TOP ANDROID INTERVIEW QUESTIONS AND ANSWERS

FOR BEGINNER & EXPERIANCED

Android Authority

Android Interview Questions and Answers:

Reason for OutOfMemory

In order to prevent it, first we need to know the reasons of "Out of Memory" exception:

- The biggest reason is memory leak i.e, Context leaking or can say Activity Leaking, a Service has the same problems as Activity in this regard.
- You are doing the process that demands continuous memory and at a point, it goes beyond max memory limit of a process.
- When you are dealing with large Bitmap and load all of them at runtime.

Serialization & Transient?

Before understanding the transient keyword, one has to understand the concept of serialization. If the reader knows about serialization, please skip the first point.

- **What is serialization?**
  Serialization is the process of making the object's state persistent. That means the state of the object is converted into a stream of bytes and stored in a file. In the same way, we can use the deserialization to bring back the object's state from bytes. This is one of the important concepts in Java programming because serialization is mostly used in networking programming. The objects that need to be transmitted through the network have to be converted into bytes. For that purpose, every class or interface must implement the Serializable interface. It is a marker interface without any methods.
- **Now what is the transient keyword and its purpose?**
  By default, all of object's variables get converted into a persistent state. In some cases, you may want to avoid persisting some variables because you don't have the need to persist those variables. So you can declare those variables as transient. If the variable is declared as transient, then it will not be persisted. That is the main purpose of the transient keyword.

I want to explain the above two points with the following example:
package javabeat.samples;

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
class NameStore implements Serializable{
private String firstName;
```

```
private transient String middleName;
private String lastName;
`public NameStore (String fName, String mName, String lName){`
   `this.firstName = fName;`
   `this.middleName = mName;`
   `this.lastName = lName;`
`}`

`public String toString(){`
   `StringBuffer sb = new StringBuffer(40);`
   `sb.append("First Name : ");`
   `sb.append(this.firstName);`
   `sb.append("Middle Name : ");`
   `sb.append(this.middleName);`
   `sb.append("Last Name : ");`
   `sb.append(this.lastName);`
   `return sb.toString();`
`}`
}
public class TransientExample
{
public static void main(String args[]) throws Exception
{
NameStore nameStore = new NameStore("Steve", "Middle","Jobs");
   `ObjectOutputStream o = new ObjectOutputStream(new
FileOutputStream("nameStore"));`

   `o.writeObject(nameStore);`

   `o.close();`

   `ObjectInputStream in = new ObjectInputStream(new
FileInputStream("nameStore"));`

