Android - 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 e.t.c. You can also customize the map according to your choices.

Google Map - Layout file

Now you have to add the map fragment into xml layout file. Its syntax is given below −

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

Google Map - AndroidManifest file

The next thing you need to do is to add some permissions along with the Google Map API key in the AndroidManifest.XML file. Its syntax is given below −

```
<!--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="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0" />
```

Customizing Google Map

You can easily customize google map from its default view , and change it according to your demand.

Adding Marker

You can place a maker with some text over it displaying your location on the map. It can be done by via **addMarker()** method. Its syntax is given below −

```
final LatLng TutorialsPoint = new LatLng(21 , 57);
Marker TP = googleMap.addMarker(new MarkerOptions()
   .position(TutorialsPoint).title("TutorialsPoint"));
```

Changing Map Type

You can also change the type of the MAP. There are four different types of map and each give a different view of the map. These types are Normal,Hybrid,Satellite and terrain. You can use them as below

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

Enable/Disable zoom

Sr.No	Method & description
1	**addCircle(CircleOptions options)** This method add a circle to the map
2	**addPolygon(PolygonOptions options)** This method add a polygon to the map
3	**addTileOverlay(TileOverlayOptions options)** This method add tile overlay to the map
4	**animateCamera(CameraUpdate update)** This method Moves the map according to the update with an animation
5	**clear()** This method removes everything from the map.
6	**getMyLocation()** This method returns the currently displayed user location.
7	**moveCamera(CameraUpdate update)** This method repositions the camera according to the instructions defined in the update
8	**setTrafficEnabled(boolean enabled)** This method Toggles the traffic layer on or off.
9	**snapshot(GoogleMap.SnapshotReadyCallback callback)** This method Takes a snapshot of the map
10	**stopAnimation()** This method stops the camera animation if there is one in progress

You can also enable or disable the zoom gestures in the map by calling the **setZoomControlsEnabled(boolean)** method. Its syntax is given below −

```
googleMap.getUiSettings().setZoomGesturesEnabled(true);
```

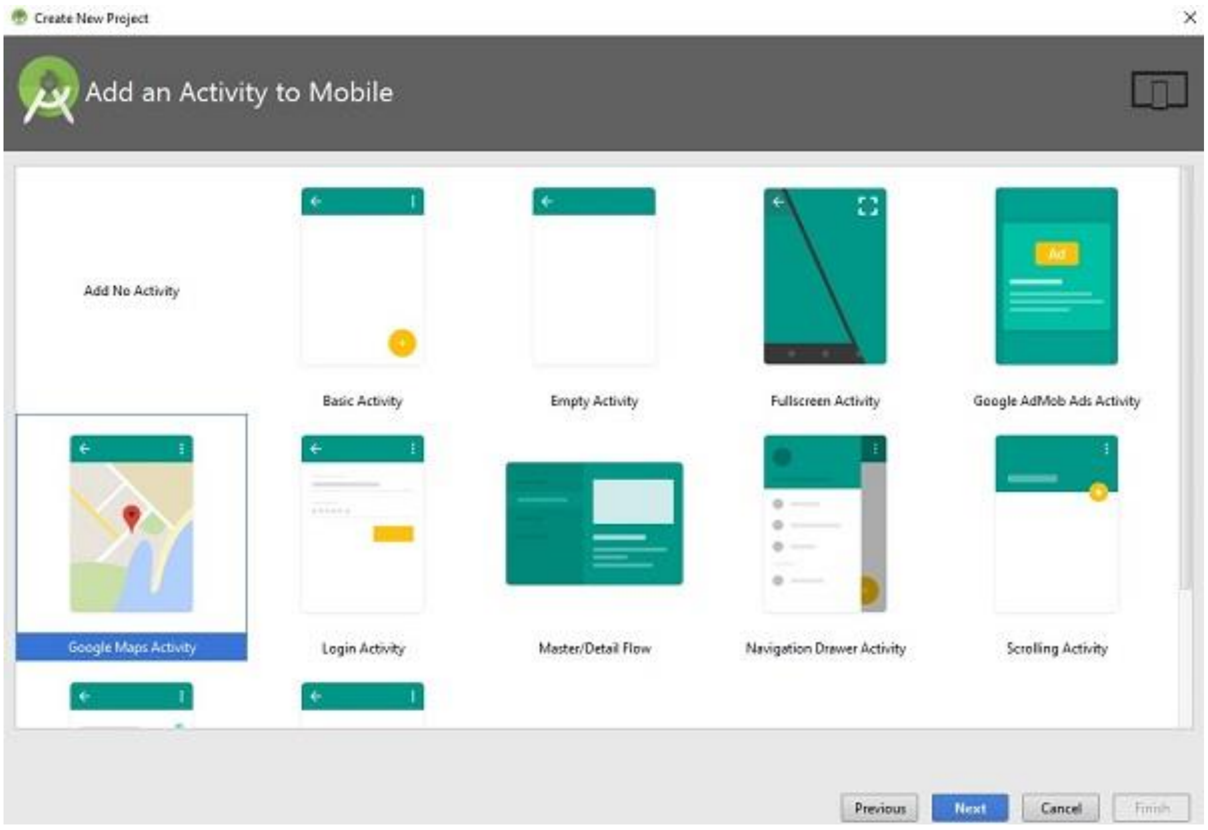
Apart from these customization, there are other methods available in the GoogleMap class , that helps you more customize the map. They are listed below −

Example

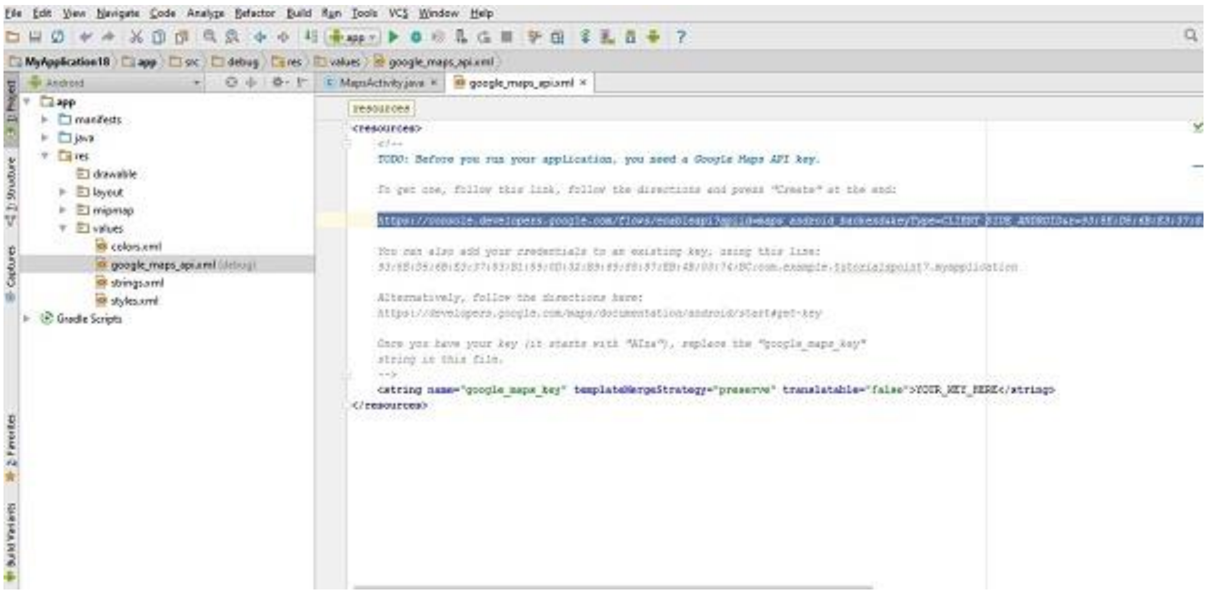
Here is an example demonstrating the use of GoogleMap class. It creates a basic M application that allows you to navigate through the map.

