Unit 05: deepankar

✓ Bluetooth Adapter class   ✓ Google maps   ✓ Methods of Google Map
✓ Paired Devices   ✓ Types of Google maps ✓ Need of Callback methods
✓ Enable/Disable Devices   ✓ Geocoder class

BluetoothAdapter class   (public final class BluetoothAdapter extends Object)

It represents the local device Bluetooth adapter. This class lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a BluetoothDevice and create a BluetoothServerSocket using a known MAC address to listen for the connection requests from other devices and start a scan for Bluetooth LE devices.

Paired Devices
```
BluetoothManager btm = getSystemService (BluetoothManager. class);
BluetoothAdapter bta = btm. getAdapter () ;
```

To create a connection between two devices you must implement both server side and the client side mechanisms, because one device must open a server socket and the other one must initiate the connection using server device MAC address.

Connect as Server
→ get a BluetoothServerSocket
→ listen for connection requests by calling accept()
→ close() to close the socket
```
BluetoothServerSocket bss ;
bss = bta. listenUsingRfcommWithServiceRecord (name, MY_UUID);
BluetoothSocket = bss. accept();
bss. close ();
```

Connect as Client
→ using BluetoothDevice get a BluetoothSocket.
→ initiate connection by calling connect().
```
BluetoothDevice bd = current device ;
BluetoothSocket bs = bd. createRfcommSocketToServiceRecord (My_UUID);
bs. connect () ;
bs. close ();
```
`<uses-permission android:name = "android.permission.BLUETOOTH"/>`
`<uses-permission android:name = "android.permission.BLUETOOTH_ADMIN"/>`

Enable/Disable Devices
① The first step is adding the Bluetooth permissions to your manifest file in order to use the API's.
② Get the Bluetooth Adapter.
③ Enable/Disable Bluetooth.
```
BluetoothManager btm = getSystemService (BluetoothManager. class) ;
BluetoothAdapter ba = btm. getAdapter(); // null if device doesn't support BT
if ( !ba. isEnabled()) {
    Intent i = new Intent ( BluetoothAdapter. ACTION_REQUEST_ENABLE);
    startActivityForResult (i, REQUEST_ENABLE_BT);
}
BluetoothAdapter ba = BluetoothAdapter. getDefaultAdapter();
if ( ba. isEnabled()) {
    ba. disable ();
}
```

Google Maps

Android allows us to integrate google maps in our application. You can show any location on the map, or can show different routes on the map etc. You can also customize the maps according to your choices.

You need to add map fragment to your xml layout file.
```
<fragment
    android : id = "@+ Id/ map"
    android : name = "com.google.android.gms.maps.MapFragment"
    android : layout_width = "match_parent"
    android : layout_height = "wrap_content"
/>
```

Also you need to add some permissions along with Google Map API key in AndroidManifest.xml
```
<!-- Permissions -->
<uses-permission   android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission   android:name="android.permission.INTERNET"/>
<uses-permission   android:name= "com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission   android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- Google MAP API key-->
<meta-data
    android:name = "com.google.android.maps.v2.API-KEY"
    android:value = "  key_value  " />
```

```
@ Override
public void onMapReady ( GoogleMap gm) {
    GoogleMap gmap = gm;

    SupportMapFragment smf = (SupportMapFragment) getSupportFragmentManager() .
                     findFragmentById ( R. id . map) ;
    smf. getMapAsync (this);
    gmap. addMarker (new MarkerOptions (). position (
                         new LatLng ( latitude , longitude)
                    )) ;
}
```

Types of Google Maps

There are 4 types of maps available in Maps JavaScript API :
① Roadmap → displays the default roadmap view. This is the default map type.
② Satellite → displays the Google Earth satellite images.
③ Hybrid → displays the mixture of normal and satellite views
④ Terrain → displays a physical map based on terrain information.
```
var map = new google. maps. Map ();
map. setMapTypeID ('terrain');
         → 'roadmap'
         → 'satellite'
         → 'terrain'
```
} MAPS JS API

Google Maps in Android

Android provides facility to integrate Google Maps in our application. Aside from the various features provided by the maps JS API   GoogleMap gm ;
                          → location → navigation
we can customize google Map according to our requirements.

Google Map Types in Android
① Normal → roadmap, natural features like rivers   gm. set MapType (GoogleMap. MAP_TYPE_NORMAL);
② Hybrid → roadmap + terrain + satellite   gm. set MapType (GoogleMap. MAP_TYPE_HYBRID);
③ Satellite → satellite = google Earth   gm. set MapType (GoogleMap. MAP_TYPE_SATELLITE);
④ Terrain → photographic data   gm. set MapType (GoogleMap. MAP_TYPE_TERRAIN);
⑤ None → empty grid, no tiles loaded (optional)   gm. set MapType (GoogleMap. MAP_TYPE_NONE);

Methods of Google Map
① addCircle (CircleOptions cm) → add circle to map
② addPolygon (PolygonOptions p) → add polygon to map
③ addTileOverlay (TileOverlayOptions too) → add tile overlay to map
④ animateCamera (CameraUpdate u) → moves the map according to the update with an animation
⑤ clear () → removes everything from the map
⑥ getMyLocation () → returns currently displayed user's location
⑦ moveCamera (CameraUpdate u) → reposition the camera according to instructions in update
⑧ setTrafficEnabled (Boolean b) → set traffic layer on/off
⑨ snapshot (GoogleMap. SnapshotReadyCallback src) → takes snapshot of the map
⑩ stopAnimation () → stops the camera animation, if any in progress.

Geocoder class

Searching location in google MAP API is done using the geocoder class. Geocoder class is used to handle geocoding and reverse geocoding.

Geocoding → The process in which street address is converted into a coordinate (latitude, longitude).
Reverse Geocoding → The process in which coordinate is converted into an address.
```
Geocoder gc;
```
some methods of geocoder class are :
① List <Address> getFromLocation (double latitude , double longitude , int maxResults) ;
② List <Address> getFromLocationName (String loc, int results, left lat, leftLong, rightLat, rightLong) ;
③ List <Address> getFromLocationName (String loc, int results) ;
④ static boolean isPresent () → if the methods are implemented, return true
```
List <Address> al = gc. getFromLocationName ( "Delhi", 1);
Address a = al. get(0);
LatLng llm = new LatLng (a. getLatitude() , a. getLongitude());
```

public interface OnMapReadyCallback → callback interface for when map is ready to be used.

To get the googleMap object in our class, we need to implement OnMapReadyCallback interface and override its onMapReady() method. Once an instance of the fragment is set on the MapFragment or MapView, the function onMapReady (gm) is triggered and provides non-null instance of googleMap.

```
public class Activity implements OnMapReadyCallback {
    private GoogleMap mgm ;                    → a non-null instance of the GoogleMap
                                                 associated with the MapFragment
    @Override                                  or MapView that defined
    public void onMapReady ( GoogleMap gm) {   the fragment
        mgm = gm ;
    }
}
```
→ if Google Play Services is not installed, user will be prompted to install it. this method will only trigger when google play services is installed