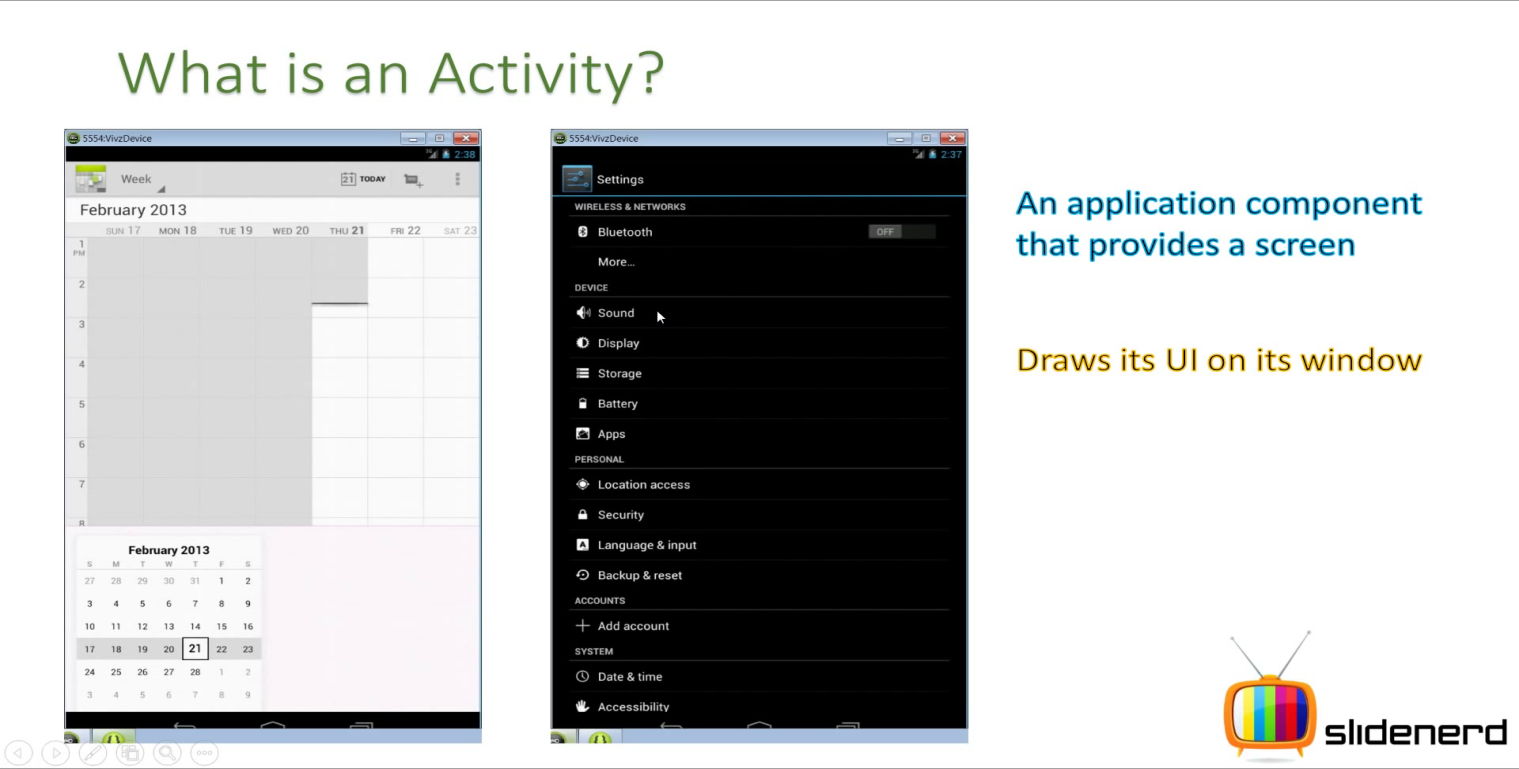
**Activity :-**

It is class defines in an android.app package an object of this class provides a screen contains different-different UI with which user can interact.

An application can have many screens i. e. many activities, and one of them must be marked as launcher activity in AndroidManifest.xml file, that is presented on screen when the apps lunches.

**Note:-** An Application can have multiple activities and all those activities must be defined inside AndroidManifest.xml file by using <activity> under <application>.