To experiment with this example , you can run this on an actual device or in an emulator.

Create a project with google maps activity as shown below −

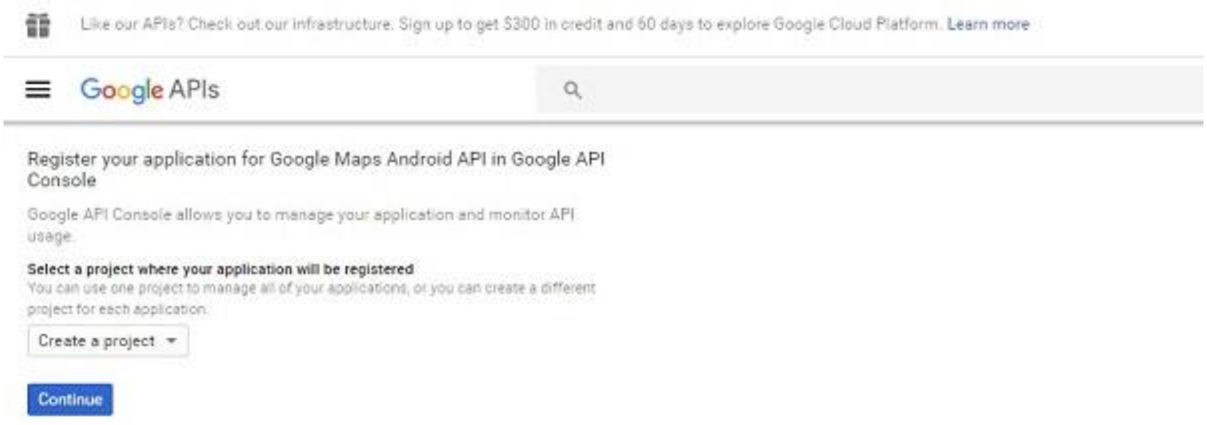


It will open the following screen and copy the console url for API Key as shown below −



```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=93:5E:D6:4B:E3:37:

    You can also add your credentials to an existing key, using this line:
    93:5E:D6:6B:E3:37:93:B1:59:0D:32:B9:89:F0:97:EB:4B:03:74:BC;com.example.tutorialspoint7.myapplication

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
</resources>
```

Copy this and paste it to your browser. It will give the following screen −



Like our APIs? Check out our infrastructure. Sign up to get $300 in credit and 60 days to explore Google Cloud Platform. Learn more

Google APIs

Register your application for Google Maps Android API in Google API Console

Google API Console allows you to manage your application and monitor API usage.

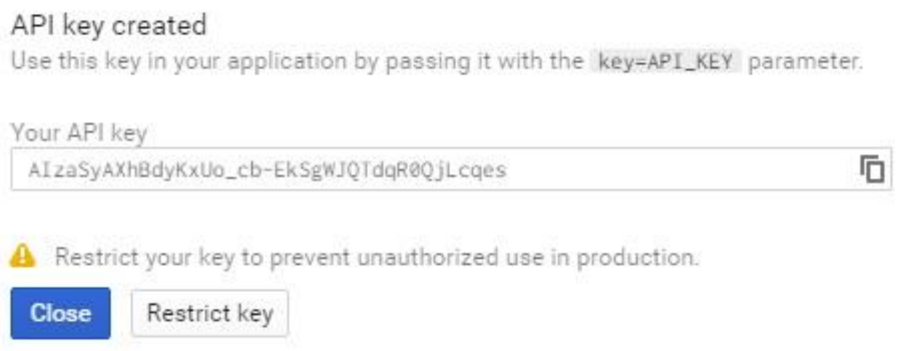
Select a project where your application will be registered

You can use one project to manage all of your applications, or you can create a different project for each application.

Create a project

Continue

Click on continue and click on Create API Key then it will show the following screen



API key created

Use this key in your application by passing it with the key=API_KEY parameter.

Your API key

AIzaSyAXhBdyKxUo_cb-EkSgWJQTdqR0QjLcqes

Restrict your key to prevent unauthorized use in production.

Close   Restrict key

Here is the content of **activity_main.xml**.

```
<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="com.example.tutorialspoint7.myapplication.MapsActi
vity" />
```

Here is the content of **MapActivity.java**.

In the below code we have given sample latitude and longitude details

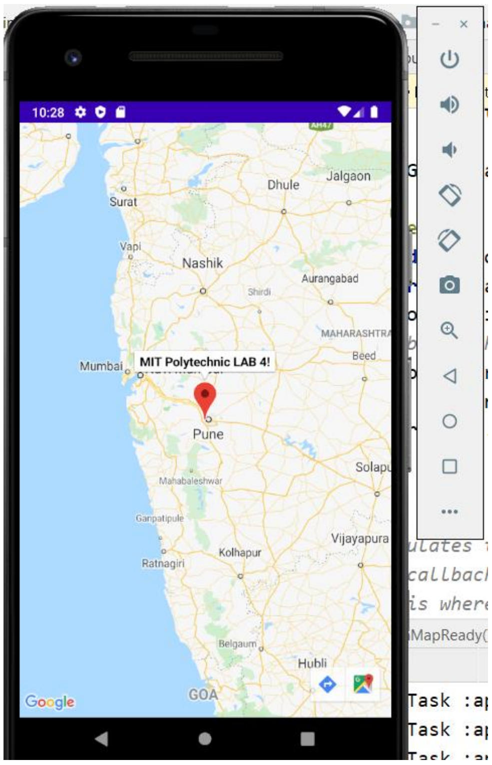
```
package com.example.tutorialspoint7.myapplication;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

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 {

   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(this);
   }

   /**
      * 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.
      * 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;
      // Add a marker in Sydney and move the camera
      LatLng TutorialsPoint = new LatLng(21, 57);
      mMap.addMarker(new

MarkerOptions().position(TutorialsPoint).title("Tutorialspoint.co
m"));

mMap.moveCamera(CameraUpdateFactory.newLatLng(TutorialsPoint));
   }
}
```

