Question Bank-MAD

State intent. List types of intent.

State intent. List types of intent.

Intent is the message that is passed between components such as activities.

Android uses Intent for communicating between the components of an Application and also from one application to another application.

Types:

- Explicit Intent
- Implicit Intent

Define:
i)Fragment
ii) Broadcast receiver

Define:
i)Fragment
ii) Broadcast receiver

Fragment:
Fragment is the part of activity, it is also known as sub-activity.

Broadcast receiver:

A broadcast receiver is a dormant component of the Android system. The Broadcast Receiver's job is to pass a notification to the user, in case a specific event occurs.

Name two classes used to play audio and video in Android.

Name two classes used to play audio and video in Android.

1) MediaPlayer

2) MediaController

3) AudioManager

Explain two methods of Google Map.

Explain two methods of Google Map.

getMyLocation(): This method returns the currently displayed user location.

moveCamera(CameraUpdate update): This method reposition the camera according to the instructions defined in the update.

List types of permissions in android.

1. Normal Permissions
2. Dangerous Permissions
3. Signature Permissions

Define Geocoding and Reverse Geocoding.

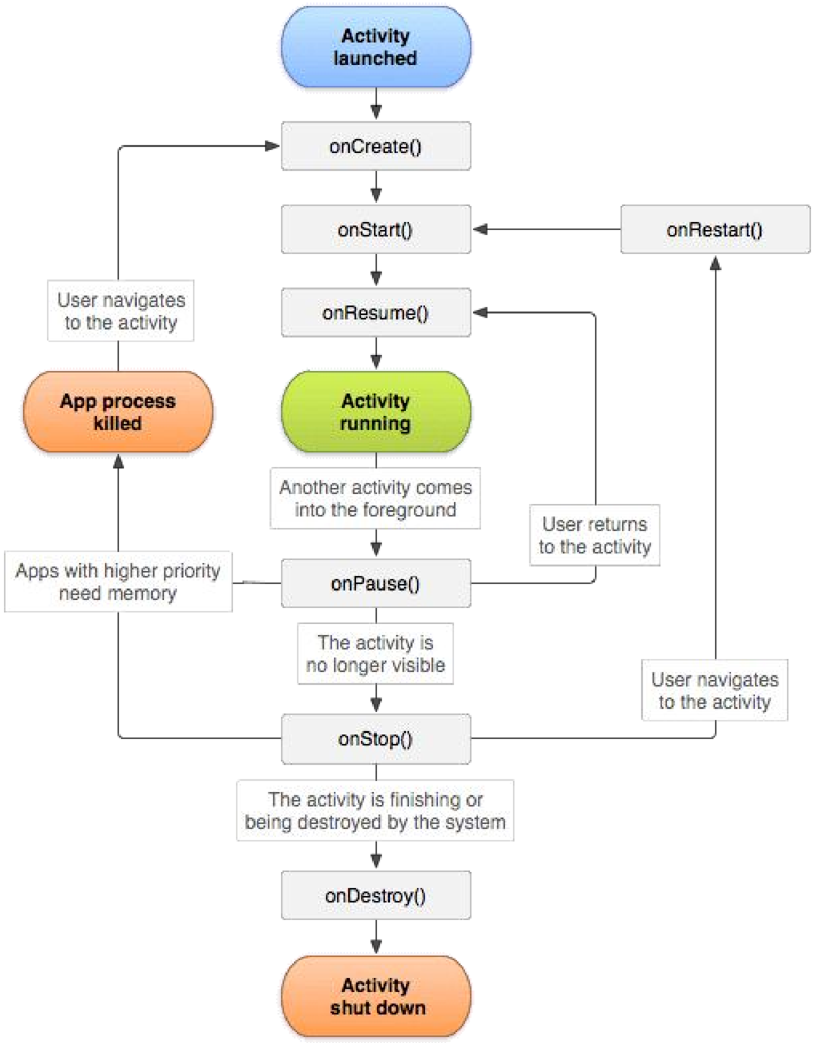
Geocoding :

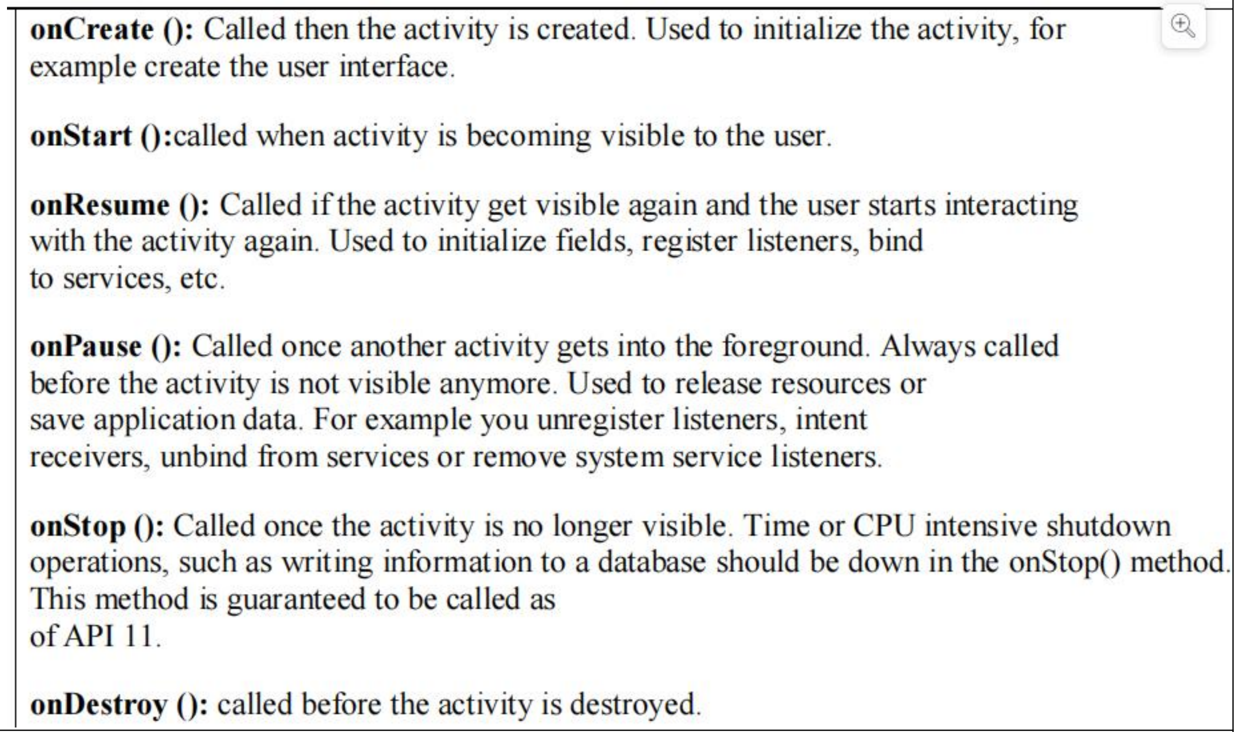
Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate.

Reverse Geocoding :

Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into a (partial) address.

Draw and explain activity life cycle.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down



onCreate (): Called then the activity is created. Used to initialize the activity, for example create the user interface.

onStart ():called when activity is becoming visible to the user.

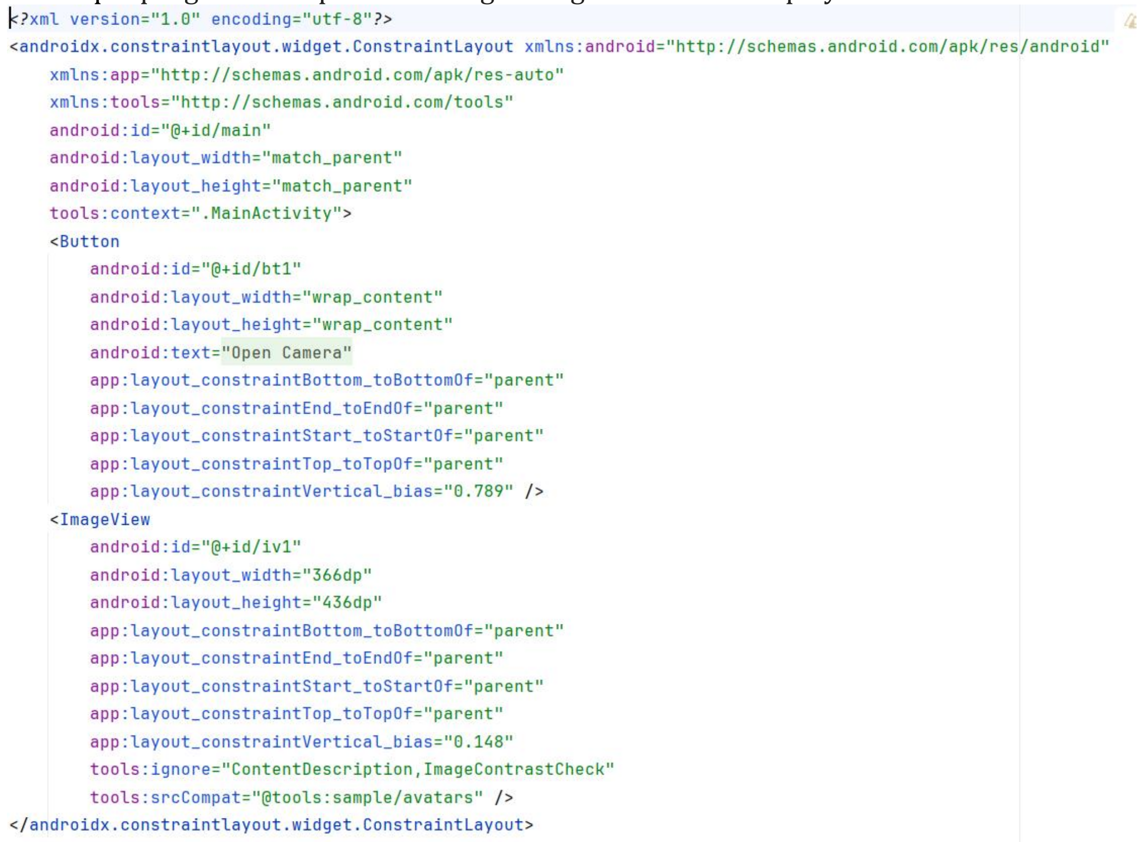
onResume (): Called if the activity get visible again and the user starts interacting with the activity again. Used to initialize fields, register listeners, bind to services, etc.

onPause (): Called once another activity gets into the foreground. Always called before the activity is not visible anymore. Used to release resources or save application data. For example you unregister listeners, intent receivers, unbind from services or remove system service listeners.