   `NameStore nameStore1 = (NameStore)in.readObject();`

   `System.out.println(nameStore1);`

`}`
}
```

what is ServiceConnection and why to use?

ServiceConnection android.content.ServiceConnection is an interface which is used to monitor the state of service. We need to override following methods.
onServiceConnected(ComponentName name, IBinder service) : This is called when service is connected to the application.
onServiceDisconnected(ComponentName name) : This is called when service is disconnected.

What is DDMS and what can it do?

- DDMS is short for Dalvik Debug Monitor Server. It ships natively with Android and contains a number of useful debugging features including: location data spoofing, port-forwarding, network traffic tracking, incoming call/SMS spoofing, thread and heap information, screen capture, and the ability to simulate network state, speed, and latenc

What is START_STICKY and START_NOT_STICKY Service?

- Both codes are only relevant when the phone runs out of memory and kills the service before it finishes executing. START_STICKY tells the OS to recreate the service after it has enough memory and call onStartCommand() again with a null intent. START_NOT_STICKY tells the OS to not bother recreating the service again. There is also a third code START_REDELIVER_INTENT that tells the OS to recreate the service and redeliver the same intent to onStartCommand().

How to handle screen rotation?

When you rotate your device, your present activity gets completely destroyed, ie goes through

- onSaveInstanceState()
- onPause()
- onStop()
- onDestroy() and a new activity is created completely which goes through
- onCreate()
- onStart()
- onRestoreInstanceState()
  The Two Methods in the bold, onSaveInstanceState() saves the instance of the present activity which is going to be destroyed. onRestoreInstanceState This method restores the saved state of the previous activity. This way you don't lose your previous state of the app.
  Here is how you use these methods.
  @Override public void onSaveInstanceState(Bundle outState, PersistableBundle outPersistentState) { super.onSaveInstanceState(outState, outPersistentState);
- `outState.putString("theWord", theWord); // Saving the Variable theWord`
- `outState.putStringArrayList("fiveDefns", fiveDefns); // Saving the ArrayList fiveDefns`

  }
  @Override public void onRestoreInstanceState(Bundle savedInstanceState, PersistableBundle persistentState)
  { super.onRestoreInstanceState(savedInstanceState, persistentState);
  `theWord = savedInstanceState.getString("theWord"); // Restoring theWord`
  `fiveDefns = savedInstanceState.getStringArrayList("fiveDefns"); //Restoring fiveDefns`
  }

what is android?

- Open source OS
- based on linux kernal-(not linux os)
- NOT programming language

what can we use for SQLite Security?

- SQLCipher

what is Intent?

- It is messaging object.
- It uses when you want to pass **data** from one screen to another.

Types -

1. Explicit Where you have to specify target component. ex. Intent i=new Intent(ActivityOne.this,ActivityTwo.class)
2. Implicit- where you dont have to specify target component,System specifies what action has to be done. ex - Intent chooser=Intent.createChooser(send,title)

What is ANR?
When the application tries to run a long operation on the main thread, but gets no response after a long time.

which are orientations?

- vertical
- Horizontal

What is ADB(Android Debug Bridge)?
It provides a connection between System and Emulator.

What Android Architecture Consist of?

- Android Architecture - Camera Calculator

- Framework - Content Provider ,View System,Managers-(Activity,Location,Package,Notification)
- Libraries -
- Linux Kernal

Difference between Activity and Service?

- Activity -Runs always on Foreground
- Service -Runs on Background,Useful to run the long process.

what is Manifest File?

- Play important role in app
- where we declare all permissions-location, internet.
- declare all Services
- declare all Activities

which api you worked on?

- Analytics
- Cloud Messaging
- Authentication
- Realtime Database
- Storage
- Performance Monitoring
- Remote Config
- Test Lab
- Sign in with Google
- Maps
- Places
- cleverTap

Difference between ArrayAdapter and BaseAdapter.

- **BaseAdapter** as the name suggests, is a base class for all the adapters.
- When you are extending the Base adapter class you need to implement all the methods like getCount(), getId() etc.
- **ArrayAdapter** is a class which can work with array of data. You need to override only getview() method.
- **ListAdapter** is a an interface implemented by concrete adapter classes.
- BaseAdapter is an abstract class whereas ArrayAdapter and ListAdapter are the concrete classes.
- ArrayAdapter and ListAdapter classes are developed since in general we deal with the array data sets and list data sets.

Difference between RecyclerView and ListView?

RecyclerView was created as a ListView improvement, so yes, you can create an attached list with ListView control, but using RecyclerView is easier as it

Reuses cells while scrolling up/down - this is possible with implementing View Holder in the listView adapter, but it was an optional thing, while in the RecycleView it's the default way of writing adapter. Decouples list from its container - so you can put list items easily at run time in the different containers (linearLayout, gridLayout) with setting LayoutManager.
Example:

```
mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
//or
mRecyclerView.setLayoutManager(new GridLayoutManager(this, 2));
```

Animates common list actions - Animations are decoupled and delegated to ItemAnimator.

What is RecyclerView?

- The RecyclerView widget is a more advanced and flexible version of ListView.

Why RecyclerView?

- RecyclerView is a container for displaying large data sets that can be scrolled very efficiently by maintaining a limited number of views.

When you should use RecyclerView?

- You can use the RecyclerView widget when you have data collections whose elements changes at runtime based on user action or network events

new extended functionalities available in RecyclerView that are not directly supported by ListView as following:

- Direct horizontal scrolling is not supported in ListView. Whereas in RecyclerView you can easily implement horizontal and vertical scrolling.
- ListView supports only simple linear list view with vertical scrolling. Whereas RecyclerView supports three types of lists using RecyclerView.LayoutManager. 1) StaggeredGridLayoutManager 2) GridLayoutManager, 3) LinearLayoutManager. I will give you brief introduction about all these 3 lists later.
- ListView gives you an option to add divider using dividerHeight parameter whereas RecyclerView enable you to customize divider (spacing) between two element

Name 4 ways Android allows you to store data?

Any of the following 5 possible options are acceptable:

- SharedPreferences
- Internal Storage
- External Storage
- SQLite Database
- Network connection

What items or folders are important in every Android project?

Answer: The developer should name at least 4 of these 6 items below, as these are essential within each Android project:

- AndroidManifest.xml
- build.xml
- bin/
- src/
- res/
- assets/

What are containers?

Containers hold objects and widgets together, depending on which items are needed and in what arrangement they need to be in. Containers may hold labels, fields, buttons, or even child containers, as examples.

What information do you need before you begin coding an Android app for a client?

You want to find out that this person will seek to truly understand what you are trying to accomplish with your app, and the functionality. The following items are good to hear:

- Objective statement or purpose of the app for the app publisher
- Description of the target audience or user demographics
- Any existing apps that it might be similar to
- Wireframes
- Artwork; The best developers will say they require the artwork to be completed before development. This avoids delays, and helps the developer understand the look, feel and branding you are trying to achieve .

What is Service?

- A service is a component which runs in the background without direct interaction with the user
- service has no user interface, it is not bound to the lifecycle of an activity.
- 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.

States of Services?

1. Started- A service is started when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
2. Bound- A service is bound when an application component binds to it by calling bindService(). 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).

Different Type of Services?

These are the three different types of services:

1. Foreground

A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services must display a status bar icon. Foreground services continue running even when the user isn't interacting with the app.

2. Background

A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service. Note: If your app targets API level 26 or higher, the system imposes restrictions on running background services when the app itself is not in the foreground. In most cases like this, your app should use a scheduled job instead.

3. Bound

A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed

How to start service again after reboot device in Android?

You can use broadcast receiver to restart service, <action android:name="android.intent.action.BOOT_COMPLETED" /> BOOT_COMPLETED event we can use to identify system is booted which calls broadcast receiver and broadcast receiver will startservice again.

What is BroadCast Reciever?

- Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents **to make BroadcastReceiver works for the system broadcasted intents** −
- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

Creating the Broadcast Receiver

- A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as a Intent object parameter.

