled(String provider) {
```

Practical No: 31

```
        // TODO Auto-generated method stub
    }

    @Override
    public void onProviderEnabled(String provider) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // TODO Auto-generated method stub
    }

    /**
     * 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.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        mMap.setMyLocationEnabled(true);
        LocationManager locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
        Criteria criteria = new Criteria();
        String bestProvider = locationManager.getBestProvider(criteria, true);
        Location location = locationManager.getLastKnownLocation(bestProvider);
        if (location != null) {
            onLocationChanged(location);
        }
        locationManager.requestLocationUpdates(bestProvider, 20000, 0, this);
    }

    private boolean isGooglePlayServicesAvailable() {
        int status = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
        if (ConnectionResult.SUCCESS == status) {
            return true;
        } else {
            GooglePlayServicesUtil.getErrorDialog(status, this, 0).show();
            return false;
        }
    }
}
```

Practical No: 31

activity_maps.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" ></fragment>

    <TextView
        android:id="@+id/latlongLocation"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom"
        android:layout_alignParentBottom="true"
        android:background="#ff058fff"
        android:paddingTop="5dp"
        android:paddingBottom="5dp"
        android:textColor="#ffffffff"
        android:paddingLeft="5dp"
        android:paddingRight="5dp" />
</RelativeLayout>
```

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mylocation">

    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but you must specify either coarse or fine
         location permissions for the "MyLocation" functionality.
    -->
    <permission
        android:name="com.javapapers.currentlocationinmap.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission
android:name="com.javapapers.currentlocationinmap.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Practical No: 31

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyLocation">

        <!--
             The API key for Google Maps-based APIs is defined as a string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the
APK.
             You need a different API key for each encryption key, including the
release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets in src/debug/
and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```