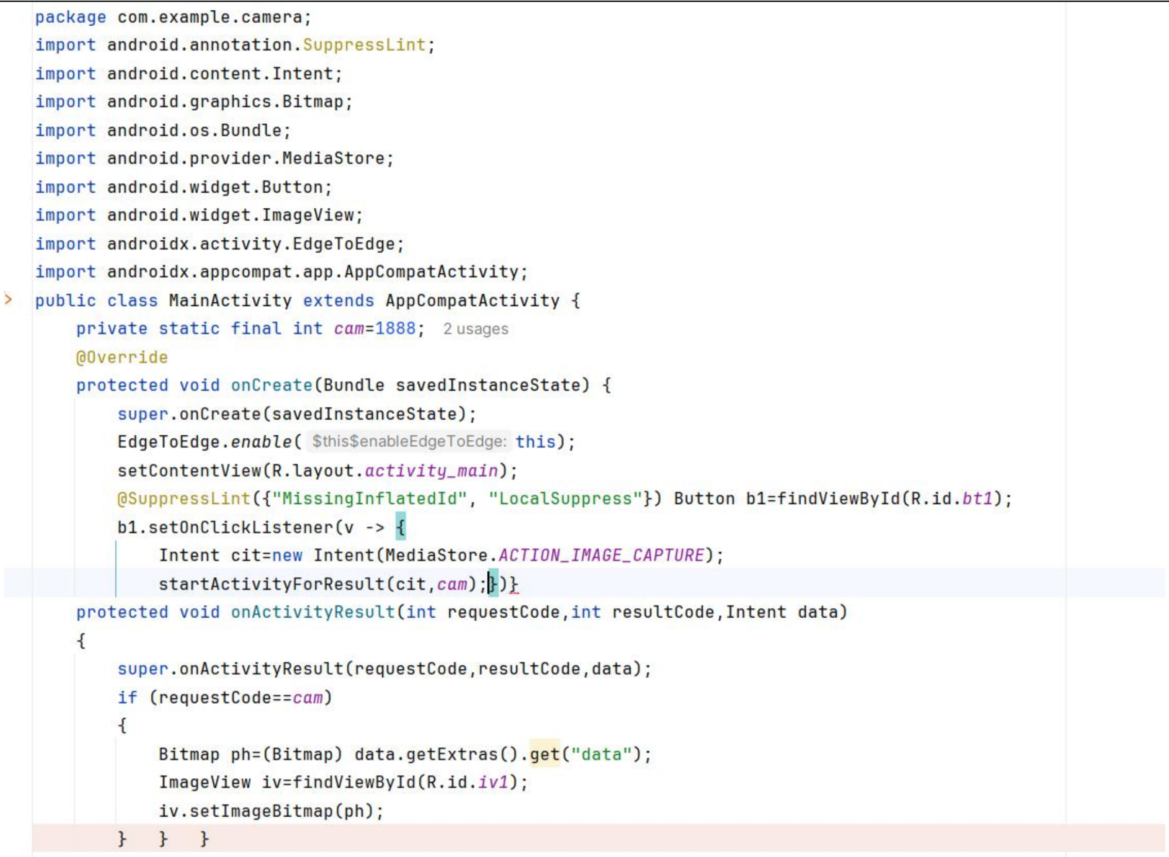
onStop (): Called once the activity is no longer visible. Time or CPU intensive shutdown operations, such as writing information to a database should be down in the onStop() method. This method is guaranteed to be called as of API 11.

onDestroy (): called before the activity is destroyed.

Develop a program to capture an image using camera and display it.



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/bt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open Camera"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.789" />
    <ImageView
        android:id="@+id/iv1"
        android:layout_width="366dp"
        android:layout_height="436dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.148"
        tools:ignore="ContentDescription,ImageContrastCheck"
        tools:srcCompat="@tools:sample/avatars" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



```java
package com.example.camera;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.widget.Button;
import android.widget.ImageView;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    private static final int cam=1888;  2 usages
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        @SuppressLint({"MissingInflatedId", "LocalSuppress"}) Button b1=findViewById(R.id.bt1);
        b1.setOnClickListener(v -> {
            Intent cit=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cit,cam);});}
    protected void onActivityResult(int requestCode,int resultCode,Intent data)
    {
        super.onActivityResult(requestCode,resultCode,data);
        if (requestCode==cam)
        {
            Bitmap ph=(Bitmap) data.getExtras().get("data");
            ImageView iv=findViewById(R.id.iv1);
            iv.setImageBitmap(ph);
        }   }   }
```

<uses-permission android:name=”android.permission.READ_EXTERNAL.STORAGE”/>
<uses-permission android:name=”android.permission.CAMERA”/>

Explain android security model.

The Android security model is primarily based on a sandbox and permission mechanism. Each application is running in a specific Dalvik virtual machine with a unique user ID assigned to it, which means the application code runs in isolation from the code of all other applications. Therefore, one application has not granted access to other applications' files.

Android application has been signed with a certificate with a private key Know the owner of the application is unique. This allows the author of the application will be identified if needed. When an application is installed in the phone is assigned a user ID, thus avoiding it from affecting it other applications by creating a sandbox for it. This user ID is permanent on which devices and applications with the same user ID are allowed to run in a single process. This is a way to ensure that a malicious application has Cannot access / compromise the data of the genuine application. It is mandatory for an application to list all the resources it will Access during installation. Terms are required of an application, in the installation process should be user-based or interactive Check with the signature of the application.

Declaring and Using Permissions

The purpose of a permission is to protect the privacy of an Android user. Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

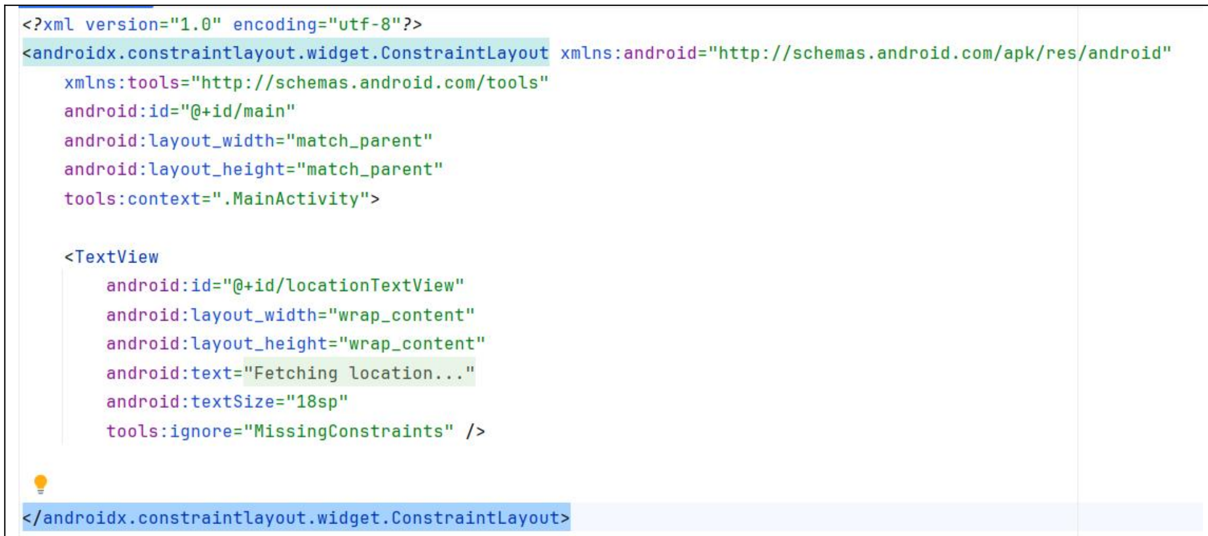
Permissions are divided into several protection levels. The protection level affects whether runtime permission requests are required. There are three protection levels that affect third-party apps: normal, signature, and dangerous permissions.

Normal permissions: Normal permissions cover areas where your app needs to access data or resources outside the app's sandbox, but where there's very little risk to the user's privacy or the operation of other apps. For example, permission to set the time zone is a normal permission. If an app declares in its manifest that it needs a normal permission, the system automatically grants the app that permission at install time. The system doesn't prompt the user to grant normal permissions, and users cannot revoke these permissions.

