**Activity**

In android, **Activity** represents a single screen with a user interface (UI) of an application and it will acts an entry point for users to interact with an app.

* Activity is an Application component.
* Activities are loosely tied (minimal dependency between the activities) together to make up an app.
* it can be organised in parent-child relationship in the Android Manifest.
* by using activity, UI components or widgets can be palced in a single screen.
* Java file and xml file makes together Activity complete.
* An activity is the single screen in android. It is like window or frame of Java.

To use activities in our application we need to define an activities with required attributes in manifest file (**AndroidMainfest.xml**)

**android:name -** represent the name of class

<application>

<activity  
 android:name=".MainActivity"  
 android:exported="true">  
<intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category

android:name="android.intent.category.LAUNCHER"/>  
</intent-filter>  
 <meta-data  
 android:name="android.app.lib\_name"  
 android:value=""/>

</activity>

</application>

* In android application, activities can be implemented as a subclass of **Activity** class

public class MainActivity extends Activity {

}

* The onCreate() and onDestroy() methods are called only once throughout the activity lifecycle.

(methods of android.app.Activity class.)

* **Activity** class have 7 callback methods to describe how the activity will behave at different stages.

onCreate(), onStart(), onPause(), onRestart(), onResume(), onStop() and onDestroy()

1. **onCreate()**

This is the first callback method and it fires when the system creates an activity for the first time. During the activity creation, activity entered into a **Created** state.

If we have an application start-up logic that needs to perform only once during the life cycle of an activity, then we can write that logic in onCreate() method.

@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity\_main);  
}

1. **onStart()**

The onStart() method will make an activity visible to the user and this method execution will finish very quickly.

e.g. protected void onStart(){  
    super.onStart()  
 }

1. **onResume()**
   1. In this state activity start interacting with the user that means user can see the functionality and designing part of an application on the single screen.
   2. The app will stay in this **Resumed** state until an another activity happens , like getting a phone call or screen turned off, etc.
   3. In case if any interruption events happen in **Resumed** state, the activity will enter into **Paused** state .
   4. After an activity returned from **Paused** state to **Resumed** state, the system again will call onResume() method
   5. due to this we need to implement onResume() method to initialize the components that we release during onPause() method

e.g. public void onResume() {  
     super.onResume();   
     if (mCamera == null) {  
        initializeCamera();  
    }  
 }

1. **onPause()**
   1. Whenever the user leaves an activity or the current activity is being Paused then the system invokes onPause() method.
   2. It is used to pause operations like stop playing the music, end of Call

e.g. public void onPause() {  
     super.onPause();  
   if (mCamera != null) {  
        mCamera.release();  
        mCamera = null;  
    }  
 }

the next method is either onStop() or onResume() depending on what happens after an activity entered into a **Paused** state.

1. **onStop()**
   1. The system will invoke onStop() callback method when an activity no longer visible to the user, the activity will enter into Stopped state.
   2. This happens due to current activity entered into **Resumed** state or newly launched activity covers complete screen or it’s been destroyed.
   3. The onStop() method is useful to release all the app resources which are no longer needed to the user.

e.g. protected void onStop(){  
     super.onStop();  
 }

1. **onRestart()**
   1. The system will invoke onRestart() method when an activity restarting itself after stopping it. The onRestart() method will restore the state of activity from the time that is being stopped.
   2. The onRestart() callback method in android activity will always be followed by onStart() method.
2. **onDestroy()**
3. The system will invoke onDestroy() method before an activity is destroyed and this is the final callback method received by the android activity.
4. Method is called either the activity is finishing or system destroying the activity to save space.

e.g. public void onDestroy(){

super.onDestroy();

}