**How to make an activity as launcher activity?**

When the user selects your app icon from the Home screen, the system calls the [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) method for the [Activity](http://developer.android.com/reference/android/app/Activity.html) in your app that you've declared to be the "launcher" (or "main") activity. This is the activity that serves as the main entry point to your app's user interface.

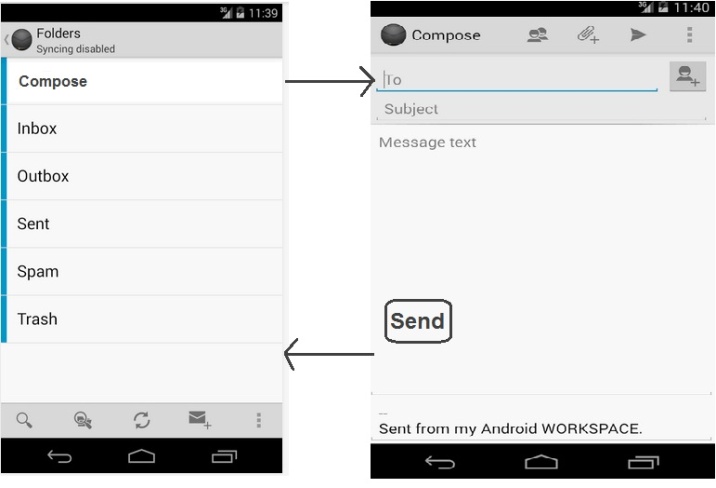
You can define which activity to use as the main activity in the Android manifest file, [AndroidManifest.xml](http://developer.android.com/guide/topics/manifest/manifest-intro.html), which is at the root of your project directory.

The main activity for your app must be declared in the manifest with an [<intent-filter>](http://developer.android.com/guide/topics/manifest/intent-filter-element.html) that includes the [MAIN](http://developer.android.com/reference/android/content/Intent.html#ACTION_MAIN) action and [LAUNCHER](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_LAUNCHER) category. For example:

<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>

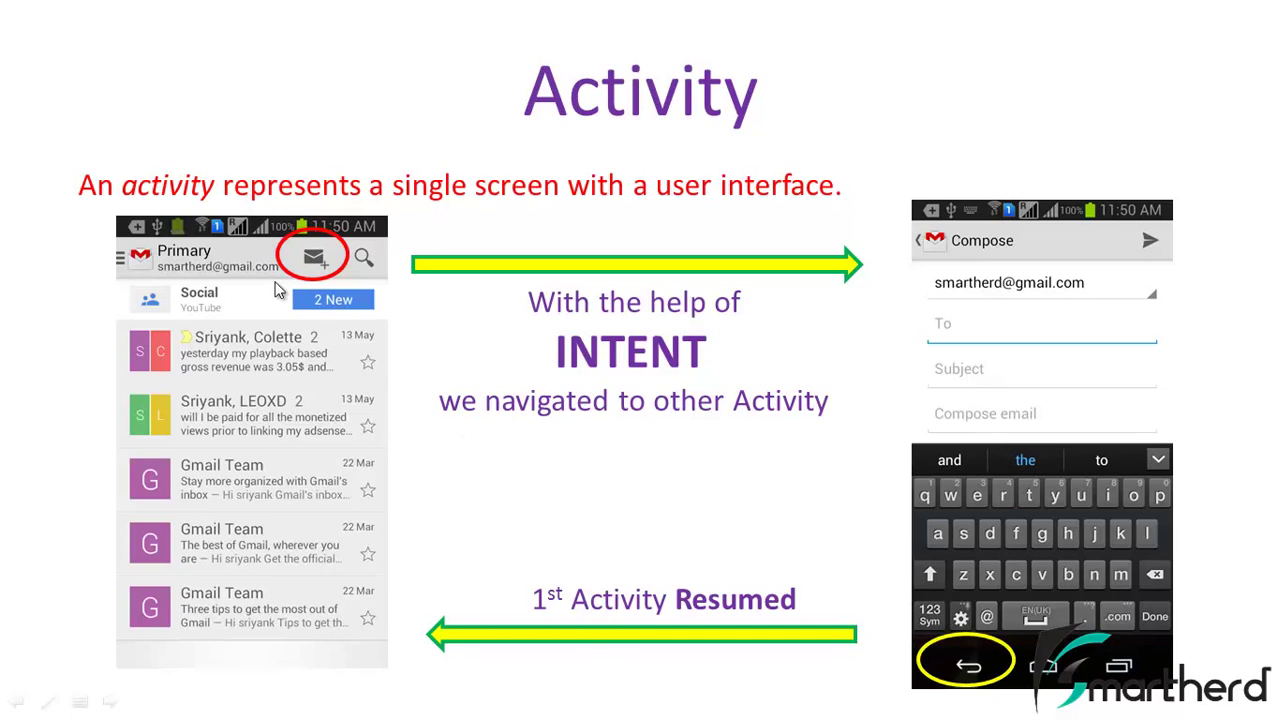
**Note:** When you create a new Android project with the Android SDK tools, the default project files include an [Activity](http://developer.android.com/reference/android/app/Activity.html) class that's declared in the manifest with this filter.

