Q 1) What are the different data storage options available on the Android platform?

Answer: Android platform provides a wide range of data storage options. These options must be used based on the need such as data is secure and used with permission only or can be accessed publicly.

Below is the list of data storage options on the Android platform:

- **SharedPreference**: It stores data in XML files. It is the simplest way to store private data in the key-value pair.
- **SQLite**: It stores structured data in the private database.
- **Internal Storage**: It stores data in the device file system and any other app cannot read this data.
- **External Storage**: Data is stored in the file system but it is accessible to all apps in the device

Q2) Explain the AndroidManifest.xml file and why do you need this?

Answer: Every application must have an AndroidManifest.xml file in the root directory. It contains information about your app and provides the same to the Android system.
The information includes the package name, Android components such as Activity, Services, Broadcast Receivers, Content Providers, etc. Every Android system must have this information before running any app code.

AndroidManifest.xml file performs the following tasks:

- It provides a name to the Java package and this name is a unique identifier for the application.
- It describes the various components of the application which include Activity, Services, Content Providers, etc. Also, it defines the classes which implement these components.
- It is responsible to protect the application and it declares the permission for accessing the protected part of the app.
- It also declares the Android API which is going to be used by the application.
- It contains the library file details which are used and linked to the application.

Q 3) What is Orientation?

Answer: Orientation is the key feature in Smartphones nowadays. It has the ability to rotate the screen between Horizontal or Vertical mode.

Android supports two types of screen Orientations as mentioned below:

- **Portrait**: When your device is vertically aligned.
- **Landscape**: When your device is horizontally aligned.

setOrientation() is a method using which you can set a screen alignments. HORIZONTAL and VERTICAL are two values that can be set in the setOrientation() method. Whenever there is a change in the display orientation i.e. from Horizontal to Vertical or vice versa then onCreate() method of the Activity gets fired.

Basically, when the orientation of the Android mobile device gets changed then the current activity gets destroyed and then the same activity is recreated in the new display orientation. Android developers define the orientation in the AndroidManifest.xml file.

Q 4) Which are the dialog boxes supported by the Android platform?

Answer: Android supports four types of dialog boxes:

AlertDialog: It has a maximum of 3 buttons and sometimes AlertDialog includes checkboxes and Radio buttons to select the element.
ProgressDialog: It displays the progress bar or wheels.
TimePickerDialog: Using this dialog box, a user selects the Time.
DatePickerDialog: Using this dialog box, a user selects the Date

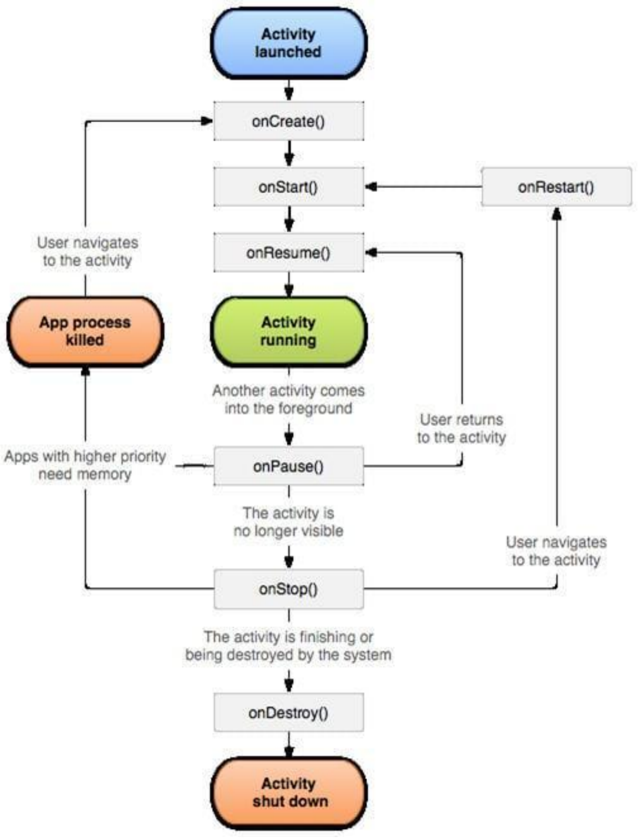
Q 5) What is .apk extension in Android?

Answer: It is a default file format that is used by the Android Operating System. Application Package Kit (APK) is used for the installation of mobile apps. The .apk contains resource file, certificate, manifest file, and other code.
APK files are archive files in the zip format with .apk extension.