Following is the content of **AndroidManifest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

   <!--
      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.
   -->

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

      <!--
         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="AIzaSyAXhBdyKxUo_cb-EkSgWJQTdqR0QjLcqes"
/>

      <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>
```

Output should be like this −



Sending SMS

In Android, you can use SmsManager API or devices Built-in SMS application to send SMS's. In this tutorial, we shows you two basic examples to send SMS message −

SmsManager API

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage("phoneNo", null, "sms message", null,
null);
```

Built-in SMS application

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);
```

Of course, both need **SEND_SMS permission**.

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

Apart from the above method, there are few other important functions available in SmsManager class. These methods are listed below −

Sr.No.	Method & Description
1	**ArrayList<String> divideMessage(String text)** This method divides a message text into several fragments, none bigger than the maximum SMS message size.
2	**static SmsManager getDefault()** This method is used to get the default instance of the SmsManager
3	**void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)** This method is used to send a data based SMS to a specific application port.
4	**void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)** Send a multi-part text based SMS.
5	**void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)** Send a text based SMS.

Example

Following example shows you in practical how to use SmsManager object to send an SMS to the given mobile number.

To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

Step	Description
1	You will use Android Studio IDE to create an Android application and name it as *tutorialspoint* under a package *com.example.tutorialspoint*.
2	Modify *src/MainActivity.java* file and add required code to take care of sending sms.
3	Modify layout XML file *res/layout/activity_main.xml* add any GUI component if required. I'm adding a simple GUI to take mobile number and SMS text to be sent and a simple button to send SMS.
4	No need to define default string constants at res/values/strings.xml. Android studio takes care of default constants.
5	Modify *AndroidManifest.xml* as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.tutorialspoint/MainActivity.java**.

```
package com.example.tutorialspoint;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.app.Activity;

import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.telephony.SmsManager;

import android.util.Log;
import android.view.Menu;
import android.view.View;

import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
   private static final int MY_PERMISSIONS_REQUEST_SEND_SMS =0 ;
   Button sendBtn;
   EditText txtphoneNo;
   EditText txtMessage;
   String phoneNo;
   String message;

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

      sendBtn = (Button) findViewById(R.id.btnSendSMS);
      txtphoneNo = (EditText) findViewById(R.id.editText);
      txtMessage = (EditText) findViewById(R.id.editText2);

      sendBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            sendSMSMessage();
         }
      });
   }

   protected void sendSMSMessage() {
      phoneNo = txtphoneNo.getText().toString();
      message = txtMessage.getText().toString();

      if (ContextCompat.checkSelfPermission(this,
         Manifest.permission.SEND_SMS)
         != PackageManager.PERMISSION_GRANTED) {
            if
(ActivityCompat.shouldShowRequestPermissionRationale(this,
               Manifest.permission.SEND_SMS)) {
            } else {
               ActivityCompat.requestPermissions(this,
                  new String[]{Manifest.permission.SEND_SMS},
                  MY_PERMISSIONS_REQUEST_SEND_SMS);
            }
      }
   }

   @Override
   public void onRequestPermissionsResult(int requestCode,String
permissions[], int[] grantResults) {
      switch (requestCode) {
         case MY_PERMISSIONS_REQUEST_SEND_SMS: {
            if (grantResults.length > 0
               && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
```

```
                  SmsManager smsManager =
SmsManager.getDefault();
                  smsManager.sendTextMessage(phoneNo, null,
message, null, null);
                  Toast.makeText(getApplicationContext(), "SMS
sent.",
                     Toast.LENGTH_LONG).show();
            } else {
               Toast.makeText(getApplicationContext(),
                  "SMS faild, please try again.",
Toast.LENGTH_LONG).show();
               return;
            }
         }
      }

   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

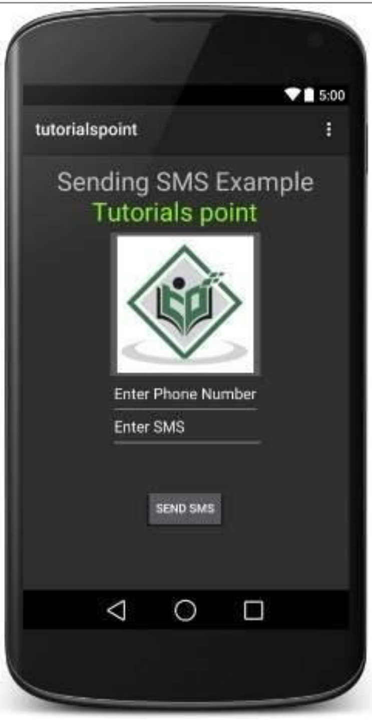
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="MainActivity">

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Sending SMS Example"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_below="@+id/textView1"
      android:layout_alignRight="@+id/imageButton"
      android:layout_alignEnd="@+id/imageButton" />
   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true" />

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText"
      android:hint="Enter Phone Number"
      android:phoneNumber="true"

android:textColorHint="@color/abc_primary_text_material_dark"
      android:layout_below="@+id/imageButton"
      android:layout_centerHorizontal="true" />

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText2"
      android:layout_below="@+id/editText"
      android:layout_alignLeft="@+id/editText"
      android:layout_alignStart="@+id/editText"

android:textColorHint="@color/abc_primary_text_material_dark"
      android:layout_alignRight="@+id/imageButton"
      android:layout_alignEnd="@+id/imageButton"
      android:hint="Enter SMS" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Send Sms"
      android:id="@+id/btnSendSMS"
      android:layout_below="@+id/editText2"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="48dp" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">tutorialspoint</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

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

   <uses-permission android:name="android.permission.SEND_SMS" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name="com.example.tutorialspoint.MainActivity"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>

   </application>
</manifest>
```



Now you can enter a desired mobile number and a text message to be sent on that number. Finally click on **Send SMS** button to send your SMS. Make sure your GSM/CDMA connection is working fine to deliver your SMS to its recipient.

You can take a number of SMS separated by comma and then inside your program you will have to parse them into an array string and finally you can use a loop to send message to all the given numbers. That's how you can write your own SMS client. Next section will show you how to use existing SMS client to send SMS.

Using Built-in Intent to send SMS

You can use Android Intent to send SMS by calling built-in SMS functionality of the Android. Following section explains different parts of our Intent object required to send an SMS.

Intent Object - Action to send SMS

You will use **ACTION_VIEW** action to launch an SMS client installed on your Android device. Following is simple syntax to create an intent with ACTION_VIEW action.