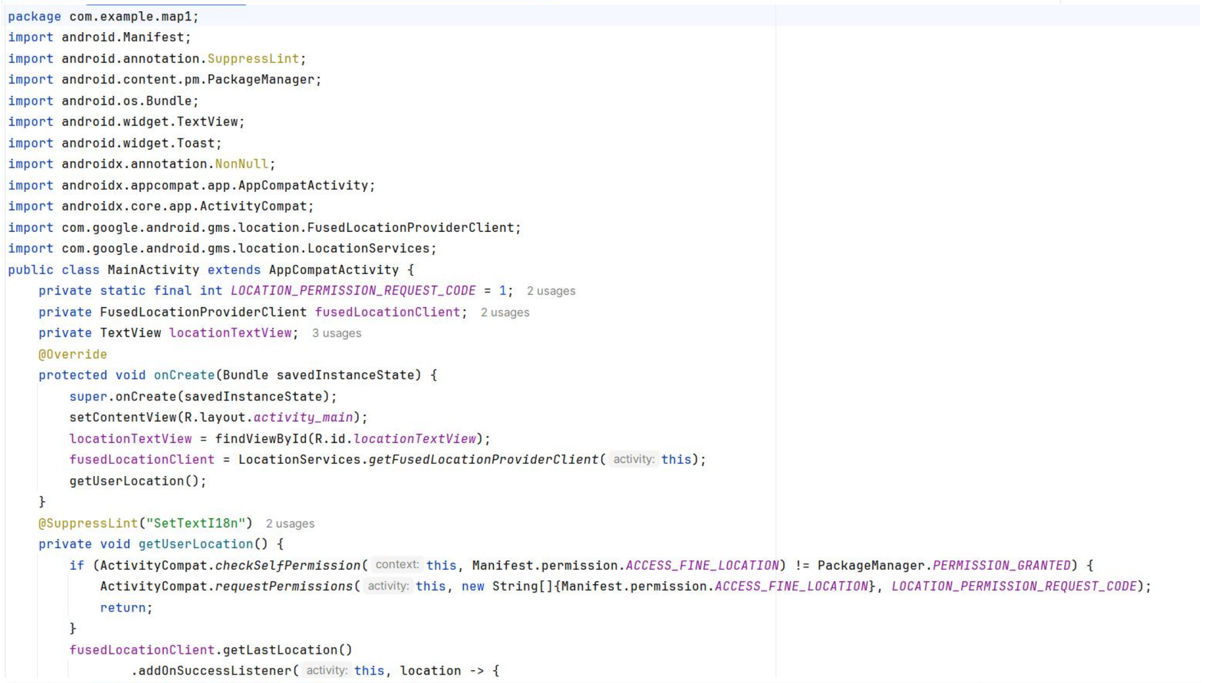
Signature permissions: The system grants these app permissions at install time, but only when the app that attempts to use permission is signed by the same certificate as the app that defines the permission.

Dangerous permissions: Dangerous permissions cover areas where the app wants data or resources that involve the user's private information, or could potentially affect the user's stored data or the operation of other apps. For example, the ability to read the user's contacts is a dangerous permission. If an app declares that it needs a dangerous permission, the user must explicitly grant the permission to the app. Until the user approves the permission, your app cannot provide functionality that depends on that permission. To use a dangerous permission, your app must prompt the user to grant permission at runtime. For more details about how the user is prompted, see Request prompt for dangerous permission.

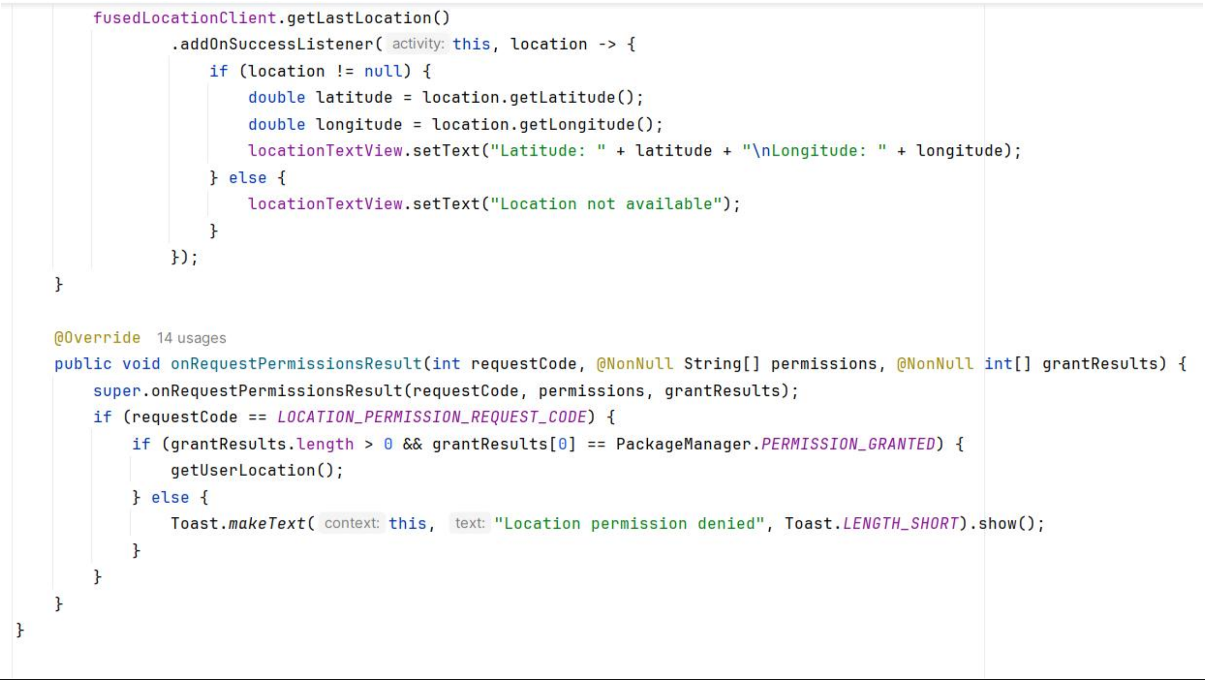
Develop a program to show users current location.



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/locationTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fetching location..."
        android:textSize="18sp"
        tools:ignore="MissingConstraints" />


</androidx.constraintlayout.widget.ConstraintLayout>
```



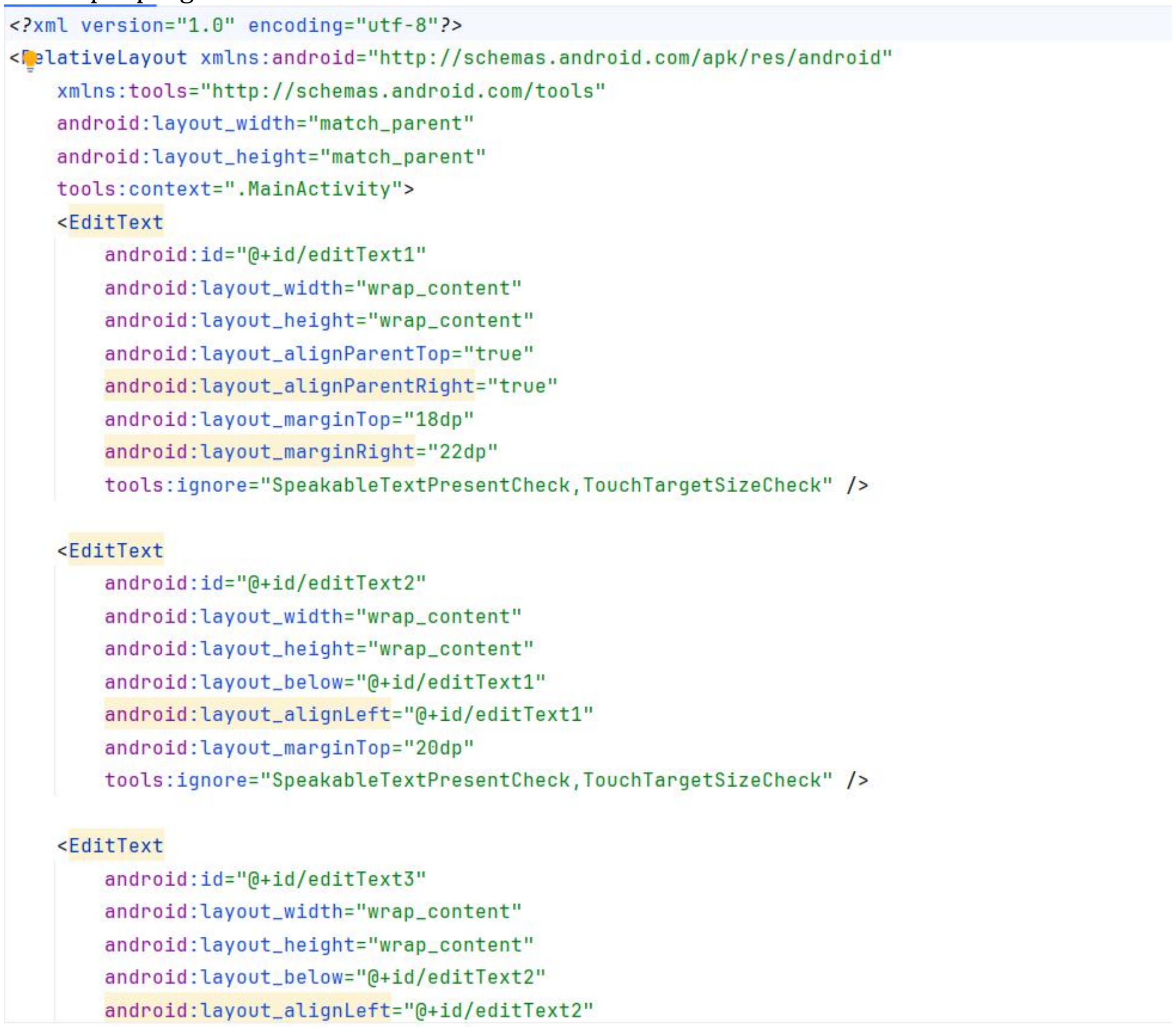
```java
package com.example.map1;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
public class MainActivity extends AppCompatActivity {
    private static final int LOCATION_PERMISSION_REQUEST_CODE = 1;  2 usages
    private FusedLocationProviderClient fusedLocationClient;  2 usages
    private TextView locationTextView;  3 usages
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        locationTextView = findViewById(R.id.locationTextView);
        fusedLocationClient = LocationServices.getFusedLocationProviderClient( activity: this);
        getUserLocation();
    }
    @SuppressLint("SetTextI18n")  2 usages
    private void getUserLocation() {
        if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_PERMISSION_REQUEST_CODE);
            return;
        }
        fusedLocationClient.getLastLocation()
                .addOnSuccessListener( activity: this, location -> {
```



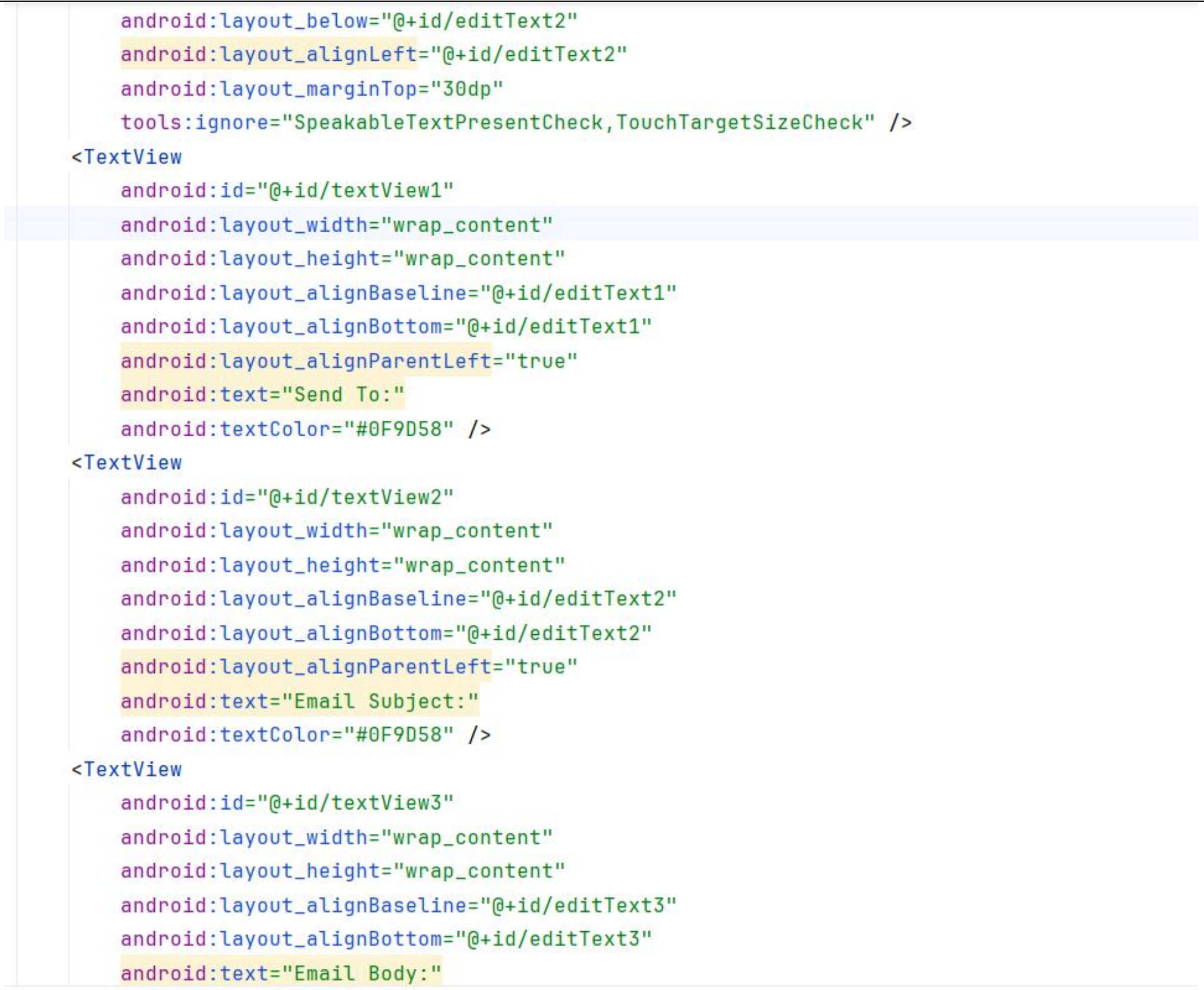
```java
        fusedLocationClient.getLastLocation()
                .addOnSuccessListener( activity: this, location -> {
                    if (location != null) {
                        double latitude = location.getLatitude();
                        double longitude = location.getLongitude();
                        locationTextView.setText("Latitude: " + latitude + "\nLongitude: " + longitude);
                    } else {
                        locationTextView.setText("Location not available");
                    }
                });
    }