Q 6) Explain Activity Lifecycle briefly.

Answer: When a user interacts with the app and moves here and there, out of the app, returns to the app, etc. During all this process "Activity" instances also move in the different stages in their lifecycle.
There are seven different states like – onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop(), and onDestroy(). These are termed as a 'callback'. Android system invokes these callbacks to know that the state has been changed.

The below-given diagram describes the Activity Lifecycle:


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

When a user is working on an app, then there are many activities involved in it like Open, Close, Save, Delete, Send, etc.

Based on the user action these activities are partially disconnected from the UI but these activities always reside in the memory so that when the user calls back the same activity, the user will be in the same state where he has left off.

Q 7) What is meant by Services?

Answer: Service is an Android component that runs in the background and acts independently. It does not provide any user interface.
Though the services are running behind the scene, a user can continue their work on different apps. Most of the time, the users are not aware of the services which are running in the background. These services allow the system to kill the process without interrupting the user's ongoing work.

A service is implemented as a subclass of Service class:

```
Public class MainService extends Service
{
}
```

Q 8) Explain briefly – what is meant by Activities?

Answer: Activities are the part of the mobile app which the user can see and interact with.
For Example, if you open an SMS app which has multiple activities like create new SMS, add a contact from the address book, write the content in the SMS body, send SMS to the selected contact, etc.
Activity keeps a track of the following:

- Keeps track of what a user is currently looking for in an app.
- Keeps a track of previously used processes, so that the user can switch between ongoing process and previous process.
- It helps to kill the processes so that the user can return to their previous state

An activity is implemented as a subclass of Activity class as shown below:

```
Public class MyActivity extends Activity
{
}
```

Q 9) Provide the important core components of Android.

Answer: The core components of Android operating systems are:

- Activity
- Intents
- Services
- Content Provider
- Fragment

Q10) What are the different tools available in Android development? Explain their functions.

Answer: There are a variety of tools available to help Android developers:

- **Android Software Development Kit (SDK) and Virtual Device Manager:** This tool is used to generate and handle Android Virtual Devices (AVD) and SDKs. Through the emulator in the AVD, you can specify the supported SDK version, storage in the SD card, screen resolution, and other abilities such as GPS and touch screen.
- **The Android Emulator:** The AE is the implementation of the Android Virtual Machine, designed to run processes within a virtual device itself, which can be used on a development computer. The main use of this tool is in testing and debugging of Android applications.
- **Android Debug Bridge (ADB):** The ADB is a command-line debugging application doled out with the SDK. It enables developers to communicate with the device, and facilitates actions such as the installation and debugging of an application.
- **Android Asset Packaging Tool (AAPT):** The AAPT builds the '.apk' distributable Android package file.

Q 11): Describe Folder, File & Description of Android Applications

Answer:gen: gen contains the compiler-generated .R file which references all the resources in the project

- **src:** src holds the .java source files in our project
- **bin:** bin contains the .apkk file built by the ADT during the build process, along with all the other things needed to run an Android application
- **AndroidManifest.xml:** This file is the manifest file that explains the basic features of the application and defines all its components
- **res/values:** res/values is a directory for other various XML files that contain resources such as strings, color definitions and more.
- **res/drawable-hdpi:** This is a directory for objects that are drawable and designed for high-density screens
- **res/layout:** It is a directory of files that define the UI for your application

Q 12) What are 'activities'? Describe the lifecycle of an activity.

Answer: Activities in Android are containers/windows to the UI. The lifecycle of an activity is as follows

- **OnCreate():** Here, the views are created and data is collected from bundles.
- **OnStart():** It is called if the activity is visible to the user. It may be succeeded by onResume() if the activity reaches the foreground and onStop() if it converts into hidden.
- **OnResume():** It is called when the activity starts an interaction with the user.
- OnPause(): This is called when the activity is going to the background but hasn't been killed yet.
- **OnStop():** This is called when you are no longer visible to the user.
- **OnDestroy():** Called when the activity is finishing
- **OnRestart():** Called after the activity has been stopped, prior to it being started again

Q 13) State some advantages of Android.