```
Intent smsIntent = new Intent(Intent.ACTION_VIEW);
```

Intent Object - Data/Type to send SMS

To send an SMS you need to specify **smsto:** as URI using setData() method and data type will be to **vnd.android-dir/mms-sms** using setType() method as follows −

```
smsIntent.setData(Uri.parse("smsto:"));
smsIntent.setType("vnd.android-dir/mms-sms");
```

Intent Object - Extra to send SMS

Android has built-in support to add phone number and text message to send an SMS as follows −

```
smsIntent.putExtra("address"  , new
String("0123456789;3393993300"));
smsIntent.putExtra("sms_body"  , "Test SMS to Angilla");
```

Here address and sms_body are case sensitive and should be specified in small characters only. You can specify more than one number in single string but separated by semi-colon (;).

Example

Following example shows you in practical how to use Intent object to launch SMS client to send an SMS to the given recipients.

To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *tutorialspoint* under a package *com.example.tutorialspoint*.
2	Modify *src/MainActivity.java* file and add required code to take care of sending SMS.
3	Modify layout XML file *res/layout/activity_main.xml* add any GUI component if required. I'm adding a simple button to launch SMS Client.

4	No need to define default constants.Android studio takes care of default constants.
5	Modify *AndroidManifest.xml* as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.tutorialspoint/MainActivity.java**.

```
package com.example.tutorialspoint;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.button);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            sendSMS();
         }
      });
   }

   protected void sendSMS() {
      Log.i("Send SMS", "");
      Intent smsIntent = new Intent(Intent.ACTION_VIEW);

      smsIntent.setData(Uri.parse("smsto:"));
      smsIntent.setType("vnd.android-dir/mms-sms");
      smsIntent.putExtra("address"  , new String ("01234"));
      smsIntent.putExtra("sms_body"  , "Test ");

      try {
         startActivity(smsIntent);
         finish();
         Log.i("Finished sending SMS...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this,
         "SMS faild, please try again later.",
Toast.LENGTH_SHORT).show();
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if
it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

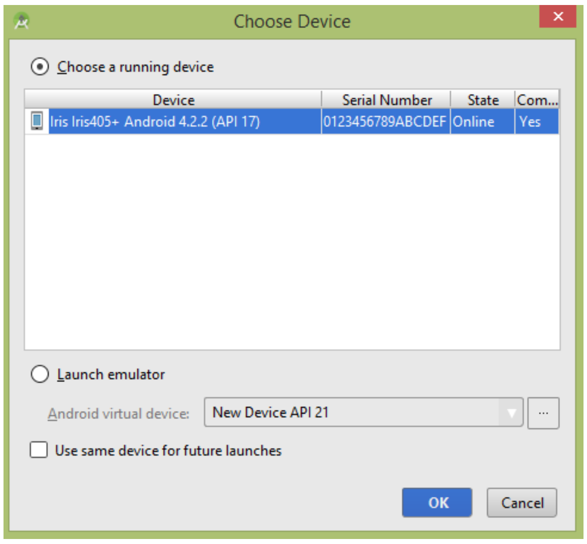
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Drag and Drop Example"
      android:id="@+id/textView"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials Point "
      android:id="@+id/textView2"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true"
      android:textSize="30dp"
      android:textColor="#ff14be3c" />

   <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageView"
      android:src="@drawable/abc"
      android:layout_marginTop="48dp"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Compose SMS"
      android:id="@+id/button"
      android:layout_below="@+id/imageView"
      android:layout_alignRight="@+id/textView2"
      android:layout_alignEnd="@+id/textView2"
      android:layout_marginTop="54dp"
      android:layout_alignLeft="@+id/imageView"
      android:layout_alignStart="@+id/imageView" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants –

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">tutorialspoint</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** –

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

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name="com.example.tutorialspoint.MainActivity"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>