```
public class MyReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
}
}
```

Registering Broadcast Receiver

An application listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file.

<application android:icon="@drawable/ic_launcher" android:label="@string/app_name"android:theme="@style/AppTheme" > <receiver android:name="MyReceiver">

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

</receiver> </application>

What is Content Provider?

- A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class.
- A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network.

onCreate() This method is called when the provider is started.

query() This method receives a request from a client. The result is returned as a Cursor object.

insert() This method inserts a new record into the content provider.

delete() This method deletes an existing record from the content provider.

update() This method updates an existing record from the content provider.

getType() This method returns the MIME type of the data at the given URI.

Android OS Naughat Changes?

- Multi-window Support
- Notification Bar Toggles
- Notifications: redesigned, bundled and Quick Reply-able
- Notification prioritization
- Customizable Quick Settings
- Doze Mode on the Go
- Multi-language support, emoji and app links
- New Settings menu - The essential information contained in each Settings section is now displayed on the main page.
- Do Not Disturb
- Data Saver - Data Saver denies internet access to background apps when you're connected to cellular data.
- Seamless updates
- File-based encryption
- Emergency info

What are the new features of oreo?

AutoFill:
With your permission, AutoFill remembers your logins to get you into your favourite apps at supersonic speed. **Picture-in-picture:**
Allows you to see two apps at once. It's like having super strength and laser vision.
Dive into more apps with fewer taps

- Notification dots:
  Press the notification dots to quickly see what's new, and easily clear them by swiping away.

Android Instant Apps:

Teleport directly into new apps straight from your browser, no installation needed.

Activity LifeCycle?

** What is the difference between Activity and AppCompatActivity?**

- AppCompatActivity provides native ActionBar support that is consistent across the application. Also it provides backward compatibility for other material design components till SDK version 7(ActionBar was natively available since SDK 11). Extending an Activity doesn't provide any of these.

Activity, AppCompatActivity, FragmentActivity and ActionBarActivity. How are they related?

- Activity is the base class. FragmentActivity extends Activity. AppCompatActivity extends FragmentActivity. ActionBarActivity extends AppCompatActivity.
- FragmentActivity is used for fragments.
- Since the build version 22.1.0 of the support library, ActionBarActivity is deprecated. It was the base class of appcompat-v7.
- At present, AppCompatActivity is the base class of the support library. It has come up with many new features like ToolBar, tinted widgets, material design color pallets etc.

Describe the structure of Android Support Library?

- Android Support Library is not just a single library. It's a collection of libraries that have different naming conventions and usages. On a higher level, it's divided into three types;
- **Compatibility Libraries**: These focus on back porting features so that older frameworks can take advantage of newer releases. The major libraries include v4 and v7-appcompat. v4 includes classes like DrawerLayout and ViewPager while appcompat-v7 provides classes for support ActionBar and ToolBar.

- **Component Libraries**: These include libraries of certain modules that don't depend on other support library dependencies. They can be easily added or removed. Examples include v7-recyclerview and v7-cardview.
- **Miscellaneous libraries**: The Android support libraries consists of few other libraries such as v8 which provides support for RenderScript, annotations for supporting annotations like @NonNull.

Mention two ways to clear the back stack of Activities when a new Activity is called using intent.

- The first approach is to use a FLAG_ACTIVITY_CLEAR_TOP flag.
```
Intent intent= new Intent(ActivityA.this, ActivityB.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
finish();
```
- The second way is by using FLAG_ACTIVITY_CLEAR_TASK and FLAG_ACTIVITY_NEW_TASK in conjunction.
```
Intent intent= new Intent(ActivityA.this, ActivityB.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(intent);
```

Difference Between FLAG_ACTIVITY_CLEAR_TASK and FLAG_ACTIVITY_CLEAR_TOP?

FLAG_ACTIVITY_CLEAR_TASK is used to clear all the activities from the task including any existing instances of the class invoked. The Activity launched by intent becomes the new root of the otherwise empty task list. This flag has to be used in conjunction with FLAG_ ACTIVITY_NEW_TASK.

FLAG_ACTIVITY_CLEAR_TOP on the other hand, if set and if an old instance of this Activity exists in the task list then barring that all the other activities are removed and that old activity becomes the root of the task list. Else if there's no instance of that activity then a new instance of it is made the root of the task list. Using FLAG_ACTIVITY_NEW_TASK in conjunction is a good practice, though not necessary.

How do you disable onBackPressed()?

The onBackPressed() method is defined as shown below: @Override
public void onBackPressed() {
super.onBackPressed();
}

- To disable the back button and preventing it from destroying the current activity and going back we have to remove the line super.onBackPressed();

Fragment LifeCycle?

getFragmentManager()

- Return the FragmentManager for interacting with fragments associated with this activity.
- **FragmentManager** which is used to create transactions for adding, removing or replacing fragments.

fragmentManager.beginTransaction();

- Start a series of edit operations on the Fragments associated with this FragmentManager.
- The FragmentTransaction object which will be used.
- fragmentTransaction.replace(R.id.fragment_container, mFeedFragment);
- Replaces the current fragment with the mFeedFragment on the layout with the id: R.id.fragment_container

fragmentTransaction.addToBackStack(null);
Add this transaction to the back stack. This means that the transaction will be remembered after it is committed, and will reverse its operation when later popped off the stack.
Useful for the return button usage so the transaction can be rolled back. The parameter name:

Is an optional name for this back stack state, or null.

Types of Fragments?

Basically fragments are divided as three stages as shown below.

- **Single frame fragments** − Single frame fragments are using for hand hold devices like mobiles, here we can show only one fragment as a view.
- **List fragments** − fragments having special list view is called as list fragment
- **Fragments transaction** − Using with fragment transaction. we can move one fragment to another fragment.

Difference between add and replace?

- replace removes the existing fragment and adds a new fragment.
- add retains the existing fragments and adds a new fragment that means existing fragment will be active and they wont be in 'paused' state hence when a back button is pressed onCreateView() is not called for the existing fragment(the fragment which was there before new fragment was added).

why to use replace?

- Replace is the first removal of the same id all the fragments, and then add the current fragment.
- multi-layer is certainly more than a layer of waste, so it is recommended to use replace

What is SQLite and Methods?

- SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 KByte).

Methods - onCreate(SQLiteDatabase db)-

- called only once when database is created for the first time **onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) -**
- called when database needs to be upgraded.

How to Use Cursor?