Answer: Below are some advantages of Android:

- **Low investment and better returns:** Android development has a low entry barrier and is suitable for new developers looking to become proficient in the programming field.
- **Free SDK:** One of the most prominent features of Android is that the Software Development Kit is open source and is provided free of charge, which eradicates the cost of licensing, distribution and development fee.
- **Easy Adoption:** Android applications are scripted in Java, which is one of the most used programming languages in the world.
- **Reusable:** Android components can be reused and even replaced by the framework.
- **Multi-Platform Support:** The Android platform supports major OSs such as Linux, Mac OS, and Windows.
- **Support for Wearable Devices:** The market is now flooded with wearable devices and Android has emerged as leading support for such devices that are now readily available in the market.

Q 14) What is Android Runtime?

Answer: Android Runtime (ART) is an application used by the Android OS as a runtime environment. It has now replaced Dalvik, a discontinued process Virtual Machine (VM). ART translates the bytecode of the application into native instructions, which are carried out by the device's runtime environment.

Question: Explain the dialog boxes supported on Android.

Answer: Android supports four dialog boxes

- **AlertDialog:** The most suggested dialog box, the AlertDialog supports 0-3 buttons, along with a list of selectable items, such as radio buttons and checkboxes.
- **DatePickerDialog:** Used for the selection of date by the user
- **TimePickerDIalog:** Used for the selection of time by the user
- **ProgressDialog:** Used to display a progress bar and is an extension of the AlertDIalog. It also supports the addition of buttons.

Question: What are some of the disadvantages of Android?

Answer: Below are some disadvantages of the Android Operating System

- **Fake Applications:** There are thousands of fake apps available on the market at any given time, which when installed may try to steal your data.
- **Streamlining issues:** There are a variety of Android devices available in the market, with different screen sizes and dimensions, but more importantly, different Android Operating Systems. Every application developer has to constantly work towards updating their application for the new OS but with various OS version and upgrades, the process is quite difficult. An application which runs smoothly on one version of the Android OS might crash on a different Android OS.
- **Background Processes:** The abundance of running processes in the background is always an issue, as they eat up the battery quickly.

Q 15) What is Context?

Answer: The Context in Android, as its name suggests, in the context of the current state of your application or object. The context comes with services such as giving access to databases and preferences, resolving resources and much more. There are two types of context:

- **Activity Context:** This context is attached to the lifecycle of an activity. It should be used when you are passing the context in the scope of an activity or you need the context whose lifecycle is attached to the current context.
- **Application Context:** This context is attached to the lifecycle of an application. The application context can be used where you need a context whose lifecycle is separate from the current context or when you are passing a context beyond the scope of activity.

Q 16) Is there any difference between activities and services?

Answer: Yes, there are a lot of differences between activities and services. These differences are stated as under:

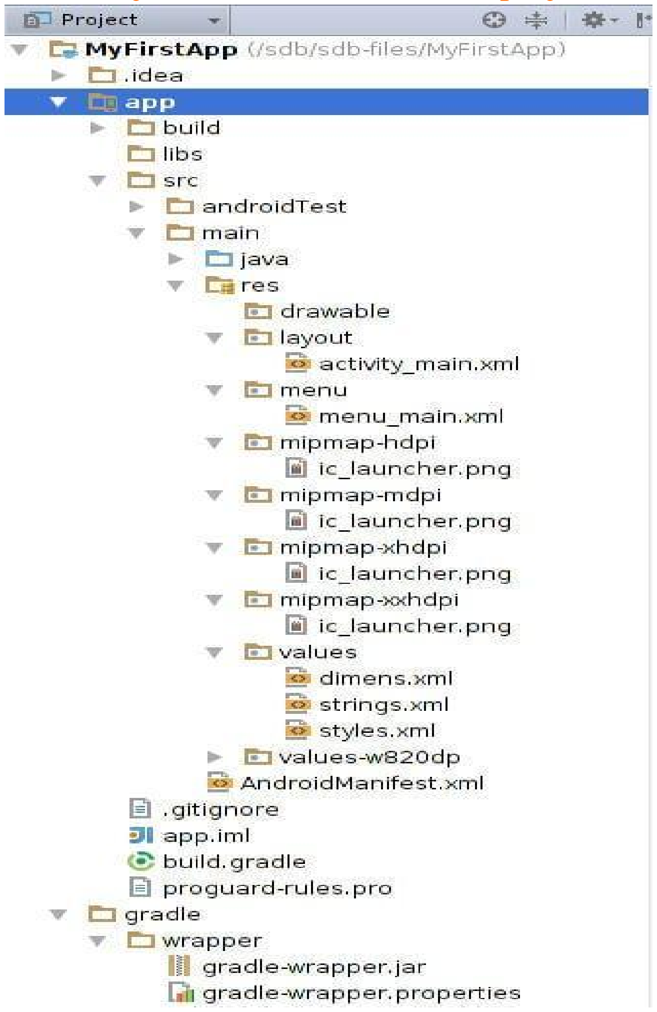
Activities	**Services**
These are closed	These are open
These can be terminated at any time	These cannot be terminated at any time
They are designed to run before the scenes	These are designed to run behind the scenes
They are dependent	They act independently
They are not continuous	They are not continuous

Q 17) Which components are necessary for a New Android project?