   </application>
</manifest>
```



Choose Device

(●) Choose a running device

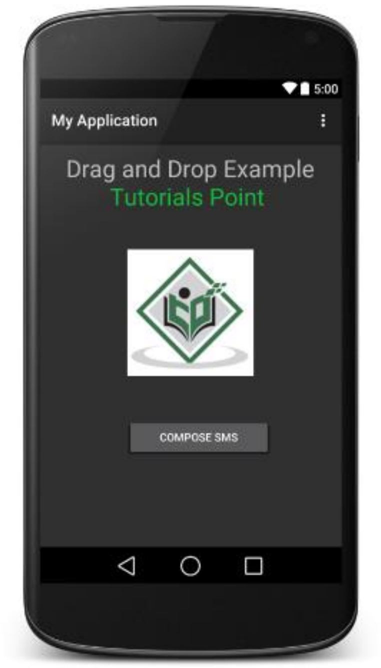
Device	Serial Number	State	Com...
Iris Iris405+ Android 4.2.2 (API 17)	0123456789ABCDEF	Online	Yes

( ) Launch emulator

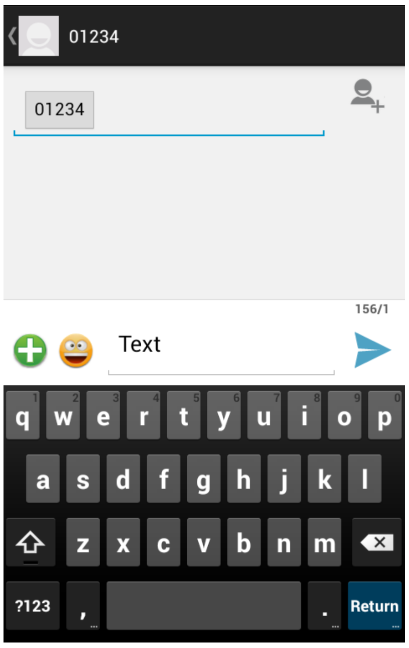
Android virtual device: New Device API 21

[ ] Use same device for future launches

Select your mobile device as an option and then check your mobile device which will display following screen −



Now use **Compose SMS** button to launch Android built-in SMS clients which is shown below −



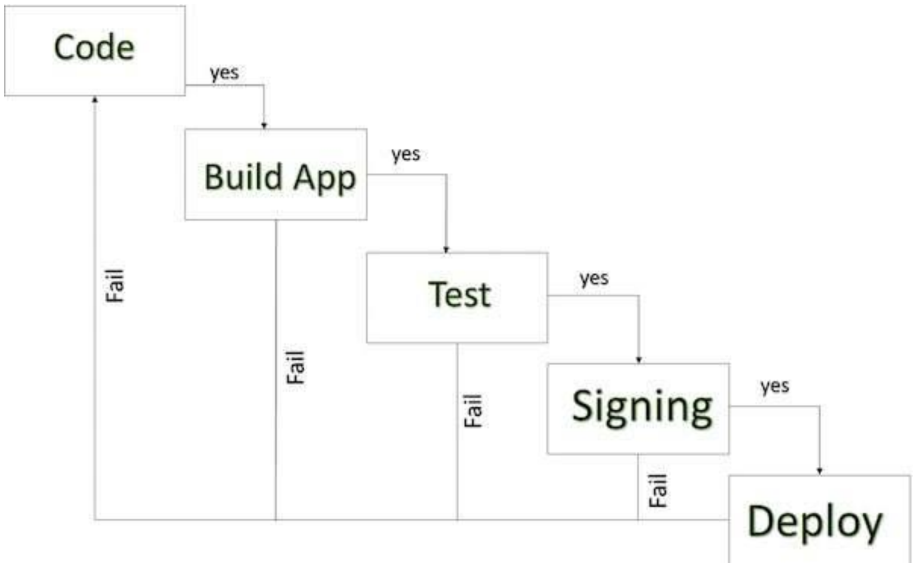
You can modify either of the given default fields and finally use send SMS button to send your SMS to the mentioned recipient.

Publishing Android Application

Step	Activity
1	**Regression Testing** Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all of the devices that you are targeting. So perform all the required testing on different devices including phone and tablets.
2	**Application Rating** When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.
3	**Targeted Regions** Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.
4	**Application Size** Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.
5	**SDK and Screen Compatibility** It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.

6	**Application Pricing** Deciding whether you app will be free or paid is important because, on Google Play, free app's must remain free. If you want to sell your application then you will have to specify its price in different currencies.
7	**Promotional Content** It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.
8	**Build and Upload release-ready APK** The release-ready APK is what you you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.
9	**Finalize Application Detail** Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colourful graphics, screen shots, and videos to localized descriptions, release details, and links to your other apps. So you can decorate your application page and provide as much as clear crisp detail you can provide.