If either the [MAIN](http://developer.android.com/reference/android/content/Intent.html#ACTION_MAIN) action or [LAUNCHER](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_LAUNCHER) category are not declared for one of your activities, then your app icon will not appear in the Home screen's list of apps.

Example- An Email app can have one activity that shows a list of new Emails, another activity for reading Emails, another activity to Compose Email.

**Note:-**

1. An application can start any activity belonging to itself or of other application.
2. When a new activity starts the previous activity is stopped and added to a stack memory known as Back Stackfor reuse by android OS to improve efficiency (when user comes back to previous screen).



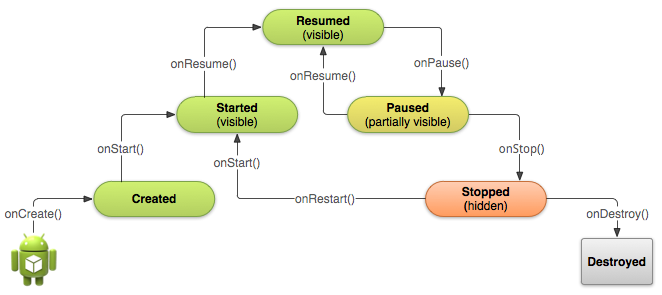
Note:-Activity Creation is a costly process in terms of memory and processing time.

## Understand the Lifecycle Callbacks -

Android OS calls certain methods on the object of activity class to notify weather your apps is currently running or not, or is being stopped, or is being Paused etc.

During the life of an activity, the system calls a core set of lifecycle methods in a sequence similar to a step pyramid. That is, each stage of the activity lifecycle is a separate step on the pyramid. As the system creates a new activity instance, each callback method moves the activity state one step toward the top. The top of the pyramid is the point at which the activity is running in the foreground and the user can interact with it.

As the user begins to leave the activity, the system calls other methods that move the activity state back down the pyramid in order to dismantle the activity. In some cases, the activity will move only part way down the pyramid and wait (such as when the user switches to another app), from which point the activity can move back to the top (if the user returns to the activity) and resume where the user left off.



**Figure 1.** A simplified illustration of the Activity lifecycle, expressed as a step pyramid. This shows how, for every callback used to take the activity a step toward the Resumed state at the top, there's a callback method that takes the activity a step down. The activity can also return to the resumed state from the Paused and Stopped state.

Depending on the complexity of your activity, you probably don't need to implement all the lifecycle methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect. Implementing your activity lifecycle methods properly ensures your app behaves well in several ways, including that it:

* Does not crash if the user receives a phone call or switches to another app while using your app.
* Does not consume valuable system resources when the user is not actively using it.
* Does not lose the user's progress if they leave your app and return to it at a later time.
* Does not crash or lose the user's progress when the screen rotates between landscape and portrait orientation.

As you'll learn in the following lessons, there are several situations in which an activity transitions between different states that are illustrated in figure 1. However, only three of these states can be static. That is, the activity can exist in one of only three states for an extended period of time:

**Resumed**

In this state, the activity is in the foreground and the user can interact with it. (Also sometimes referred to as the "running" state.)

**Paused**

In this state, the activity is partially obscured by another activity—the other activity that's in the foreground is semi-transparent or doesn't cover the entire screen. The paused activity does not receive user input and cannot execute any code.