    @Override  14 usages
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                getUserLocation();
            } else {
                Toast.makeText( context: this,  text: "Location permission denied", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

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

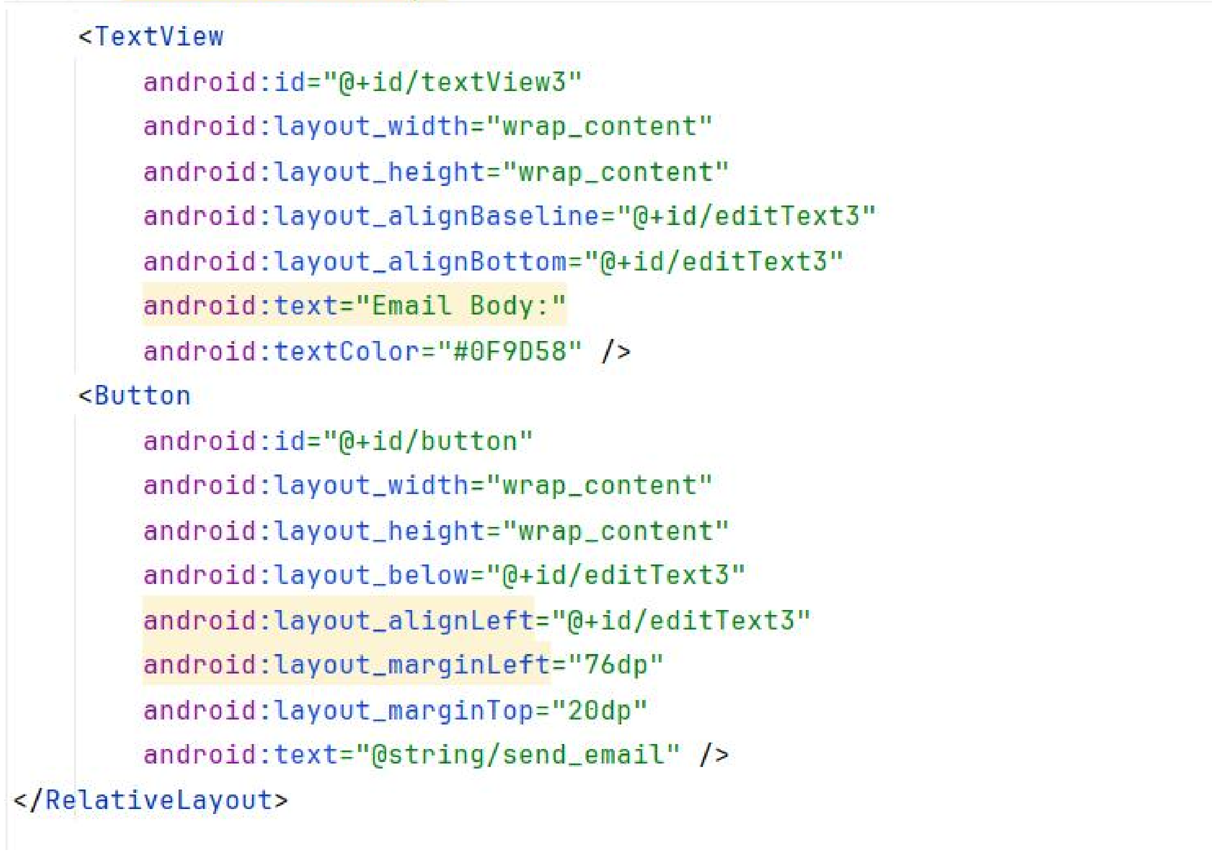
Develop a program to send e-mail.



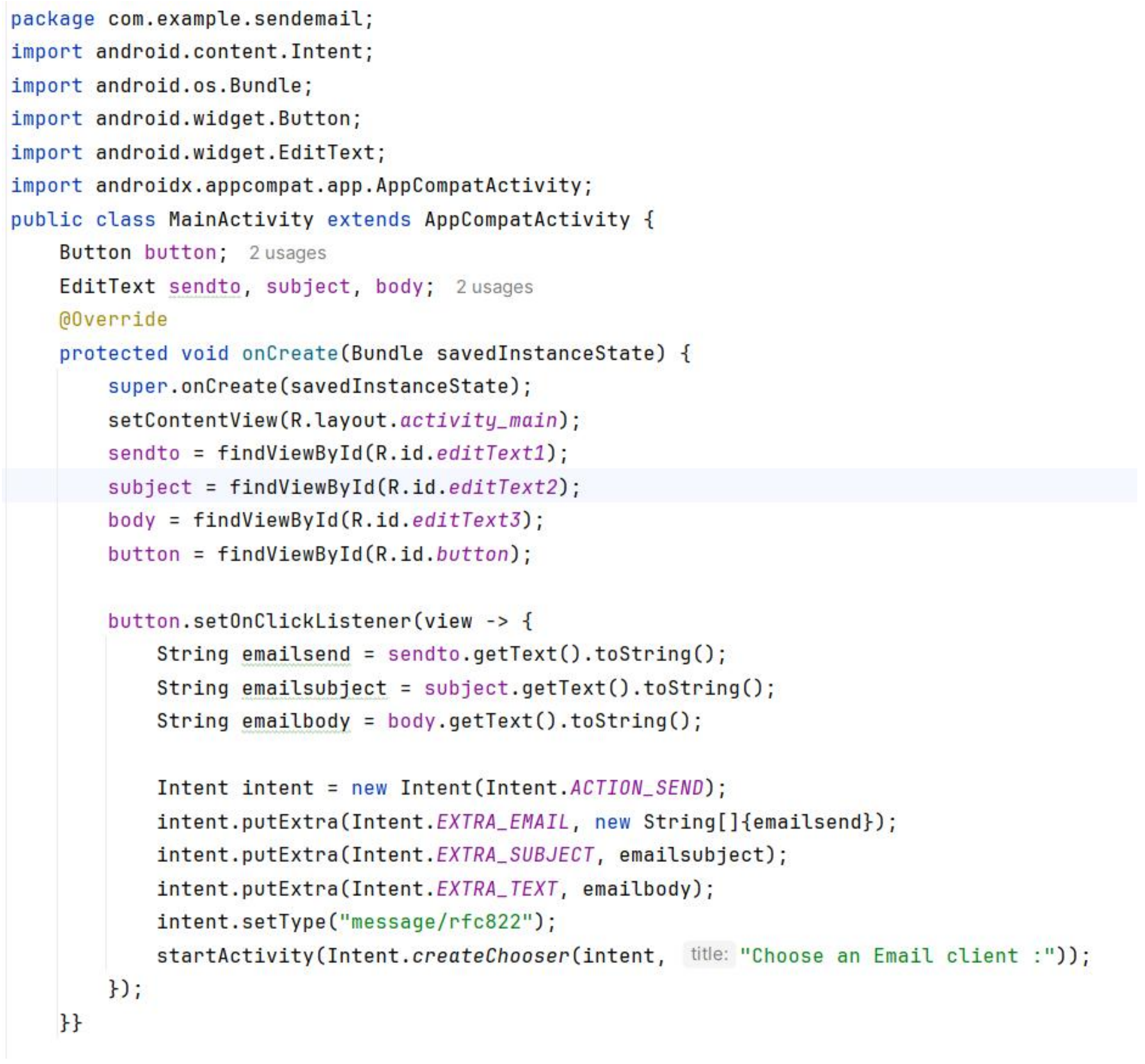
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_marginTop="18dp"
        android:layout_marginRight="22dp"
        tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_alignLeft="@+id/editText1"
        android:layout_marginTop="20dp"
        tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />

    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText2"
        android:layout_alignLeft="@+id/editText2"
```



```
        android:layout_below="@+id/editText2"
        android:layout_alignLeft="@+id/editText2"
        android:layout_marginTop="30dp"
        tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText1"
        android:layout_alignBottom="@+id/editText1"
        android:layout_alignParentLeft="true"
        android:text="Send To:"
        android:textColor="#0F9D58" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText2"
        android:layout_alignBottom="@+id/editText2"
        android:layout_alignParentLeft="true"
        android:text="Email Subject:"
        android:textColor="#0F9D58" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText3"
        android:layout_alignBottom="@+id/editText3"
        android:text="Email Body:"
```

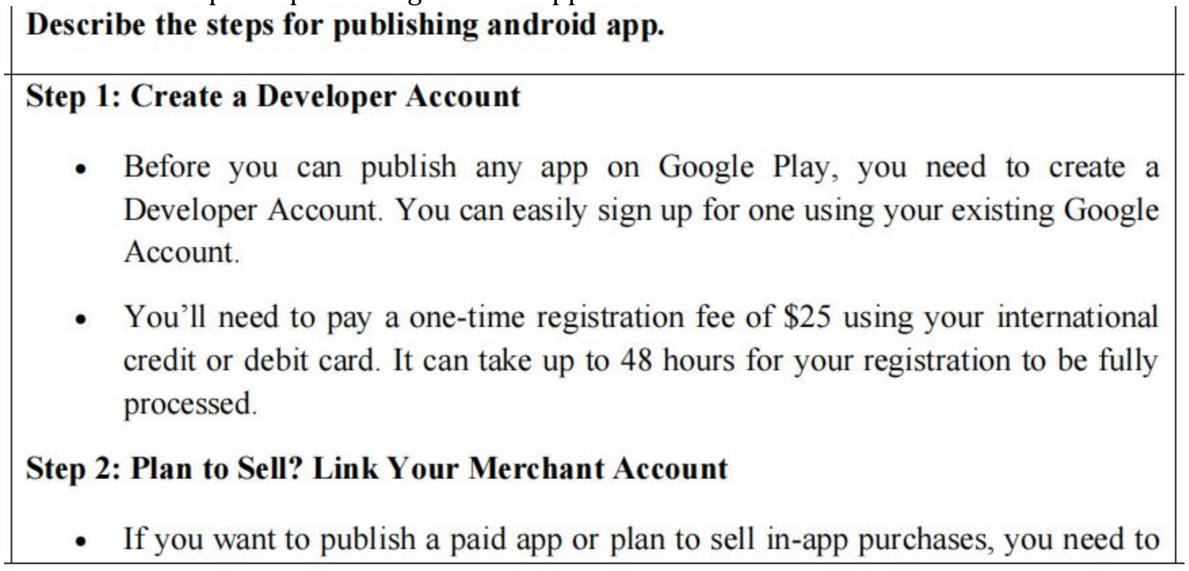


```
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText3"
        android:layout_alignBottom="@+id/editText3"
        android:text="Email Body:"
        android:textColor="#0F9D58" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText3"
        android:layout_alignLeft="@+id/editText3"
        android:layout_marginLeft="76dp"
        android:layout_marginTop="20dp"
        android:text="@string/send_email" />
</RelativeLayout>
```



```java
package com.example.sendemail;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    Button button;  2 usages
    EditText sendto, subject, body;  2 usages
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sendto = findViewById(R.id.editText1);
        subject = findViewById(R.id.editText2);
        body = findViewById(R.id.editText3);
        button = findViewById(R.id.button);

        button.setOnClickListener(view -> {
            String emailsend = sendto.getText().toString();
            String emailsubject = subject.getText().toString();
            String emailbody = body.getText().toString();