Android application publishing is a process that makes your Android applications available to users. Infect, publishing is the last phase of the Android application development process.

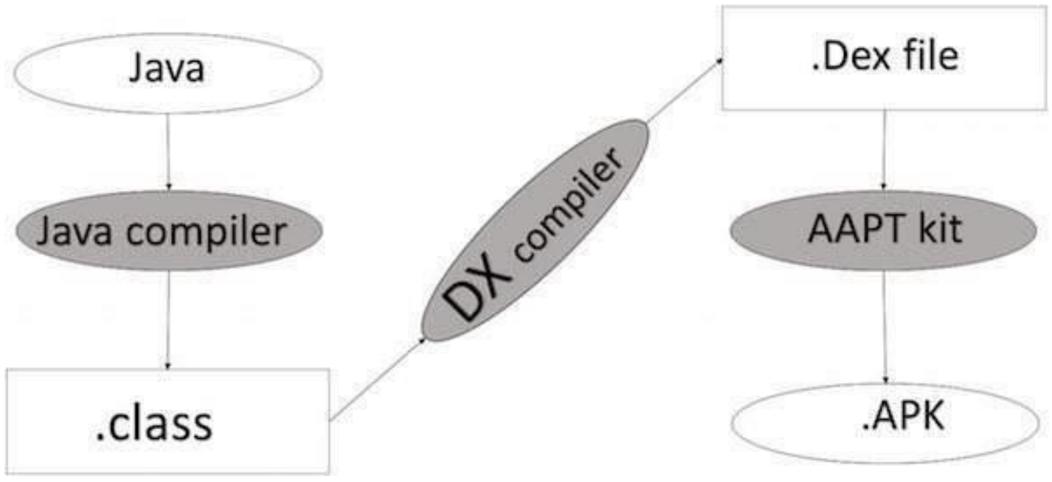


Android development life cycle

Once you developed and fully tested your Android Application, you can start selling or distributing free using Google Play (A famous Android marketplace). You can also release your applications by sending them directly to users or by letting users download them from your own website.

You can check a detailed publishing process at Android official website, but this tutorial will take you through simple steps to launch your application on Google Play. Here is a simplified check list which will help you in launching your Android application −

Export Android Application Process



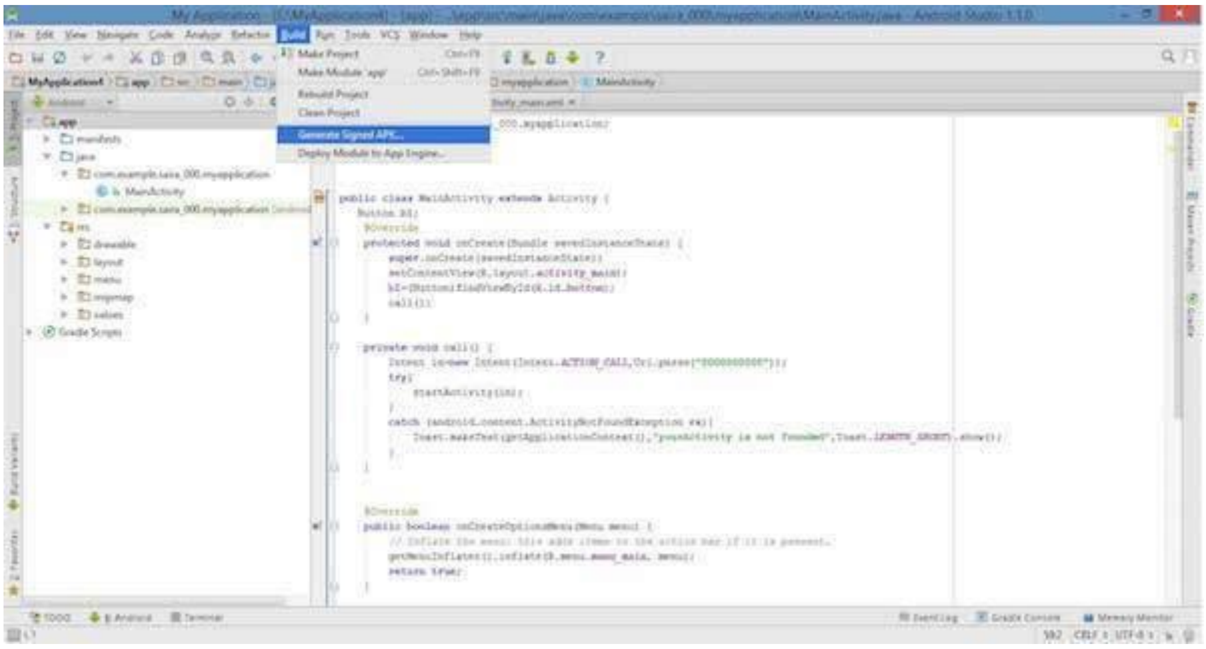
Apk Development Process

Before exporting the apps, you must some of tools

- **Dx tools**(Dalvik executable tools): It going to convert **.class file** to **.dex file**. it has useful for memory optimization and reduce the boot-up speed time
- **AAPT**(Android assistance packaging tool):it has useful to convert **.Dex file** to**.Apk**
- **APK**(Android packaging kit): The final stage of deployment process is called as .apk.

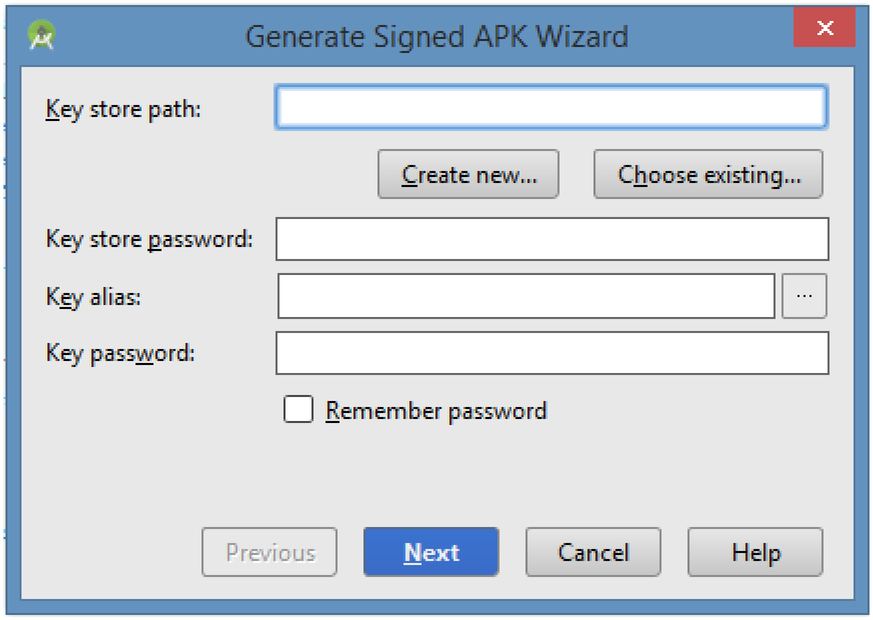
You will need to export your application as an APK (Android Package) file before you upload it Google Play marketplace.

To export an application, just open that application project in Android studio and select **Build → Generate Signed APK** from your Android studio and follow the simple steps to export your application −

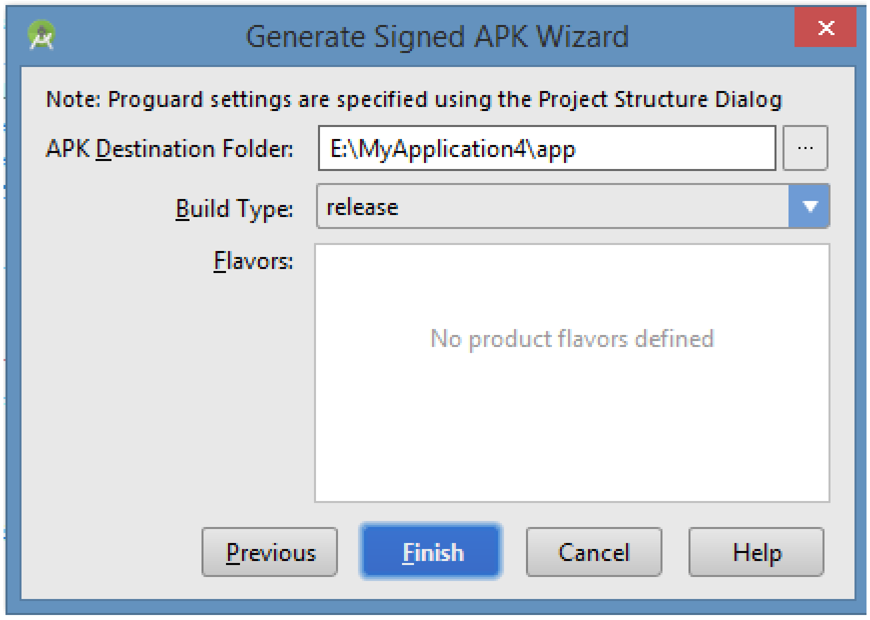

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Make Project Ctrl+F9
Make Module 'app' Ctrl+Shift+F9
Rebuild Project
Clean Project
Generate Signed APK...
Deploy Module to App Engine...
MyApplication4 app src main
app
manifests
java
com.example.saira_000.myapplication
MainActivity
res
drawable
layout
menu
mipmap
values
Gradle Scripts

