



# SNS COLLEGE OF TECHNOLOGY



(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)  
Approved by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

## Department of MCA

### Topic: Location Awareness

**COURSE**

**19CAT701**

**Mobile Application  
Development**

**UNIT - IV**

**SPRUCING  
UP MOBILE  
APPS**

**CLASS**

**III Semester /  
II MCA**



# Location Awareness

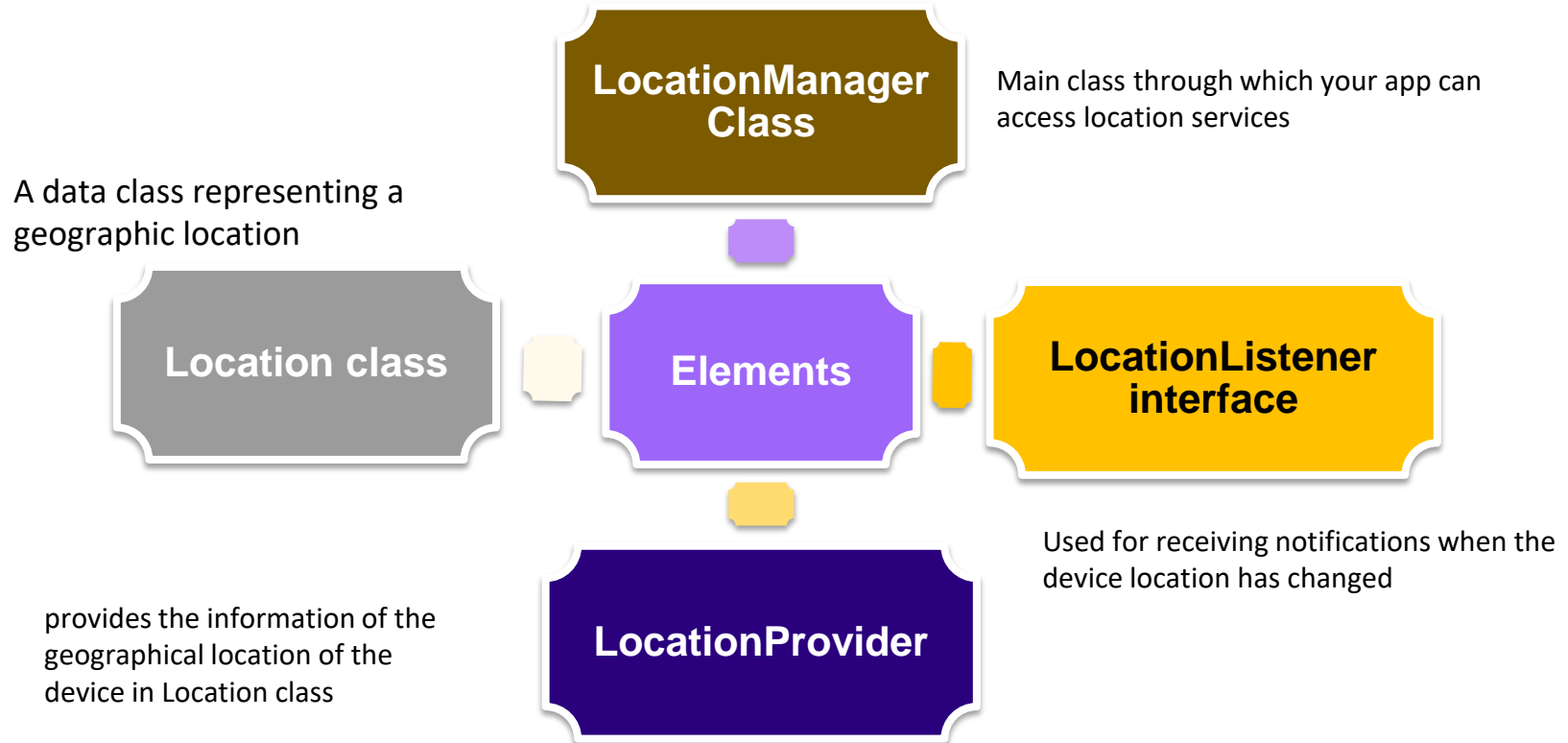


- ❑ Android makes use of information from GPS and WiFi networks to get the location of the device on this Earth
- ❑ To build location-aware applications with the help of Google Play services
- ❑ Google Play services facilitates adding location awareness to your app with automated location tracking, geofencing, and activity recognition
- ❑ Android provides Location Framework contains classes and interfaces to implement the location feature in our app