            Intent intent = new Intent(Intent.ACTION_SEND);
            intent.putExtra(Intent.EXTRA_EMAIL, new String[]{emailsend});
            intent.putExtra(Intent.EXTRA_SUBJECT, emailsubject);
            intent.putExtra(Intent.EXTRA_TEXT, emailbody);
            intent.setType("message/rfc822");
            startActivity(Intent.createChooser(intent,  title: "Choose an Email client :"));
        });
    }}
```

Describe the steps for publishing android app.



Describe the steps for publishing android app.

Step 1: Create a Developer Account

- Before you can publish any app on Google Play, you need to create a Developer Account. You can easily sign up for one using your existing Google Account.
- You'll need to pay a one-time registration fee of $25 using your international credit or debit card. It can take up to 48 hours for your registration to be fully processed.

Step 2: Plan to Sell? Link Your Merchant Account

- If you want to publish a paid app or plan to sell in-app purchases, you need to create a payments center profile, i.e. a merchant account.

- A merchant account will let you manage your app sales and monthly payouts, as well as analyze your sales reports right in your Play Console.

Step 3: Create an App

- Now you have create an application by clicking on 'Create Application'. Here you have to select your app's default language from the drop-down menu and then type in a title for your app. The title of your app will show on Google Play after you've published.

Step 4: Prepare Store Listing

- Before you can publish your app, you need to prepare its store listing. These are all the details that will show up to customers on your app's listing on Google Play. You not necessarily complete it at once , you can always save a draft and revisit it later when you're ready to publish.
- The information required for your store listing is divided into several categories such as Product Details containing title, short and full description of the app, Your app's title and description should be written with a great user experience in mind. Use the right keywords, but don't overdo it. Make sure your app doesn't come across as spam-y or promotional, or it will risk getting suspended on the Play Store.
- Graphic Assets where you can add screenshots, images, videos, promotional graphics, and icons that showcase your app's features and functionality.

Languages & Translations, Categorization where in category can be selected to which your app belong to. **Contact Details, Privacy Policy** for apps that request access to sensitive user data or permissions, you need to enter a comprehensive privacy policy that effectively discloses how your app collects, uses, and shares that data.

Step 5: Upload APK to an App Release

Finally upload your app, by uploading APK file. Before you upload APK, you need to create an app release. You need to select the type of release you want to upload your first app version to. You can choose between an internal test, a closed test, an open test, and a production release. The first three releases allow you to test out your app among a select group of users before you make it go live for everyone to access.

This is a safer option because you can analyze the test results and optimize or fix your app accordingly if you need to before rolling it out to all users.

Once you create a production release, your uploaded app version will become accessible to everyone in the countries you choose to distribute it in and click on 'Create release.'

Step 6: Provide an Appropriate Content Rating

If you don't assign a rating to your app, it will be listed as 'Unrated'. Apps that are 'Unrated' may get removed from Google Play.

To rate your app, you need to fill out a content rating questionnaire An appropriate content rating will also help you get to the right audience, which will eventually improve your engagement rates.

Step 7: Set Up Pricing & Distribution

Before you can fill out the details required in this step, you need to determine your app's monetization strategy. Once you know how your app is going to make money, you can go ahead and set up your app as free or paid.

You can always change your app from paid to free later, but you cannot change a free app to paid. For that, you'll need to create a new app and set its price.

Step 8: Rollout Release to Publish Your App

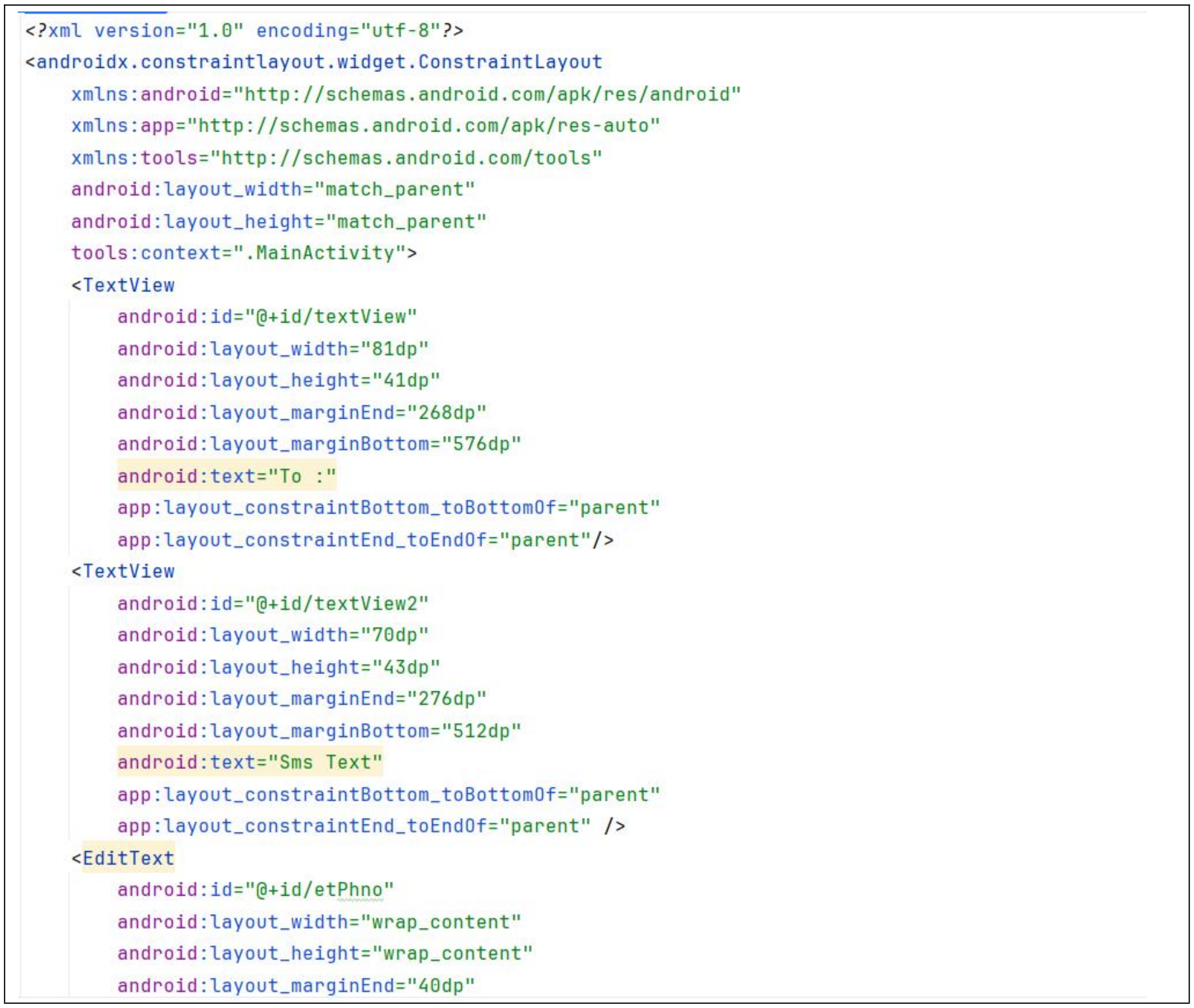
The final step involves reviewing and rolling out your release after making sure you've taken care of everything else.

Before you review and rollout your release, make sure the store listing, content rating, and pricing and distribution sections of your app each have a green check mark next to them.

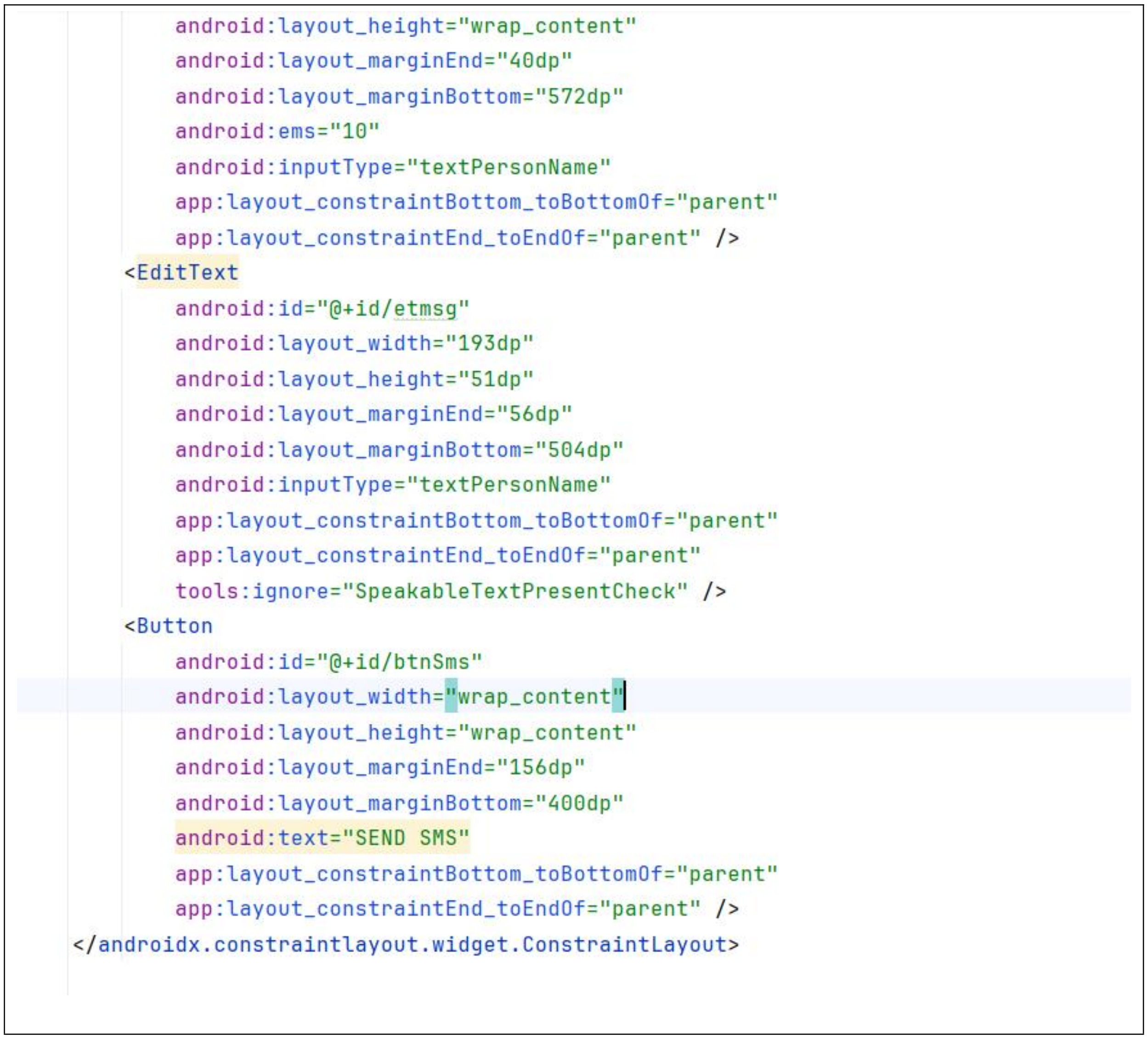
Once you're sure about the correctness of the details, select your app and navigate to 'Release management' – 'App releases.' You can always opt for reviews by clicking on 'Review' to be taken to the 'Review and rollout release' screen. Here, you can see if there are any issues or warnings you might have missed out on.

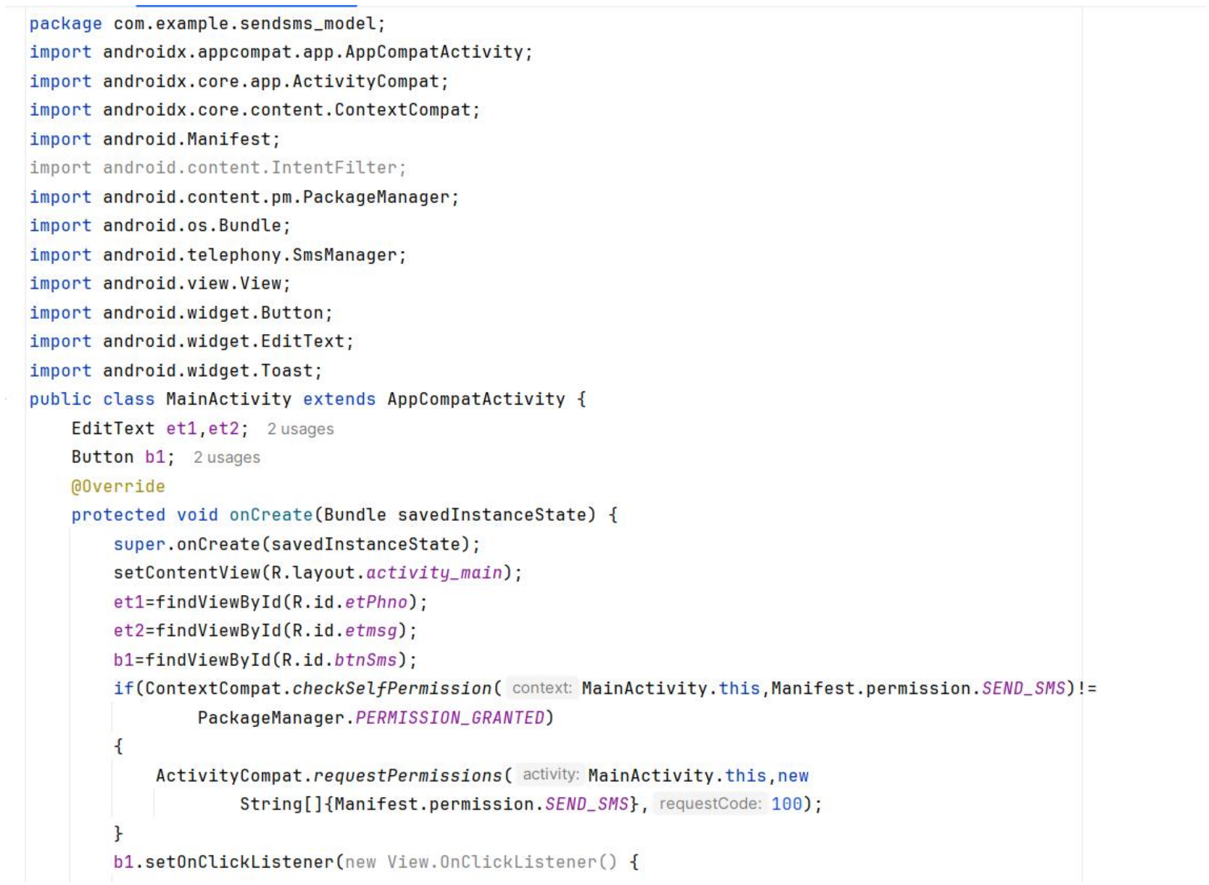
Finally, select 'Confirm rollout.' This will also publish your app to all users in your target countries on Google Play.

Develop and application to send SMS (Design minimal UI as per your choice. Write XML, Java and manifest file).

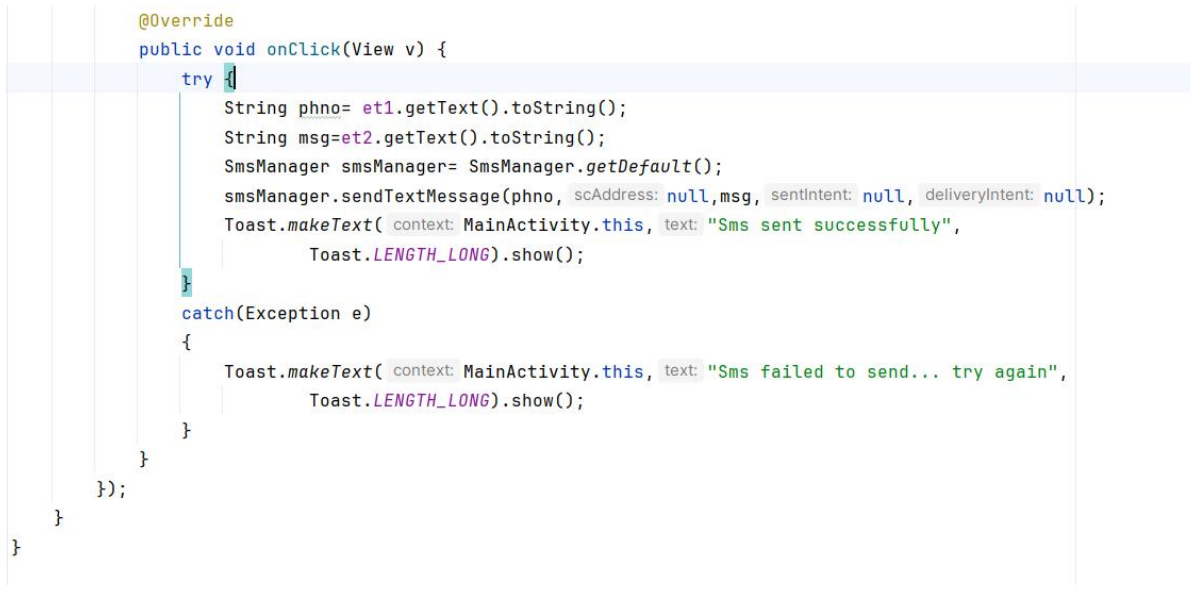