Ans: Whenever a new Android project is created, the below components are required:

- **manifest:** It contains an **XML** file.
- **build/:** It contains build output.
- **src/:** It contains the code and resource files.
- **res/:** It contains bitmap images, UI Strings and XML Layout i.e. all non-code resources.
- **assets/:** It contains a file that should be compiled into a **.apk** file.

The below image shows the Project View once an Android project is created:



```
MyFirstApp (/sdb/sdb-files/MyFirstApp)
  .idea
  app
    build
    libs
    src
      androidTest
      main
        java
        res
          drawable
          layout
            activity_main.xml
          menu
            menu_main.xml
          mipmap-hdpi
            ic_launcher.png
          mipmap-mdpi
            ic_launcher.png
          mipmap-xhdpi
            ic_launcher.png
          mipmap-xxhdpi
            ic_launcher.png
          values
            dimens.xml
            strings.xml
            styles.xml
          values-w820dp
        AndroidManifest.xml
    .gitignore
    app.iml
    build.gradle
    proguard-rules.pro
  gradle
    wrapper
      gradle-wrapper.jar
      gradle-wrapper.properties
```

Q 18) Define and explain the Android Framework.

Answer: Android framework is a set of API's using which the Android developers write code for the mobile apps. It contains the methods and classes to write the programming code.

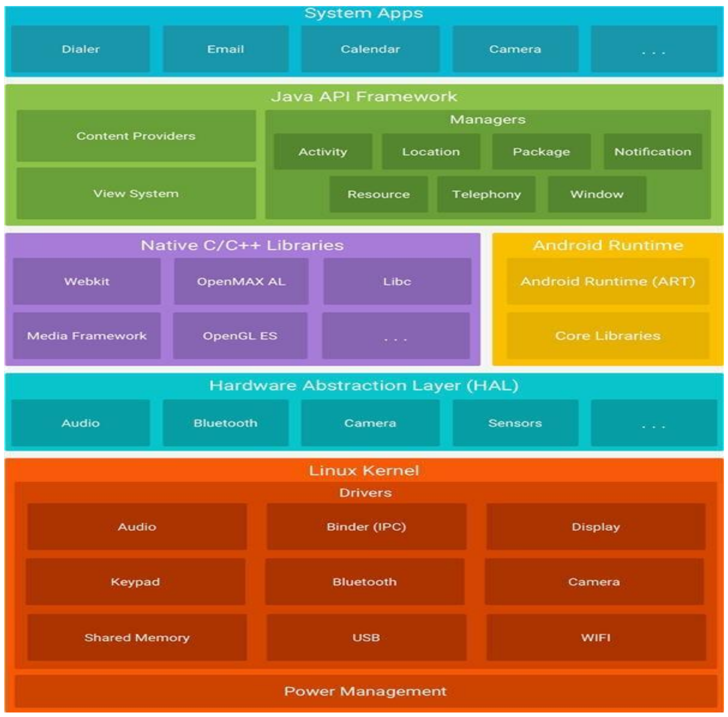
Android framework includes a different set of tools to create image pane, text field, buttons, etc. It also includes "Activities" with which the user interacts and "Services", which are the programs that run in the background. It is a package of different components like Intents, Broadcast Receivers, Content Providers, etc.

Q19) Explain Android Architecture briefly.

Answer: Android architecture is in the form of software stack components.

The below diagram describes the different layers in the Android architecture.