**Stopped**

In this state, the activity is completely hidden and not visible to the user; it is considered to be in the background. While stopped, the activity instance and all its state information such as member variables is retained, but it cannot execute any code.

The other states (**Created and Started**) are transient and the system quickly moves from them to the next state by calling the next lifecycle callback method. That is, after the system calls [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)), it quickly calls [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart()), which is quickly followed by [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume()).

The Activity class defines following call backs but we don’t need to override all callbacks in our Activity class(Subclasss). However it’s important to know all callbacks that ensure our application behaves the way user expect.

1. **onCreate():-**

This is first callback which is called by android OS when the activity has been first created.

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

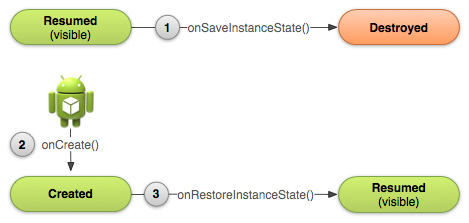
**Explation:-**

**Bundle:-** It is class defined in android.os package and object of this class is represented a buffer which is used by android OS to save or recover state information of our activity.

**savedInstanceState:-** It is an object of Bundle class, at first time creation of activity it contains NULL that means fresh launch of activity but when activity is killed by OS OR manually killed by user OR Orientation has been changed kind of things then OS will store state information of activity in this object of Bundle class.

***Note:****-*  Android OS will use this savedInstanceState information of activity while the activity is recreated.

To save **additional data** about the activity state, you must override the [onSaveInstanceState()](http://developer.android.com/reference/android/app/Activity.html#onSaveInstanceState(android.os.Bundle)) callback method. The system calls this method when the user is leaving your activity and passes it the [Bundle](http://developer.android.com/reference/android/os/Bundle.html) object that will be saved in the event that your activity is destroyed unexpectedly. If the system must recreate the activity instance later, it passes the same [Bundle](http://developer.android.com/reference/android/os/Bundle.html) object to both the [onRestoreInstanceState()](http://developer.android.com/reference/android/app/Activity.html#onRestoreInstanceState(android.os.Bundle)) and [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) methods.



**Note-**

[onSaveInstanceState()](http://developer.android.com/reference/android/app/Activity.html#onSaveInstanceState(android.os.Bundle)) callback method is called by OS before onStop() and

[onRestoreInstanceState()](http://developer.android.com/reference/android/app/Activity.html#onRestoreInstanceState(android.os.Bundle)) callback method is called before onResume().

**super.onCreate(savedInstanceState):-** By calling this statement we are trying to tell AVM to run our code in addition to that existing code in the onCreate() method of parent Activity class because onCreate() method of activity class having all the logic to create an activity (screen).

***Note:-*** If we leave out this statement then only our code will run means predefined code will be ignored completely.

**setContentView(R.layout.activity\_main):-** This is a method of an Activity class that receive a unique ID of xml Layout resource and inflate that Layout resource in activity object. In other words it sets the Activity content from layout resource in the object of Activity class.

**How to get any sub contents (like- widget object) from Activity object?**

By using following method of Activity class

*View findViewById(int id);*

This method takes an unique ID of xml resource that is inflated in Activity object and returns that xml resource in the form of View object.

\*\*So in all overriding callback methods we have to call same method of parent Activity class. Otherwise your app will be crashed.

You must implement the [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) method to perform basic application startup logic that should happen only once for the entire life of the activity. For example, your implementation of [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) should define the user interface and possibly instantiate some class-scope variables.

For example, the following example of the [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) method shows some code that performs some fundamental setup for the activity, such as declaring the user interface (defined in an XML layout file), defining member variables, and configuring some of the UI.

TextView mTextView; // Member variable for text view in the layout  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    // Set the user interface layout for this Activity  
    // The layout file is defined in the project res/layout/main\_activity.xml file  
    setContentView(R.layout.main\_activity);  
      
    // Initialize member TextView so we can manipulate it later  
    mTextView = (TextView) findViewById(R.id.text\_message);  
      
    // Make sure we're running on Honeycomb or higher to use ActionBar APIs  
    if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB) {  
        // For the main activity, make sure the app icon in the action bar  
        // does not behave as a button  
        ActionBar actionBar = getActionBar();  
        actionBar.setHomeButtonEnabled(false);  
    }  
}

**Caution:** Using the [SDK\_INT](http://developer.android.com/reference/android/os/Build.VERSION.html#SDK_INT) to prevent older systems from executing new APIs works in this way on Android 2.0 (API level 5) and higher only. Older versions will encounter a runtime exception.