```
public Contact getContact(int id) {
SQLiteDatabase db = this.getReadableDatabase();
`Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,`<br>
        `KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",`<br>
        `new String[] { String.valueOf(id) }, null, null, null, null);`<br>
`if (cursor != null)`<br>
    `cursor.moveToFirst();`<br>

`Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),`<br>
        `cursor.getString(1), cursor.getString(2));`<br>
`// return contact`<br>
`return contact;`<br>
}
```

- SQLite supports the following data types:
- TEXT(similar to String in Java)
- INTEGER(similar to long in Java)
- REAL (similar to double in Java)

When to use dependancies?

- In Android Studio, dependencies allows us to include external library or local jar files or other library modules in our Android project.

add Dependancies -

What is Proguard?

- Proguard is free Java class file shrinker, optimizer, obfuscator, and preverifier. It detects and removes unused classes, fields, methods, and attributes. It optimizes bytecode and removes unused instructions. It renames the remaining classes, fields, and methods using short meaningless names.

- Shrinking – detects and removes unused classes, fields, methods, and attributes.
- Optimization – analyzes and optimizes the bytecode of the methods.
- Obfuscation – renames the remaining classes, fields, and methods using short meaningless names.
- **Shrink Your Code and Resources**
  detects and removes unused classes, fields, methods, and attributes from your packaged app, including those from included code libraries (making it a valuable tool for working around the 64k reference limit). ProGuard also optimizes the bytecode, removes unused code instructions, and obfuscates the remaining classes, fields, and methods with short names.
- **Shrink Your Code**
- add minifyEnabled true to the appropriate build type in your build.gradle file.
- enable code shrinking on your final APK used for testing, because it might introduce bugs if you do not sufficiently customize which code to keep.

ProGuard outputs the following files:
dump.txt
Describes the internal structure of all the class files in the APK.
mapping.txt
Provides a translation between the original and obfuscated class, method, and field names.
seeds.txt
Lists the classes and members that were not obfuscated.
usage.txt>
Lists the code that was removed from the APK.
These files are saved at /build/outputs/mapping/release/.

Why to Use keep in proguard?

- To fix errors and force ProGuard to keep certain code, add a -keep line in the ProGuard configuration file. For example:

-keep public class MyClass

Security Practices in android?

- IMPLEMENT A GOOD MOBILE ENCRYPTION POLICY
- HAVE A SOLID API SECURITY STRATEGY - Use SSL,Pinned Certificate
- PUT IDENTIFICATION, AUTHENTICATION, AND AUTHORIZATION MEASURES IN PLACE.
- App Data Access Permission
- Password security
- Use Proguard - Obfuscation

What is AIDL?

- AIDL (Android Interface Definition Language) is similar to other IDLs you might have worked with. It allows you to define the programming interface that both the client and service agree upon in order to communicate with each other using interprocess communication (IPC). On Android, one process cannot normally access the memory of another process. So to talk, they need to decompose their objects into primitives that the operating system can understand, and marshall the objects across that boundary for you. The code to do that marshalling is tedious to write, so Android handles it for you with AIDL.

AsyncTask? When an asynchronous task is executed, the task goes through 4 steps: class AsyncTaskExample extends AsyncTask<Void, Integer, String> {

`private  final String TAG = AsyncTaskExample.class.getName();`
`protected void onPreExecute(){`
  `Log.d(TAG, "On preExceute...");`
`}`

`protected String doInBackground(Void...arg0) {`

```
   <br>
   `Log.d(TAG, "On doInBackground...");`<br>

   `for(int i = 0; i<5; i++){`<br>
      `Integer in = new Integer(i);`<br>
      `publishProgress(i);`<br>
   `}`<br>

   `return "You are at PostExecute";}`<br>
<br>
`protected void onProgressUpdate(Integer...a){`<br>
   `Log.d(TAG,"You are in progress update ... " + a[0]);`<br>
`}`<br>
<br>
`protected void onPostExecute(String result) {`<br>
   `Log.d(TAG,result); `<br>
`}`
}
```

- **onPreExecute()**, invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
- **doInBackground(Params...)**, invoked on the background thread immediately after onPreExecute() finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use publishProgress(Progress...) to publish one or more units of progress. These values are published on the UI thread, in the onProgressUpdate(Progress...) step.
- **onProgressUpdate(Progress...)**, invoked on the UI thread after a call to publishProgress(Progress...). The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
- **onPostExecute(Result)**, invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

which material Design you used?

- RecyclerView
- Navigation Drawer
- TextInputLayout
- Snackbar
- Floation Action Button
- SlidingTabLayout

Differennce between Thread and asyncTask?

When you use a Thread, you have to update the result on the main thread using the runOnUiThread() method, while an AsyncTask has the onPostExecute() method which automatically executes on the main thread after doInBackground() returns.