```
public class MainActivity extends Activity {
    Button b1;
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
         b1=(Button)findViewById(R.id.button);
         call();
     }

     private void call() {
         Intent in=new Intent(Intent.ACTION_CALL,Uri.parse("0000000000"));
         try{
             startActivity(in);
         }
         catch (android.content.ActivityNotFoundException ex){
             Toast.makeText(getApplicationContext(),"yourActivity is not founded",Toast.LENGTH_SHORT).show();
         }
     }

     @Override
     public boolean onCreateOptionsMenu(Menu menu) {
         // Inflate the menu; this adds items to the action bar if it is present.
         getMenuInflater().inflate(R.menu.menu_main, menu);
         return true;
     }
```

Next select, **Generate Signed APK** option as shown in the above screen shot and then click it so that you get following screen where you will choose **Create new keystore** to store your application.



Enter your key store path,key store password,key alias and key password to protect your application and click on **Next** button once again. It will display following screen to let you create an application –



Note: Proguard settings are specified using the Project Structure Dialog

APK Destination Folder: E:\MyApplication4\app

Build Type: release

Flavors: No product flavors defined

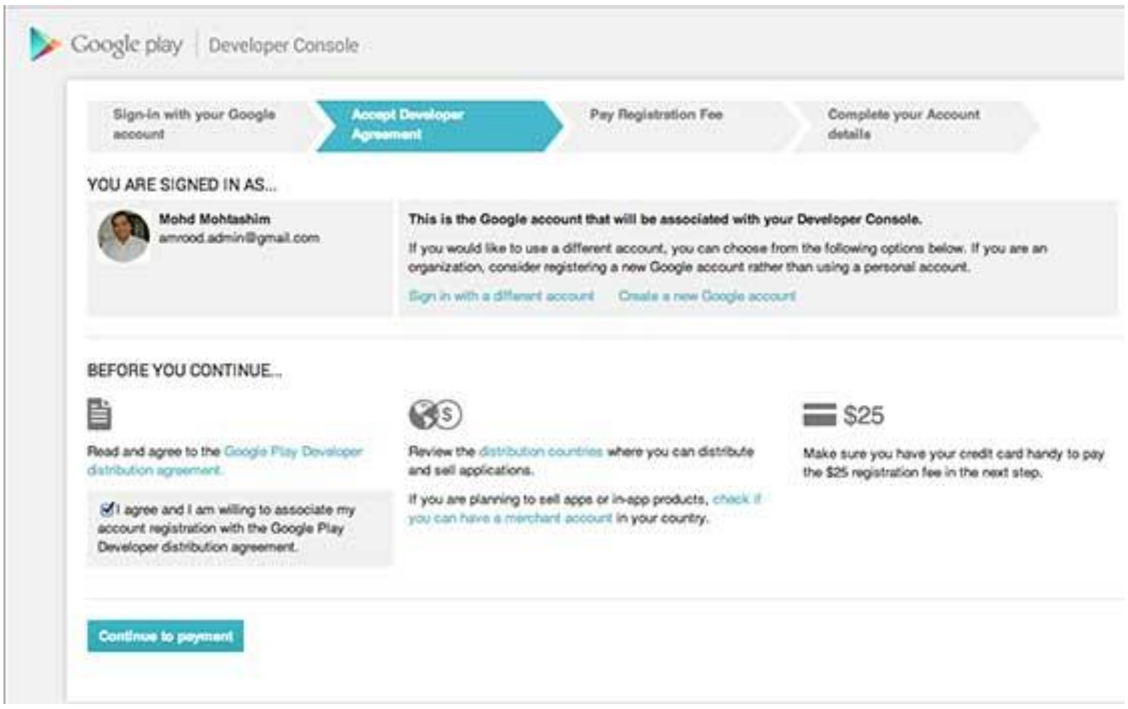
Once you filled up all the information,like app destination,build type and flavours click **finish** button While creating an application it will show as below

Gradle: Executing Tasks [:app:assembleRelease]

Finally, it will generate your Android Application as APK formate File which will be uploaded at Google Play marketplace.

Google Play Registration

The most important step is to register with Google Play using Google Play Marketplace. You can use your existing google ID if you have any otherwise you can create a new Google ID and then register with the marketplace. You will have following screen to accept terms and condition.



Google play | Developer Console

Sign-in with your Google account → Accept Developer Agreement → Pay Registration Fee → Complete your Account details

YOU ARE SIGNED IN AS...

Mohd Mohtashim
amrood.admin@gmail.com

This is the Google account that will be associated with your Developer Console.

If you would like to use a different account, you can choose from the following options below. If you are an organization, consider registering a new Google account rather than using a personal account.

Sign in with a different account   Create a new Google account

BEFORE YOU CONTINUE...

Read and agree to the Google Play Developer distribution agreement.

[✓] I agree and I am willing to associate my account registration with the Google Play Developer distribution agreement.

Review the distribution countries where you can distribute and sell applications.

If you are planning to sell apps or in-app products, check if you can have a merchant account in your country.

$25

Make sure you have your credit card handy to pay the $25 registration fee in the next step.

Continue to payment

You can use **Continue to payment** button to proceed to make a payment of $25 as a registration fee and finally to complete your account detail.

Once you are a registered user at Google Play, you can upload **release-ready APK** for your application and finally you will complete application detail using application detail page as mentioned in step 9 of the above mentioned checklist.

Signing Your App Manually

You do not need Android Studio to sign your app. You can sign your app from the command line using standard tools from the Android SDK and the JDK. To sign an app in release mode from the command line −

- Generate a private key using keytool

```
$ keytool -genkey -v -keystore my-release-key.keystore
-alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

- Compile your app in release mode to obtain an unsigned APK
- Sign your app with your private key using jarsigner

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1
-keystore my-release-key.keystore my_application.apk alias_name
```

- Verify that your APK is signed. For example −

```
$ jarsigner -verify -verbose -certs my_application.apk
```

- Align the final APK package using zipalign.

```
$ zipalign -v 4 your_project_name-unaligned.apk
your_project_name.apk
```

Geocoding

Geocoding is the process of converting the addresses (postal address) into geo coordinates as latitude and longitude. Reverse geocoding is converting a geo coordinate latitude and longitude to an address. In this tutorial we will be doing reverse geo coding and get the addresses of the passed coordinates.

Android Reverse Geocoding Example to find Address

Step1: Define Permissions

We need location access permission to find the latitude and longitude of the Android device. Following two lines are the key to give permission to access the location.

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

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

So the complete manifest file will be as below:

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="android.javapapers.com.androidgeocodelocation" >



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

    <uses-permission android:name="android.permission.INTERNET" />



    <application

        android:allowBackup="true"

        android:icon="@drawable/ic_launcher"

        android:label="@string/app_name"

        android:theme="@style/AppTheme" >

        <activity

            android:name=".MyActivity"

            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

        </activity>

    </application>



</manifest>
```

Step 2: Accessing the Geo Location for Lat and Long
Following class is the key element in accessing the latitude and longitude of the Android device. It implements the LocationListener and gets the location coordinate updates. We have designed our requirement not to be a continous update for location. On demand we will get the location coordinates.

AppLocationService.java

```
package android.javapapers.com.androidgeocodelocation;



import android.app.Service;

import android.content.Context;

import android.content.Intent;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.os.IBinder;



public class AppLocationService extends Service implements LocationListener {



    protected LocationManager locationManager;
```

```
Location location;


private static final long MIN_DISTANCE_FOR_UPDATE = 10;

private static final long MIN_TIME_FOR_UPDATE = 1000 * 60 * 2;


public AppLocationService(Context context) {

    locationManager = (LocationManager) context

            .getSystemService(LOCATION_SERVICE);

}


public Location getLocation(String provider) {

    if (locationManager.isProviderEnabled(provider)) {

        locationManager.requestLocationUpdates(provider,

                MIN_TIME_FOR_UPDATE, MIN_DISTANCE_FOR_UPDATE, this);

        if (locationManager != null) {

            location = locationManager.getLastKnownLocation(provider);

            return location;

        }

    }

    return null;

}


@Override

public void onLocationChanged(Location location) {

}


@Override
```

```
    public void onProviderDisabled(String provider) {

    }



    @Override

    public void onProviderEnabled(String provider) {

    }



    @Override

    public void onStatusChanged(String provider, int status, Bundle extras) {

    }



    @Override

    public IBinder onBind(Intent arg0) {

        return null;

    }



}
```

Step 3: Reverse Geocoding to get Location Address
Following class is the key element for reverse geocoding to get the address for the passed latitude and longitude coordinates. We access the Geocoder Google API for reverse geocoding and get every line of address like street, city, pin / zip code and etc.