# Components of Location object





# Request location permissions



- Apps that use location services must request location permissions in order to protect user privacy
- Multiple permissions related to location
- Which permissions you request, and how you request them, depend on the location requirements for your app's use case are matters
- Types of location access**
  - Each permission has a combination of the following characteristics:
    - **Category:** Either foreground location or background location
    - **Accuracy:** Either precise location or approximate location



# Foreground location



- ❑ If your app contains a feature that shares or receives location information only once, or for a defined amount of time, then that feature requires foreground location access
- ❑ App accesses the device's current location in one of the following situations
  - An App's activity is visible
  - Your app is running a foreground service
- ❑ It is recommended that you declare a foreground service type of location
- ❑ Permission can set like

```
<manifest ... >  
<!-- Always include this permission -->  
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<!-- Include only if your app benefits from precise location access. -->  
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />  
</manifest>
```

```
<service  
  android:name="MyNavigationService"  
  android:foregroundServiceType="location" ... >  
  <!-- Any inner elements would go here. -->  
</service>
```

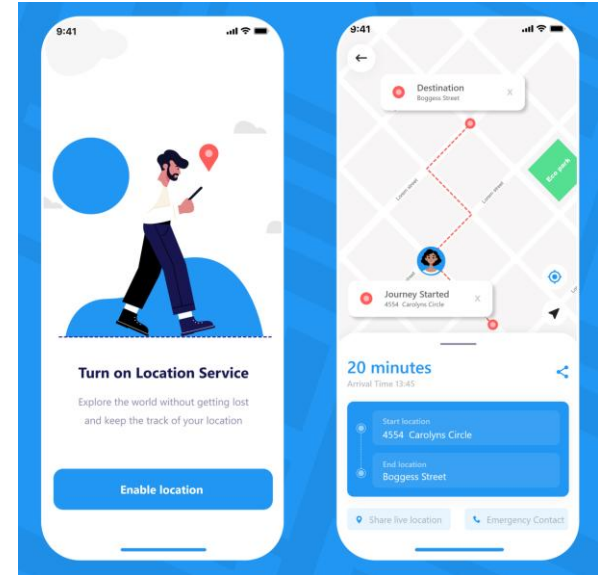


# Background location



- ❑ App requires background location access if a feature within the app constantly shares location with other users
- ❑ Permission can set like

```
<manifest ... >  
  <!-- Required only when requesting background location access  
  on Android 10 (API level 29) and higher. -->  
  <uses-permission  
  android:name="android.permission.ACCESS_BACKGROUND_LOC  
  ATION" />  
</manifest>
```





# Location Object



- ❑ Location object represents a geographic location which can consist of a latitude, longitude, time stamp, and other information such as bearing, altitude & velocity

<b>float distanceTo(Location Dest)</b>	Returns the approximate distance in meters between this location and the given location
<b>float getAccuracy()</b>	Get the estimated accuracy of this location, in meters
<b>double getAltitude()</b>	Get the altitude if available, in meters above sea level
<b>float getBearing()</b>	Get the bearing, in degrees
<b>double getLatitude()</b>	Get the latitude, in degrees
<b>double getLongitude()</b>	Get the longitude, in degree
<b>float getSpeed()</b>	Get the speed if it is available, in meters/second over ground
<b>boolean hasAccuracy()</b>	True if this location has an accuracy
<b>boolean hasAltitude().</b>	True if this location has an altitude
<b>boolean hasBearing()</b>	True if this location has a bearing



# Location Object



## ❑ Location object method continues

<b>boolean hasSpeed()</b>	True if this location has a speed
<b>void reset()</b>	Clears the contents of the location
<b>void setAccuracy(float accuracy)</b>	Set the estimated accuracy of this location, meters
<b>void setAltitude(double altitude)</b>	Set the altitude, in meters above sea level
<b>void setBearing(float bearing)</b>	Set the bearing, in degrees
<b>void setLatitude(double latitude)</b>	Set the latitude, in degrees
<b>void setLongitude(double longitude)</b>	Set the longitude, in degrees
<b>void setSpeed(float speed)</b>	Set the speed, in meters/second over ground
<b>String toString()</b>	Returns a string containing a concise, human-readable description of this object





# Access Current Location



- ❑ To get the current location,
  - create a location client which is LocationClient object
  - connect it to Location Services using connect() method, and then
  - call its getLastLocation() method, which returns the most recent location
- ❑ To have location based functionality in your activity, you will have to implement two interfaces
  - GooglePlayServicesClient.ConnectionCallbacks
  - GooglePlayServicesClient.OnConnectionFailedListener

<b>abstract void onConnected(Bundle connectionHint)</b>	called when location service is connected to the location client successfully. You will use <b>connect()</b> method to connect to the location client
<b>abstract void onDisconnected()</b>	called when the client is disconnected. You will use <b>disconnect()</b> method to disconnect from the location client
<b>abstract void onConnectionFailed(ConnectionResult result)</b>	called when there was an error connecting the client to the service