**TIPS:** Once the [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) finishes execution, the system calls the [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart()) and [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume()) methods in quick succession. Your activity never resides in the Created or Started states. Technically, the activity becomes visible to the user when [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart()) is called, but [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume()) quickly follows and the activity remains in the Resumed state until something occurs to change that, such as when a phone call is received, the user navigates to another activity, or the device screen turns off.

1. **onStart():-**

This callback is called by android OS when the activity is being started.

*protected void onStart()*

*{*

*super.onStart();*

*}*

1. **onResume():-**

This callback is called by android OS when the user starts interacting with screen.

*protected void onResume()*

*{*

*super.onResume();*

*}*

1. **onPause():-**

This callback is called by android OS when it is being partially visible, i. e. Activity goes in background.

*protected void onPause()*

*{*

*super.onPause();*

*}*

When the system calls [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity and it will soon enter the Stopped state. You should usually use the [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) callback to:

* Stop animations or other ongoing actions that could consume CPU.
* Commit unsaved changes, but only if users expect such changes to be permanently saved when they leave (such as a draft email).
* Release system resources, such as broadcast receivers, handles to sensors (like GPS), or any resources that may affect battery life while your activity is paused and the user does not need them.

For example, if your application uses the [Camera](http://developer.android.com/reference/android/hardware/Camera.html), the [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) method is a good place to release it.

@Override  
public void onPause() {  
    super.onPause();  // Always call the superclass method first  
  
    // Release the Camera because we don't need it when paused  
    // and other activities might need to use it.  
    if (mCamera != null) {  
        mCamera.release();  
        mCamera = null;  
    }  
}

Generally, you should **not** use [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) to store user changes (such as personal information entered into a form) to permanent storage. The only time you should persist user changes to permanent storage within [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) is when you're certain users expect the changes to be auto-saved (such as when drafting an email). However, you should avoid performing CPU-intensive work during [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()), such as writing to a database, because it can slow the visible transition to the next activity (you should instead perform heavy-load shutdown operations during [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop())).

1. **onStop():-**

This callback is called by android OS when the Activity is no longer visible.

*protected void onStop()*

*{*

*super.onStop();*

*}*

1. **onRestart():-**

This callback is called by android OS when the Activity is restarting after being stopped.

*protected void onRestart()*

*{*

*super.onRestart();*

*}*

1. **onDestroy():-**

This callback is called by android OS when the Activity is being terminated either by OS or by user.

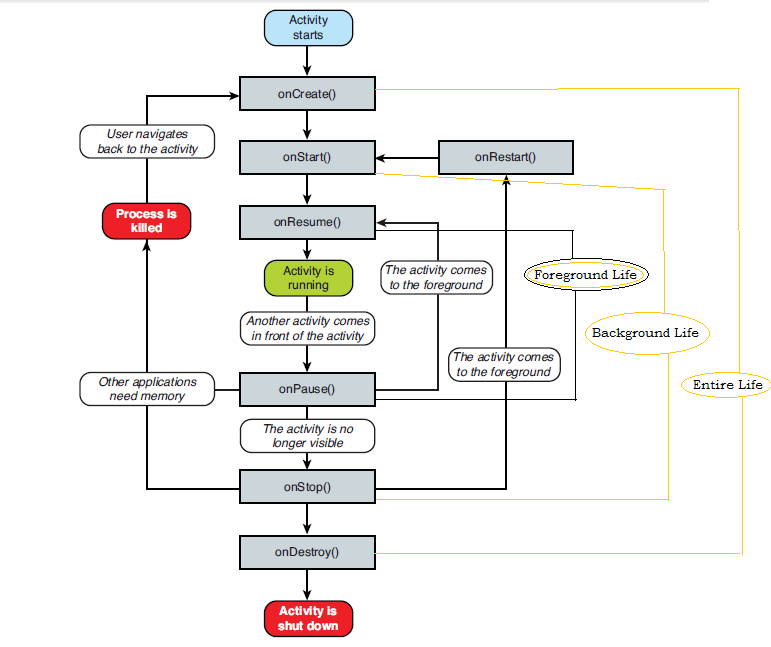
*protected void onDestroy()*

*{*

*super.onDestroy();*

*}*

Activity lifecycle diagram

****

**What is Logcat ?**

It is an output window where we can know status of running app. It is basically used for debugging purpose by printing some message on this Logcat window.