LocationAddress.java

```
package android.javapapers.com.androidgeocodelocation;



import android.content.Context;

import android.location.Address;

import android.location.Geocoder;

import android.os.Bundle;
import android.os.Handler;

import android.os.Message;

import android.util.Log;


import java.io.IOException;

import java.util.List;

import java.util.Locale;


public class LocationAddress {

    private static final String TAG = "LocationAddress";


    public static void getAddressFromLocation(final double latitude, final double longitude,

                                              final Context context, final Handler handler) {

        Thread thread = new Thread() {

            @Override

            public void run() {

                Geocoder geocoder = new Geocoder(context, Locale.getDefault());

                String result = null;

                try {

                    List<Address> addressList = geocoder.getFromLocation(

                            latitude, longitude, 1);

                    if (addressList != null && addressList.size() > 0) {

                        Address address = addressList.get(0);

                        StringBuilder sb = new StringBuilder();

                        for (int i = 0; i < address.getMaxAddressLineIndex(); i++) {

                            sb.append(address.getAddressLine(i)).append("\n");

                        }
                        sb.append(address.getLocality()).append("\n");
                        sb.append(address.getPostalCode()).append("\n");
                        sb.append(address.getCountryName());
                        result = sb.toString();
                    }
                } catch (IOException e) {
                    Log.e(TAG, "Unable connect to Geocoder", e);
                } finally {
                    Message message = Message.obtain();
                    message.setTarget(handler);
                    if (result != null) {
                        message.what = 1;
                        Bundle bundle = new Bundle();
                        result = "Latitude: " + latitude + " Longitude: " + longitude +
                                "\n\nAddress:\n" + result;
                        bundle.putString("address", result);
                        message.setData(bundle);
                    } else {
                        message.what = 1;
                        Bundle bundle = new Bundle();
                        result = "Latitude: " + latitude + " Longitude: " + longitude +
                                "\n Unable to get address for this lat-long.";
                        bundle.putString("address", result);
                        message.setData(bundle);
                    }
                    message.sendToTarget();
                }
```

```
            }

        };

        thread.start();

    }

}
```

Step 4: Android UI
How these pieces fit together is with the following Android activity.

MyActivity.java

```
package android.javapapers.com.androidgeocodelocation;



import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.location.Location;

import android.location.LocationManager;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.provider.Settings;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

public class MyActivity extends Activity {



    Button btnGPSShowLocation;

    Button btnShowAddress;
```

```
TextView tvAddress;

AppLocationService appLocationService;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    tvAddress = (TextView) findViewById(R.id.tvAddress);
    appLocationService = new AppLocationService(
            MyActivity.this);

    btnGPSShowLocation = (Button) findViewById(R.id.btnGPSShowLocation);
    btnGPSShowLocation.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Location gpsLocation = appLocationService
                    .getLocation(LocationManager.GPS_PROVIDER);
            if (gpsLocation != null) {
                double latitude = gpsLocation.getLatitude();
                double longitude = gpsLocation.getLongitude();
                String result = "Latitude: " + gpsLocation.getLatitude() +
                        " Longitude: " + gpsLocation.getLongitude();
                tvAddress.setText(result);
            } else {
                showSettingsAlert();
            }
        }
    });

    btnShowAddress = (Button) findViewById(R.id.btnShowAddress);
    btnShowAddress.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {

            Location location = appLocationService
                    .getLocation(LocationManager.GPS_PROVIDER);

            //you can hard-code the lat & long if you have issues with getting it
            //remove the below if-condition and use the following couple of lines
            //double latitude = 37.422005;
            //double longitude = -122.084095

            if (location != null) {
                double latitude = location.getLatitude();
                double longitude = location.getLongitude();
                LocationAddress locationAddress = new LocationAddress();
                locationAddress.getAddressFromLocation(latitude, longitude,
                        getApplicationContext(), new GeocoderHandler());
            } else {
                showSettingsAlert();
            }

        }
    });


}


public void showSettingsAlert() {

    AlertDialog.Builder alertDialog = new AlertDialog.Builder(

            MyActivity.this);

    alertDialog.setTitle("SETTINGS");

    alertDialog.setMessage("Enable Location Provider! Go to settings menu?");

    alertDialog.setPositiveButton("Settings",

            new DialogInterface.OnClickListener() {

                public void onClick(DialogInterface dialog, int which) {

                    Intent intent = new Intent(

                            Settings.ACTION_LOCATION_SOURCE_SETTINGS);

                    MyActivity.this.startActivity(intent);

                }

            });

    alertDialog.setNegativeButton("Cancel",

            new DialogInterface.OnClickListener() {

                public void onClick(DialogInterface dialog, int which) {

                    dialog.cancel();

                }

            });

    alertDialog.show();

}


private class GeocoderHandler extends Handler {
        @Override
        public void handleMessage(Message message) {
            String locationAddress;
            switch (message.what) {
                case 1:
                    Bundle bundle = message.getData();
                    locationAddress = bundle.getString("address");
                    break;
                default:
                    locationAddress = null;
            }
            tvAddress.setText(locationAddress);
        }
    }
}
```

Android UI layout file to show the address and location

```
<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"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MyActivity">
```

```
<TextView

    android:text="@string/hello_world"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:id="@+id/textView" />


<Button

    style="?android:attr/buttonStyleSmall"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Show Location"

    android:id="@+id/btnGPSShowLocation"

    android:layout_toEndOf="@+id/textView"

    android:layout_marginTop="53dp"

    android:layout_below="@+id/textView"

    android:layout_alignParentStart="true" />


<Button

    style="?android:attr/buttonStyleSmall"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Show Address"

    android:id="@+id/btnShowAddress"

    android:layout_toEndOf="@+id/tvAddress"

    android:layout_below="@+id/btnGPSShowLocation"

    android:layout_alignParentStart="true" />
```

```
   <TextView

       android:layout_width="fill_parent"

       android:layout_height="wrap_content"

       android:id="@+id/tvAddress"

       android:layout_alignParentBottom="true"

       android:layout_marginBottom="134dp"

       android:layout_alignParentEnd="true" />



</RelativeLayout>
```

Key Points to Note for Geocoding

- Need Google API access and so remember to run in an emulator that is configured for Google API.
- Remember to have the right permissions defined in the Mainfest.xml
- To test via an Android Emulator, we can use the DDMS and pass the latitude and longitude using the 'Emulator Control'.

Example Geo coordinates Latitude and Longitude with respective location address:

```
Latitude, Longitude: 38.898748, -77.037684

Location Address:

1600 Pennsylvania Ave NW

Washington DC 20502



Latitude, Longitude: 34.101509, -118.32691

Location Address:

Hollywood Blvd & Vine St

Los Angeles CA 90028
```