## Access updated Location



- ❑ To get the updated location,
  - implement **LocationListener** interface which has callback method

<b>abstract void onLocationChanged(Location location)</b>	used for receiving notifications from the LocationClient when the location has changed
---	--



# Location Quality of Service (QoS)



- ❑ **LocationRequest** object is used to request a quality of service (QoS) for location updates from the **LocationClient**

<b>setExpirationDuration(long millis)</b>	Set the duration of this request, in milliseconds
<b>setExpirationTime(long millis)</b>	Set the request expiration time, in millisecond since boot
<b>setFastestInterval(long millis)</b>	Explicitly set the fastest interval for location updates, in milliseconds
<b>setInterval(long millis)</b>	Set the desired interval for active location updates, in milliseconds
<b>setNumUpdates(int numUpdates)</b>	Set the number of location updates
<b>setPriority(int priority)</b>	Set the priority of the request

## ❑ **Displaying a Location Address**

- `Geocoder.getFromLocation()` method to get an address for a given latitude and longitude which need to be called from `doInBackground()` method of an `AsyncTask` class



# Maps



- ❑ While Location services determine the geographical coordinates of a location, Maps help plotting these coordinates in a much more comprehensible visual medium
- ❑ Maps to navigate to a desired location, enterprises have figured out their usage in various other scenarios
- ❑ Google Maps Android API to incorporate Maps and related functionalities
- ❑ Using API, we can include maps in our apps, mark places, and draw routes on it
- ❑ We need set permissions for `com.google.android.providers.gsf.permission.READ_GSERVICES`, `ACCESS_NETWORK_STATE`, `INTERNET`, and `WRITE_EXTERNAL_STORAGE` permissions





# Setting Up Maps



- ❑ To setup maps, ensure Google map services API installed
- ❑ Then we have to obtain a map key from Google APIs console, so that the app can access Google Maps servers through the Google Maps Android API
- ❑ In order to get a map key, we need to create a project in Google APIs console
- ❑ Once the project is created, we need to enable the Google Maps Android API
- ❑ Following this, we need to create a new Android key by providing, package name, and Secure Hash Algorithm 1 (SHA1) fingerprint of our app in the Credentials sub-section of APIs & auth section
- ❑ SHA1 fingerprint is a 40-digit hexadecimal number that represents the shorthand for RSA key required to authenticate our app, before publishing it on an app store



# Add Map



- ❑ It is generated using the keytool utility (JDK tool), and get stored inside a keystore – a database to store cryptographic keys and digital certificates

```
keytool -genkey -v -keystore <<fully_qualified_filename>>.keystore -  
alias <<aliasname>> -keyalg RSA -keysize 2048 -validity 10000
```

- ❑ To list the SHA1 fingerprint, we need to execute the following command

```
keytool -list -v -keystore <<fully_qualified_filename>>.keystore -  
alias <<aliasname>> -storepass <<password>> -keypass <<password>>
```

- ❑ outcome of the app registration is a map key displayed on Google APIs console. Once we get the map key, we have to add it to the app manifest file

```
<application>  
...  
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="<<Obtained map key>>"/>  
...  
</application>
```



## Access updated Location



- ❑ Once map is setup, it can be added to our app using MapView UI element /Map\_Frament
- ❑ Add the code into the layout file

```
<fragment  
    android:id="@+id/mapLayout"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    class="com.google.android.gms.maps.MapFragment" />
```



# Add Map



```
private void initializeMap() {
    coordinates = new LatLng(currentLocation.getLatitude(), currentLocation.getLongitude());
    myMapFragment=(MapFragment)getFragmentManager().findFragmentById(R.id.mapLayout);
    map = myMapFragment.getMap();
    if (map != null) {
        map.setMyLocationEnabled(true);
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(coordinates.latitude, coordinates.longitude), 13));
    }
}@Override
protected void onStart() {
    super.onStart();
    mClient.connect();
}
@Override
public void onStop() {
    mClient.disconnect();
    super.onStop();
}
@Override
public void onConnected(Bundle arg0) {
    initializeMap();
}
```







# Example – Access Location



```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
  package = "com.example.tutorialspoint7.myapplication">
  <uses-permission android:name = "android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name = "android.permission.INTERNET" />
  <application
    android:allowBackup = "true"
    android:icon = "@mipmap/ic_launcher"
    android:label = "@string/app_name"
    android:supportsRtl = "true"
    android:theme = "@style/AppTheme">

    <activity android:name = ".MainActivity">
      <intent-filter>
        <action android:name = "android.intent.action.MAIN" />
        <category android:name = "android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

**Android Manifest file**



## Example – Access Location