**How do we print message on this Logcat window?**

By using Log class.

**Log class:-**

It is the class define in android.util package. This class has some commonly use static method to print something in different colors on LogCat Window.

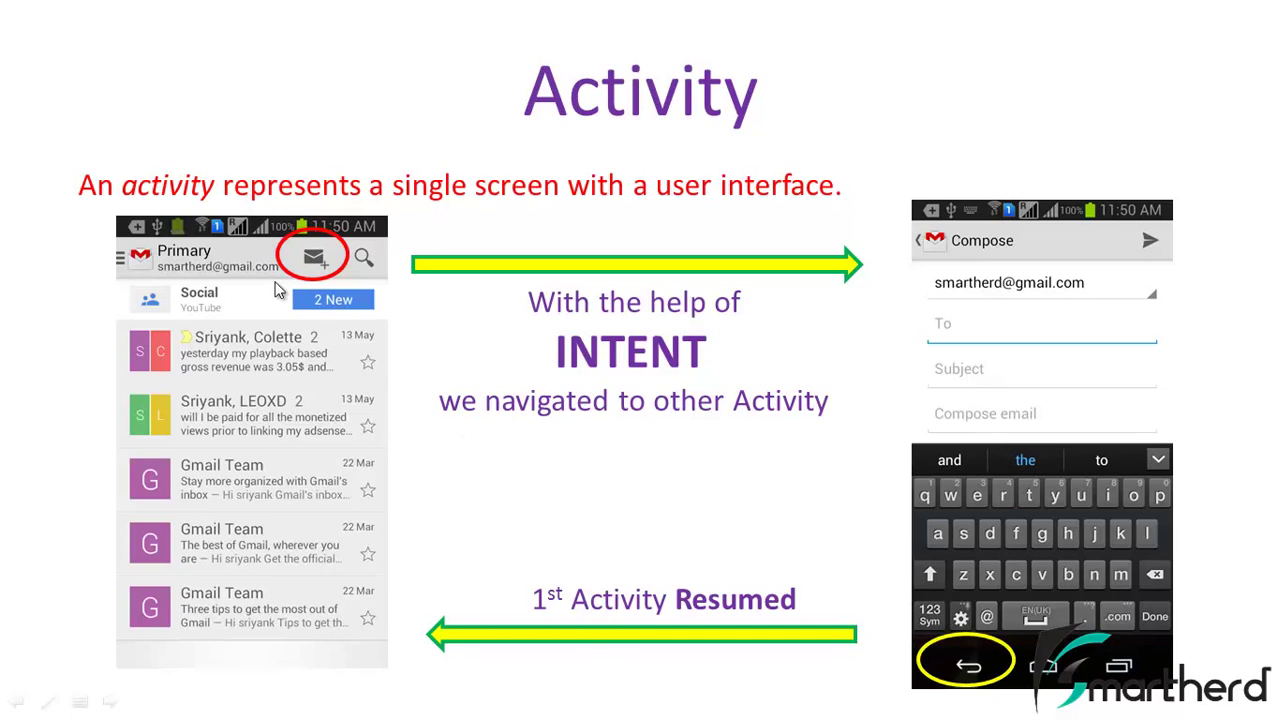
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Message color** | **Purpose** |
| d(String tag,String msg) | String tag- Print under TAG Column of LogCat window.  String msg- Print under TEXT Column of LogCat window. | Black | For debugging |
| i(String tag,String msg) | String tag- Print under TAG Column of LogCat window.  String msg- Print under TEXT Column of LogCat window. | Green | For information |
| e(String tag,String msg) | String tag- Print under TAG Column of LogCat window.  String msg- Print under TEXT Column of LogCat window. | Red | For errors |
| w(String tag,String msg) | String tag- Print under TAG Column of LogCat window.  String msg- Print under TEXT Column of LogCat window. | Yellow | For warning |
| v(String tag,String msg) | String tag- Print under TAG Column of LogCat window.  String msg- Print under TEXT Column of LogCat window. | Blue | For Verbose |