```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="81dp"
        android:layout_height="41dp"
        android:layout_marginEnd="268dp"
        android:layout_marginBottom="576dp"
        android:text="To :"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="70dp"
        android:layout_height="43dp"
        android:layout_marginEnd="276dp"
        android:layout_marginBottom="512dp"
        android:text="Sms Text"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
    <EditText
        android:id="@+id/etPhno"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="40dp"
```


```
        android:layout_height="wrap_content"
        android:layout_marginEnd="40dp"
        android:layout_marginBottom="572dp"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
    <EditText
        android:id="@+id/etmsg"
        android:layout_width="193dp"
        android:layout_height="51dp"
        android:layout_marginEnd="56dp"
        android:layout_marginBottom="504dp"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        tools:ignore="SpeakableTextPresentCheck" />
    <Button
        android:id="@+id/btnSms"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="156dp"
        android:layout_marginBottom="400dp"
        android:text="SEND SMS"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



```
package com.example.sendsms_model;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    EditText et1,et2;  2 usages
    Button b1;  2 usages
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et1=findViewById(R.id.etPhno);
        et2=findViewById(R.id.etmsg);
        b1=findViewById(R.id.btnSms);
        if(ContextCompat.checkSelfPermission( context: MainActivity.this,Manifest.permission.SEND_SMS)!=
                PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions( activity: MainActivity.this,new
                    String[]{Manifest.permission.SEND_SMS}, requestCode: 100);
        }
        b1.setOnClickListener(new View.OnClickListener() {
```