- **Linux Kernel**: Linux Kernel is placed at the bottom of the software stack and is the foundation of the Android architecture. Using Linux kernel, Android provides a connection between the other layers of the software. It helps to develop drivers like the keypad, display, audio for device manufacture, etc.
- **Hardware Abstraction Layer (HAL)**: HAL provides an interface between device drivers and API framework. It consists of library modules that are specific to the hardware component.
- **Android Runtime**: Linux kernel provides a multi-tasking execution environment so that multiple processes can execute each process runs on its own instance of Android Runtime (ART). Android has core runtime libraries like Dalvik VM specific libraries, Java Interoperability Libraries, Android Libraries, and C/C++ libraries.
- **Application Framework (Java API Framework)**: The entire android functionalities are available through the API. It consists of multiple services like Activity Manager, Resource Manager, Notification Manager, etc., which form the environment in which the android application runs.
- **Applications**: The Android application is a top layer and all types of in-built applications such as SMS, Browsers, Contact, etc are included in this top layer. It also includes third-party applications that are installed by the user such as Games, etc.



20) What are the features of Android?

Google has changed the lives of everyone by launching a product that improves the mobile experience for everyone. Android helps in understanding your tastes and needs, by giving various features such as having wallpapers, themes, and launchers which completely change the look of your device's interface.

Android has plenty of features. Some of the features are listed below:

- Open-source
- Customizable operating System
- Variety of apps can be developed.
- Reduces overall complexity
- Supports messaging services, web browser, storage(SQLite), connectivity,media and many more.

Q21). What is the difference between an implicit intent and explicit intent?

Implicit Intent is used whenever you are performing an action. For example, send email, SMS, dial number or you can use a Uri to specify the data type.

```
 For example: Intent i = new Intent(ACTION_VIEW,Uri.parse("<a
href="http://www.google.com">

http:// www.google.com </a>"));

startActivity(i);
```

Explicit, on the other hand, helps you to switch from one activity to another activity(often known as the target activity). It is also used to pass data using putExtra method and retrieved by other activity by getIntent().getExtras() methods.

For example:

```
Intent i = new Intent(this, Activitytwo.class);

 #ActivityTwo is the target component

i.putExtra("Value1","This is ActivityTwo");

i.putExtra("Value2","This Value two for ActivityTwo");
```

```
startactivity(i);
```

Q 22) What are broadcast receivers? How is it implemented?

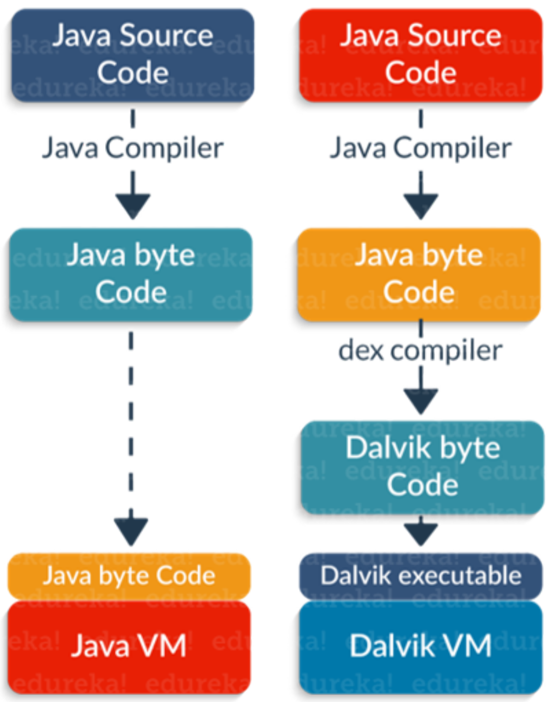
- Broadcast Receiver is a mechanism using which host application can listen for System level events.
- Broadcast receiver is used by the application whenever they need to perform the execution based on system events. Like listening for Incoming call, sms etc.
- Broadcast receivers helps in responding to broadcast messages from other application or from the system.
- It is used to handle communication between Android operating system and applications.

It is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver extends BroadcastReceiver {

Public void onReceive(context,intent){

}

}
```

Q 23) What is an Android Runtime?

- Android Runtime consists of Dalvik Virtual machine and Core Java libraries.
- DVM is optimized for low processing power and low memory environments.
- Unlike JVM, the Dalvik Virtual Machine doesn’t run .class files, instead it runs .dex files.
- Android 2.2 “Froyo“ brought trace-based just-in-time (**JIT**) compilation into Dalvik, optimizing the execution of applications and dynamically compiling frequently executed short segments of their bytecode into native machine code.



Q 24) What are the core building blocks of android?

- **Activity** : An Activity is the screen representation of any application in Android. Each activity has a layout file where you can place your UI.
- **Content Provider:** Content providers share data between applications.
- **Service**: It is a component that runs in the background to perform long-running operations without interacting with the user and it even works if application is destroyed.
- **Broadcast Receivers**: It respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents.

Q 25) What is a content provider? How is it implemented?

- Content providers manage access to a structured set of data. It is the standard interface that connects data in one process with code running in another process.
- They encapsulate the data and provide mechanisms for defining data security.
- Content providers are used to share the data between different applications.

It is implemented as a subclass of **ContentProvider**class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentprovider extends ContentProvider {
```

```
public void onCreate(){}

}
```

Q26) what are the measurements used in android ?