**Q. How do we use them?**

A. Log.d(“LIFECYCLE”,”onCreate() is Running…….”);

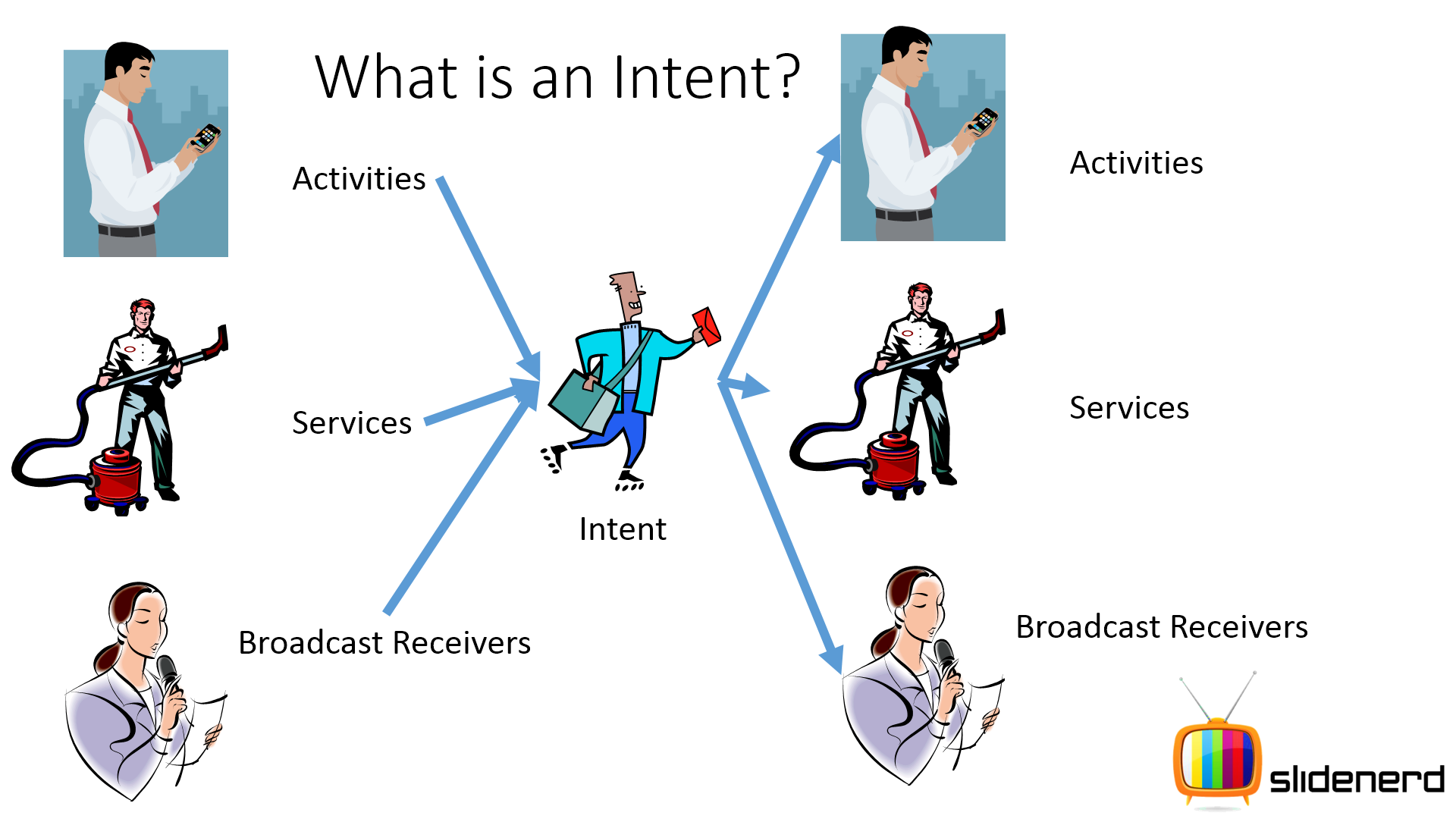
**Q. How do navigate (transfer control) either between same or different component?**

A. We can navigate one component to another component (same or different) with the help of intent.



**Q. What is intent?**

A. In general, intent is the standard message in android that expresses the intention of one component to perform the work by any other component. Intent allows the component to interact with other components defining by you or already installed in android OS.