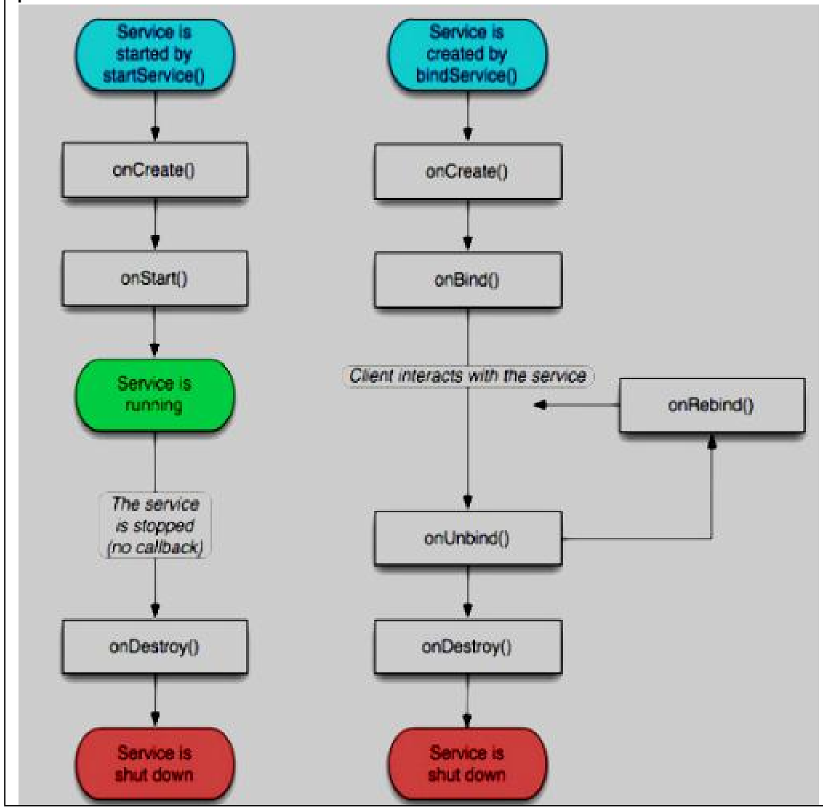


```
            @Override
            public void onClick(View v) {
                try {
                    String phno= et1.getText().toString();
                    String msg=et2.getText().toString();
                    SmsManager smsManager= SmsManager.getDefault();
                    smsManager.sendTextMessage(phno, scAddress: null,msg, sentIntent: null, deliveryIntent: null);
                    Toast.makeText( context: MainActivity.this, text: "Sms sent successfully",
                            Toast.LENGTH_LONG).show();
                }
                catch(Exception e)
                {
                    Toast.makeText( context: MainActivity.this, text: "Sms failed to send... try again",
                            Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

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

Explain Service Life Cycle.

- A service is an application component which runs without direst interaction with the user in the background.
- Services are used for repetitive and potentially long running operations, i.e., Internet downloads, checking for new data, data processing, updating content providers and the like.
- Service can either be started or bound we just need to call either startService() or bindService() from any of our android components. Based on how our service was started it will either be "started" or "bound"


Service is started by startService()
onCreate()
onStart()
Service is running
The service is stopped (no callback)
onDestroy()
Service is shut down
Service is created by bindService()
onCreate()
onBind()
Client interacts with the service
onRebind()
onUnbind()
onDestroy()
Service is shut down

Service Lifecycle

1. **Started**
   a. A service is started when an application component, such as an activity, starts it by calling startService().
   b. Now the service can run in the background indefinitely, even if the component that started it is destroyed.
2. **Bound**
   a. A service is bound when an application component binds to it by calling bindService().
   b. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with InterProcess Communication (IPC).
   c. Like any other components service also has callback methods. These will be invoked while the service is running to inform the application of its state. Implementing these in our custom service would help you in performing the right operation in the right state. •
   d. There is always only a single instance of service running in the app. If you are calling startService() for a single service multiple times in our application it just invokes the onStartCommand() on that service. Neither is the service restarted multiple times nor are its multiple instances created

1. **onCreate():**
   This is the first callback which will be invoked when any component starts the service. If the same service is called again while it is still running this method wont be invoked. Ideally one time setup and intializing should be done in this callback.
2. **onStartCommand() /startSetvice()**
   This callback is invoked when service is started by any component by calling **startService().** It basically indicates that the service has started and can now run indefinetly.
3. **onBind()**
   To provide binding for a service, you must implement the onBind() callback method. This method returns an IBinder object that defines the programming interface that clients can use to interact with the service.
4. **onUnbind()**
   This is invoked when all the clients are disconnected from the service.
5. **onRebind()**
   This is invoked when new clients are connected to the service. It is called after onRebind
6. **onDestroy()**
   This is a final clean up call from the system. This is invoked just before the service is being destroyed.

Describe sensors use in Android with example.

Most of the android devices have built-in sensors that measure motion, orientation, and various environmental condition. The android platform supports three broad categories of sensors.

1. Motion Sensors

These are used to measure acceleration forces and rotational forces along with three axes.

2. Environmental sensors

These are used to measure the environmental changes such as temperature, humidity etc.

3. Position sensors

These are used to measure the physical position of device.

Example:

TYPE_ACCELE_ROMETER

TYPE_GRAVITY

TYPE_LIGHT

TYPE_ORIENTATION

TYPE_PRESSURE

Some of the sensors are hardware based and some are software based sensors. Whatever the sensor is, android allows us to get the raw data from these sensors and use it in our application.

Android provides SensorManager and Sensor classes to use the sensors in our application.

1) **SensorManager class**

The android.hardware.SensorManager class provides methods :

o to get sensor instance,

o to access and list sensors,

o to register and unregister sensor listeners etc.

You can get the instance of SensorManager by calling the method getSystemService() and passing the SENSOR_SERVICE constant in it.

SensorManager sm = (SensorManager)getSystemService(SENSOR_SERVICE);

2) **Sensor class**

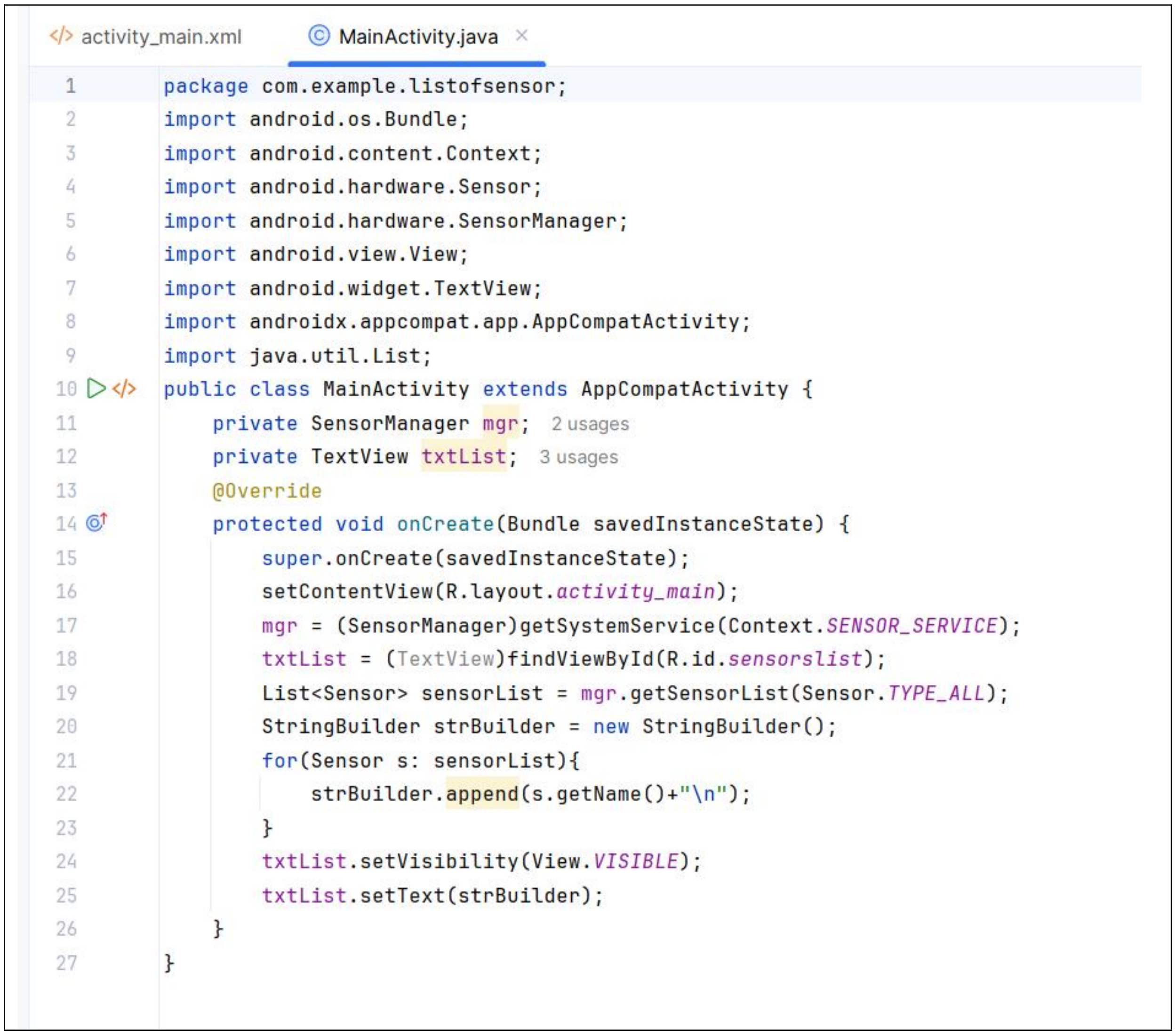
The android.hardware.Sensor class provides methods to get information of the sensor such as sensor name, sensor type, sensor resolution, sensor type etc.

3) **SensorEvent class**

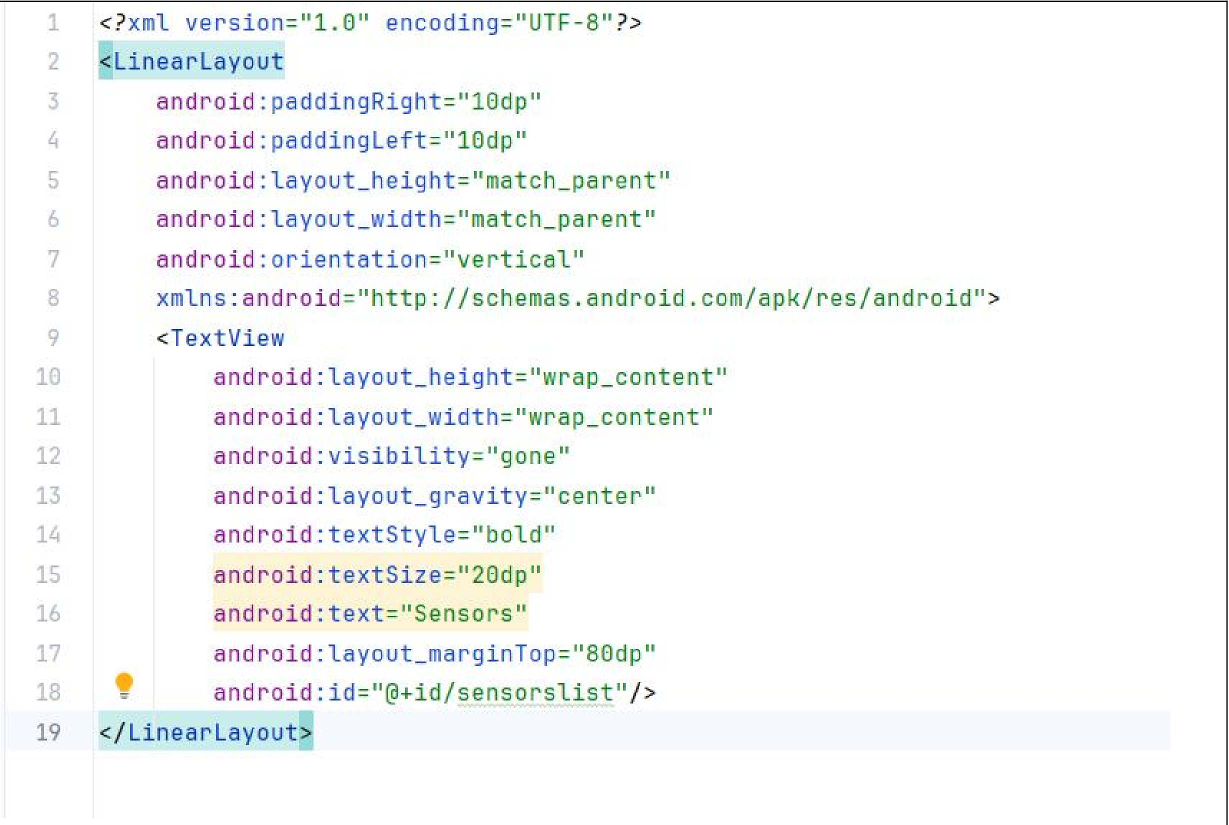
Its instance is created by the system. It provides information about the sensor.

4) **SensorEventListener interface**

It provides two call back methods to get information when sensor values (x,y and z) change or sensor accuracy changes.



```java
package com.example.listofsensor;
import android.os.Bundle;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.view.View;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.List;
public class MainActivity extends AppCompatActivity {
    private SensorManager mgr;  2 usages
    private TextView txtList;  3 usages
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mgr = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
        txtList = (TextView)findViewById(R.id.sensorslist);
        List<Sensor> sensorList = mgr.getSensorList(Sensor.TYPE_ALL);
        StringBuilder strBuilder = new StringBuilder();
        for(Sensor s: sensorList){
            strBuilder.append(s.getName()+"\n");
        }
        txtList.setVisibility(View.VISIBLE);
        txtList.setText(strBuilder);
    }
}
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout
    android:paddingRight="10dp"
    android:paddingLeft="10dp"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:visibility="gone"
        android:layout_gravity="center"
        android:textStyle="bold"
        android:textSize="20dp"
        android:text="Sensors"
        android:layout_marginTop="80dp"
        android:id="@+id/sensorslist"/>
</LinearLayout>
```

Develop a program to TURN ON and OFF Bluetooth. Write .java file and permission tags.

Code of MainActivity.java

```
package com.example.bluetooth.myapplication;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;

import android.widget.Toast;
import java.util.ArrayList;
import java.util.Set;

public class MainActivity extends Activity  {
   Button b1,b2,b3,b4;
   private BluetoothAdapter BA;
   private Set<BluetoothDevice>pairedDevices;
   ListView lv;
```

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

    b1 = (Button) findViewById(R.id.button);
    b2=(Button)findViewById(R.id.button2);
    b3=(Button)findViewById(R.id.button3);
    b4=(Button)findViewById(R.id.button4);

    BA = BluetoothAdapter.getDefaultAdapter();
    lv = (ListView)findViewById(R.id.listView);
  }

  public void on(View v){
    if (!BA.isEnabled()) {
      Intent turnOn = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(turnOn, 0);
      Toast.makeText(getApplicationContext(), "Turned
on",Toast.LENGTH_LONG).show();
    } else {
      Toast.makeText(getApplicationContext(), "Already on",
Toast.LENGTH_LONG).show();
    }
  }
  public void off(View v){
    BA.disable();
    Toast.makeText(getApplicationContext(),                    "Turned                    off"
,Toast.LENGTH_LONG).show();
  }


  public  void visible(View v){
    Intent                          getVisible                               =                               new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
    startActivityForResult(getVisible, 0);
  }

  public void list(View v){
    pairedDevices = BA.getBondedDevices();

    ArrayList list = new ArrayList();

    for(BluetoothDevice bt : pairedDevices) list.add(bt.getName());
    Toast.makeText(getApplicationContext(),                "Showing                    Paired
Devices",Toast.LENGTH_SHORT).show();
```

```
    final                ArrayAdapter                adapter                =                new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);

    lv.setAdapter(adapter);
  }
}
```

Permission Tags

AndroidManifest.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.bluetooth.myapplication" >
  <uses-permission android:name="android.permission.BLUETOOTH"/>
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>

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

    <activity
      android:name=".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>
```