**ANDROID INTERVIEW**

* Android apps are written In Java,kotling,c++.
* The Android SDK contains tools to compile source code and data to application package (APK).
* Every Android app lives in its own security sandbox.
* The Android OS is a multi-user linux system in which each app is a different user.
* By default, the system assigns unique linux user id id for each app, and sets permissions to the files so that the userid assigned to that app can only access them.
* Each app runs in its own virtual machine.
* Bydefault the system runs in its own linux process, the system starts the process when one of its application components needs to be executed and shuts down process when it’s no longer needed.

4 Types of App components:

1. Activities
2. Services
3. Broad cast receivers
4. Content Providers

**Activities:**

* An Activity represents a single screen with user interface.
* Every screen that is visible to the user is an activity.
* An Email contains an activity that shows list of emails, and another activity that allows to compose mail and so on.
* Implement an activity as a sub class of **Activity.**

**Service:**

* Service is a component that runs in the background performing long running operations.
* A service does not display a user interface.
* Example;
* A service that fetches data from the server in background without disturbing user’s interaction with activity.
* Implement service as subclass of **Service.**

**Broadcast Receiver:**

* Broadcast Receiver enables system to deliver events to the app, outside of regular user flow thus allowing the app to respond to system wide broadcast announcements.
* The system can deliver broad cast events to the apps that are not running.
* No user interfaces
* Example :
* Battery low indication .
* Listens for service that fetches data from network in the background and alert the user with notification when data fetching is done.
* Implemented as subclass of **BroadcastReceiver**:

**Content Provider:**

Content Provider manages shared set of data stored in SQLite DB, Web server,files or any persisitent storage. Other apps can query,modify the data present in these persistent storage only if content providers permits them.

Example:

Android system provides content provider to manage contact information.

Other apps can add ,delete or modify a contact only If they have proper permissions so that they can query content provider.

ContactsContract.data is the content provider

Implemented as subclass of ContentProvider.

**Activities:**

**Activity Declaration in Manifest :**

* Declared with tag as “activity “ and attribute as “android:name”
* <activity android:name = “HomeActivity” />

Intent Filter:

* Specifies the type of intent to which your activity can respond or launch.
* Intent Filter is an power feature of Android platform that provides ability to launch the activity not only based on “Explicit request ” but also due to “Implicit” request.
* For Example you may specify to launch an list of emails in an Gmail App by Explicit Intent.
* Or else you can specify to launch the email Screen of any app by specifying “implicit Intent” which shows an user interface that displays list of activities to perform that task .

**<activity**

**<intent-filter**

**<action android:name = “android.Intent.action.SEND” />**

**<category android:name = “android.intent.category.DEFAULT”/>**

**<data android:mimetype = “Text/Plain”**

**/>**

**\*\*\*Category allows your activity to receive launch requests.**

If you don’t want your activities to be launched by other apps ,there should n’t be intent filters declared for such activities. In that case the activities should be launched explicitly using StartActivity ().

Intent myIntent = new Intent();

myIntent.setAction(Intent.ACTION\_SEND)

myIntent.setType(“text/plain”)

myIntent.putExtra(“Input\_DATA”,”myData”);

startACtivity(myIntent)

**Activity Life Cycle**

**OnCreate() :** This call should be implemented. fired when the system creates activity.This is where initialization of essential components required for activity should be made.This is where app should create views and bind data to the list. This is where we define layout to the activity by calling setContentView().

**OnStart() :** Called when onCreate() is completed. This method is called just before activity is about to launch . Activity is visible but no user interaction. Final preparations to launch activity is done in this method.

**onResume() :** Once onStart() is done , onResume() is called ,this is where activity is on foreground and user can interact with the activity .Core functionality of activity is implemented in onResume()

**onPause() :** onPause() is always followed by onResume(). This is called when the activity loses its focus becoz other activity is launched on top of it . Activity is still partially visible with no user interaction. Called when user presses back button, or a phone call screen appears when mobile receives a phone call.

**onStop() :** Activity is stopped and in background and is not visible to the user. Called when the user presses back button or an activity is destroyed, other activity which is in paused state is resumed , a new activity is launched from the existing activity.

**onRestart() :** called when the activity in stopped state is about to restart. Followed by onStart() and onResume()

**onDestroy() :** Called when the activity is destroyed as user presses the back button or if user called finish() method.

**onSavedInstanceState():** The system saves the state of activity like the values of variables that keep track of user’s state in that activity. Save simple,lightweight UI like primitive data types.

**onRestroreInstanceState():** Called when the system restores your activity. The state information is received using bundle object.

Starting an Activity:

1. startActivity()
2. StartActivityForResult(): If activity wants results back from new activity.

**LAUNCHMODES**

1. **Standard : The Sys. Creates new instance of activity in this task to which it belongs and redirects intent to that task. There can be multiple instances of activity in the task and there can be multiple tasks with this activity**
2. **SingleTop : If there exists an instance of activity @ the top of the stack, the system redirects the intent of activity to that task by onNewIntent() method rather than creating new instance of an activity.If activity does not exists at the top of the stack or an activity instance did not exist in the task then a new instance of activity is created . similar to “Standard” mode**
3. **SingleTask : The creates new task and keeps the activity the top , it redirects the intent to the other task if an instance of an activity exists in that task rather than creating new instance.**
4. **SingleInstance : The activity is the only member of the task . All the subsequent activities started by this activity will be launched in separate task.**

**Intent Flags:**

**FLAG\_ACTIVITY\_NEW\_TASK :**  Behaves like “Standard” mode

**FLAG\_ACTIVITY\_SINGLE\_TOP : Behaves like “singleTop” mode**

**FLAG\_ACTIVITY\_CLEAR\_TOP :** If an activity being started is already running in the task, then all other activities on the top of it is cleared .

**Clearing the Back Stack:**

**alwaysRetainTaskState** : I f this attribute is set to “true” for the root activity of a task, all the activities in the task are preserved when the user returns to the task after leaving it for long time.

**clearTaskOnLaunch** - If this attribute is set to “true” for the root activity all the activities in the top are cleared except the root activity when the user leaves the task for a moment also

**finishOnTaskLaunch**: This attribute is set to true, it effects only on the activity to which this attribute is set not the entire task. The activity will be cleared from the task when the user leaves the task for some time and comes back.

**Parcelable**

Parcelable is the Android only interface to serialize a class , so that its properties can be transferred from one activity to other.

Implementation:

* Create a class that implements Parcelable
* Create Constructor with Parcel as argument to read data to parcel:

Public Info(Parcel p){

Name = p.readString();

}

* Over a method

**Public void writeToParcel(Parcel p, int flags){**

**p.writeToString(name)**

**}**

* Override

**Public int describeContents(){**

**Return hashCode();**

**}**

* **Public static final Parcelable.Creator<Info> CREATOR = new Parcelable.Creator<Info>(){**
* public User createFromParcel(Parcel in) {
* return new Info(in);
* }
* public Info[] newArray(int size) {
* return new Info[size];
* }

**}**