- **px (Pixels)** - Actual pixels or dots on the screen.
- **in (Inches)** - Physical size of the screen in inches.
- **mm (Millimeters)** - Physical size of the screen in millimeters.
- **pt (Points)** - 1/72 of an inch.
- **dp (Density-independent Pixels)** - An abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. "dip" and "dp" are same.
- **sp (Scale-independent Pixels)** - Similar to dp unit, but also scaled by the user's font size preference.

Q27) Explain the types of layout ?

- Linear **Layout**. A **layout** that organizes its children into a single horizontal or vertical row. ...
- Relative **Layout**. Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).
- Web View. ...
- List View. ...
- Grid View.

Q28) What are the user interface in android components ?

- TextView
- EditText
- AutoCompleteTextView
- Button
- ImageButton
- ToggleButton
- CheckBox
- RadioButton
- RadioGroup
- ProgressBar

- Spinner
- TimePicker
- DatePicker
- SeekBar
- AlertDialog
- Switch
- RatingBar

Q29) Discuss event handling functions in android ?

onClick()	OnClickListener() This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	OnLongClickListener() This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	OnFocusChangeListener() This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.

onKey()	OnFocusChangeListener() This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.
onTouch()	OnTouchListener() This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	OnMenuItemClickListener() This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.
onCreateContextMenu()	onCreateContextMenuItemListener() This is called when the context menu is being built(as the result of a sustained "long click)

Q30) What is fragment and how to create fragments ?

- **Fragments** are incomplete sentences. Usually, **fragments** are pieces of sentences that have become disconnected from the main clause. One of the easiest ways to correct them is to remove the period between the **fragment** and the main clause. Other kinds of punctuation may be needed for the newly combined sentence.

- Fragments require a dependency on the AndroidX Fragment library. You need to add the Google Maven repository to your project's `build.gradle` file in order to include this dependency.
- To create a fragment, extend the AndroidX `Fragment` class, and override its methods to insert your app logic, similar to the way you would create an `Activity` class. To create a minimal fragment that defines its own layout, provide your fragment's layout resource to the base constructor

Q31) Explain fragment life cycle ?

- Each `Fragment` instance has its own lifecycle. When a user navigates and interacts with your app, your fragments transition through various states in their lifecycle as they are added, removed, and enter or exit the screen.

To manage lifecycle, `Fragment` implements `LifecycleOwner`, exposing a `Lifecycle` object that you can access through the `getLifecycle()` method.

Each possible `Lifecycle` state is represented in the `Lifecycle.State` enum.

- `INITIALIZED`
- `CREATED`
- `STARTED`
- `RESUMED`
- `DESTROYED`

Q32) Describe fragment transactions ?

- As said before a `FragmentTransaction` gives us methods to add, replace, or remove fragments in Android. It gives us an interface for interacting with fragments.

```
fragmentTransaction.replace(R.id.fragment_container, mFeedFragment);
```

- The method `replace(int containerViewId, Fragment fragment)` replaces an existing `Fragment` object from the container `containerViewId` and adds the the `Fragment fragment`

```
fragmentTransaction.addToBackStack(null);
```

- This method, `addToBackOfStack(String name)`, adds this transaction to the back stack, this can be used so that `Fragments` are remembered and can be used again by the `Activity`

```
fragmentTransaction.commit();
```

Q33)What is multiscreen activities ?

Turn on the **split-screen** mode on your **Android** smartphone. There are times when you want to run two (or more) **applications** at the same time on your smartphone. Instead of jumping back and forth between apps, **Android** has a built-in **split-screen** mode that lets you view two apps side-by-side on your **screen**.

Q34) How to passing the data activities using intents ?

- **Intent intent** = new **Intent**(getApplicationContext(), SecondActivity. class); **intent**. putExtra("**Variable** name", "**Value** you want to **pass**"); startActivity(**intent**);
- Now on the OnCreate method of your SecondActivity you can fetch the extras like this.