```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >

    <Button
        android:id = "@+id/button"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "getlocation"/>
</LinearLayout>
```

**res/layout/activity\_main.xml file**



# Example – Access Location



```
public class MainActivity extends Activity
{   Button btnShowLocation;
    private static final int REQUEST_CODE_PERMISSION = 2;
    String mPermission = Manifest.permission.ACCESS_FINE_LOCATION;
    GPSTracker gps;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {   super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        try {
            if (ActivityCompat.checkSelfPermission(this, mPermission) !=
MockPackageManager.PERMISSION_GRANTED)
            {   ActivityCompat.requestPermissions(this, new String[]{mPermission},
REQUEST_CODE_PERMISSION);
                // If any permission above not allowed by user, this condition will execute every time, else your else part
will work
            }
        } catch (Exception e) {   e.printStackTrace();   }
        btnShowLocation = (Button) findViewById(R.id.button);    // show location button click event
```

**MainActivity.java**



## Example – Access Location



```
btnShowLocation.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // create class object      gps = new GPSTracker(MainActivity.this);
        // check if GPS enabled
        if(gps.canGetLocation())
        {
            double latitude = gps.getLatitude();
            double longitude = gps.getLongitude();      // \n is for new line
            Toast.makeText(getApplicationContext(), "Your Location is - \nLat: "      + latitude + "\nLong: " +
            longitude, Toast.LENGTH_LONG).show();
        }else
        {
            // can't get location      // GPS or Network is not enabled      // Ask user to enable
            GPS/network in settings
            gps.showSettingsAlert();
        }
    }
});
}
```

MainActivity.java



## Example – Access Location



```
public class GPSTracker extends Service implements LocationListener
{ private final Context mContext;
  // flag for GPS status
  boolean isGPSEnabled = false;
  // flag for network status
  boolean isNetworkEnabled = false;
  // flag for GPS status
  boolean canGetLocation = false;
  Location location; // location
  double latitude; // latitude
  double longitude; // longitude
  // The minimum distance to change Updates in meters
  private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10 meters
  // The minimum time between updates in milliseconds
  private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute
  // Declaring a Location Manager
  protected LocationManager locationManager;
  public GPSTracker(Context context)
  { this.mContext = context; getLocation(); }
```

GpsTracker.java



# Example – Access Location



```
public Location getLocation()
{ try
    { locationManager = (LocationManager) mContext.getSystemService(LOCATION_SERVICE);
    // getting GPS status
    isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
    // getting network status
    isNetworkEnabled = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
    if (!isGPSEnabled && !isNetworkEnabled)
    { // no network provider is enabled }
    else
    { this.canGetLocation = true;
    // First get location from Network Provider
    if (isNetworkEnabled) { locationManager.requestLocationUpdates( LocationManager.NETWORK_PROVIDER,
    MIN_TIME_BW_UPDATES, MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
    Log.d("Network", "Network"); if (locationManager != null)
    { location = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
    if (location != null)
    { latitude = location.getLatitude();
    longitude = location.getLongitude(); } } }
    // if GPS Enabled get lat/long using GPS Services
    if (isGPSEnabled) { if (location == null) { locationManager.requestLocationUpdates( LocationManager.GPS_PROVIDER,
    MIN_TIME_BW_UPDATES, MIN_DISTANCE_CHANGE_FOR_UPDATES, this); Log.d("GPS Enabled", "GPS Enabled");
```

GpsTracker.java



# Example – Access Location



```
if (locationManager != null)
{
location = locationManager .getLastKnownLocation(LocationManager.GPS_PROVIDER);
if (location != null)
{ latitude = location.getLatitude();
longitude = location.getLongitude(); } } } }
} catch (Exception e) { e.printStackTrace(); }
return location; }
/** * Stop using GPS listener * Calling this function will stop using GPS in your app * */
public void stopUsingGPS()
{
if(locationManager != null)
{ locationManager.removeUpdates(GPSTracker.this); } } /** * Function to get latitude * */
public double getLatitude()
{ if(location != null)
{ latitude = location.getLatitude(); } // return latitude return latitude; }
/** * Function to get longitude * */
public double getLongitude()
{ if(location != null)
{ longitude = location.getLongitude(); } // return longitude
return longitude; }
```

GpsTracker.java



# References



- ❑ Anubhav Pradhan, Anil V Deshpande, “Composing Mobile Apps using Android”, Wiley Edition, 2014
- ❑ <https://developer.android.com/training/location>
- ❑ [https://www.tutorialspoint.com/android/android\\_location\\_based\\_services.htm](https://www.tutorialspoint.com/android/android_location_based_services.htm)