Q35) What are the native actions ?

- Native Android applications also use Intents to launch Activities and sub-Activities.
- The following non-comprehensive list shows some of the native actions available as static string constants in the Intent class. When creating implicit Intents you

can use these actions, called Activity Intents, to start Activities and sub-Activities within your own applications.

- Later you will be introduced to Intent Filters and you'll learn how to register your own Activities as handlers for these actions.

Q36) What are the uses of intent filters ?

- An intent filter is an instance of the IntentFilter class. Intent filters are helpful while using implicit intents, It is not going to handle in java code, we have to set it up in AndroidManifest.xml. Android must know what kind of intent it is launching so intent filters give the information to android about intent and actions.
- Before launching intent, android going to do action test, category test and data test. This example demonstrate about how to use intent filters in android.

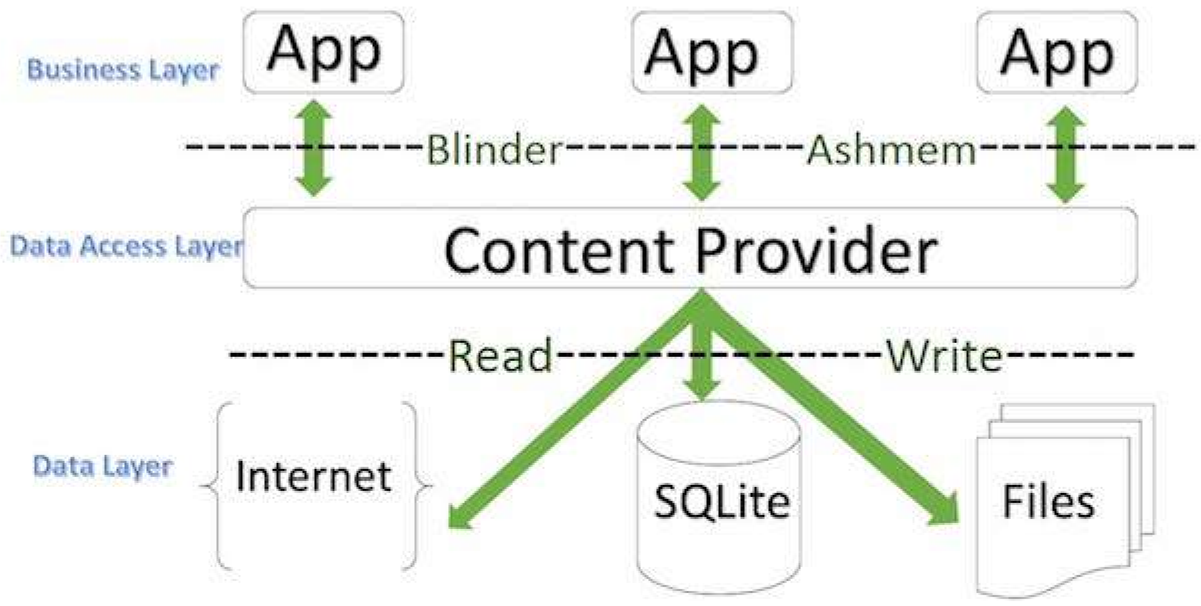
Q37) How can read the data from files ?

- **The basic steps in reading data from a file are:**

1. Tell the program where to find the **data**.
2. Open a path to the **data**.
3. Set up the program variables to access the **data**.
4. **Read** the **data**.
5. Close the **data** path.

Q38) Describe content providers with architecture ?

- A **content provider** manages access to a central repository of data. A **provider** is part of an **Android** application, which often provides its own UI for working with the data. However, **content providers** are primarily intended to be used by other applications, which access the **provider** using a **provider** client object.



●

Q39) How to creating and opening the database ?

- onCreate(): called when the **database** is created for the first time.
- onUpgrade(): called in the event that the application code contains a more recent **database** version number reference.
- onOpen(): called when the **database** is **opened**.
- getWritableDatabase(): opens or creates a **database** for reading and writing.

Q40) how to the create alarms ?

- **I have included the source code below.**

1. Open the **Android** Studio and start a new project.
2. Put the application name and company domain. ...
3. Select the **Android** minimum SDK. ...
4. Choose the basic activity, then click Next.
5. Put the activity name and layout name. ...
6. Go to activity_main. ...
7. **Create** a new content_main. ...
8. Into the MainActivity.