Programmatically, an intent is a class defined in android.content package and object of the this class contains passive data structure holding an abstract description about an operation to be performed by any component.

NOTE- To Broadcast intent as a message by one component to another component either with in the very same app or the outside of the app.

**Q. What does Intent do?**

A. 1. To start any component (Activity, Service, BroadCastReciever).

2. To start the camera.

3. To display webpage by starting browser app.

4. To dial phone number or answer to a phone call etc.

## Building an Intent

An [Intent](http://developer.android.com/reference/android/content/Intent.html) object carries information that the Android system uses to determine which component to start (such as the exact component name or component category that should receive the intent), plus information that the recipient component uses in order to properly perform the action (such as the action to take and the data to act upon).

An intent object contains following 6 fields based on what it is communicating or what action to be performed.

The primary information contained in an [Intent](http://developer.android.com/reference/android/content/Intent.html) is the following:

1. **ComponentName:-** This field of an intent object defines what component receives this intent object.

This field of the [Intent](http://developer.android.com/reference/android/content/Intent.html) is a [ComponentName](http://developer.android.com/reference/android/content/ComponentName.html) object, which you can specify using a fully qualified class name of the target component which followed by .class( including the package name of the app). For example,com.example.ExampleActivity.class. You can set the component name with [setComponent()](http://developer.android.com/reference/android/content/Intent.html#setComponent(android.content.ComponentName)), [setClass()](http://developer.android.com/reference/android/content/Intent.html#setClass(android.content.Context, java.lang.Class<?>)),[setClassName()](http://developer.android.com/reference/android/content/Intent.html#setClassName(java.lang.String, java.lang.String)), or with the [Intent](http://developer.android.com/reference/android/content/Intent.html) constructor.

1. **Action :-**

This field of an intent object is a string which defines what operation one component is wanted to perform by another component. This field is mandatory part (When doesn’t have ComponentName field) of intent object. The Intent class has number of predefined action (String constant) contains for different-different operations.

* **Example Predefined Action:**
* **ACTION\_VIEW-**

Defined as a public static constant within the Intent class

Used as an ACTION with Activities when the calling Activity wants to display some piece of information such as a photo to view in a gallery app, or an address to view in a map app and there may be a target Activity that can perform this ACTION

* ACTION Strings are defined in different Android classes such as Intents, Settings and other API classes.
* Some other Predefine ACTIONS
* Final String ACTION\_CALL ->Perform a call to someone specified by data associated with ACTION field in intent object.
* Final String ACTION\_ANSWER -> Handling an incoming phone call.
* Final String ACTION\_BATTERY\_LOW -> Indicates the battery is being low.
* **Example Custom-Defined Action:**

public static final String DOWNLOAD\_SONG="com.cetpa.MusicApp.DOWNLOAD\_SONG";

You can define your own actions, be sure to include your app's package name as a prefix. For example:

static final String ACTION\_TIMETRAVEL = "com.example.action.TIMETRAVEL";

You can specify the action for an intent with [setAction()](http://developer.android.com/reference/android/content/Intent.html#setAction(java.lang.String)) or with an [Intent](http://developer.android.com/reference/android/content/Intent.html) constructor.

NOTE- The action largely determines how the rest of the intent is structured—particularly what is contained in the data and extras.

1. **Data Field :-**

This field of an intent object defines data and type of data (mime type) in the form of URI (Uri object), an action to be performed on it (define by action field) by other components (to whom intent object has sent).

NOTE:-The primary pieces of information in the intent are Action and Data because sometime the action which should be performed with data which provides content for the action.

For example, if the action is ACTION\_EDIT, the data should contain the URI of the document to edit.

* Specify the **MIME type** of data along with its Uri.
* An activity that’s able to display images probably won’t be able to play an audio file, even though the URI formats could be similar.
* So specifying the MIME type of your data helps the Android system find the best component to receive your intent.
* To set only the data URI, call**setData().**
* To set only the MIME type, call **setType().**
* You can set both explicitly with **setDataAndType().**

**Note- A**lways use **setDataAndType()** to set both URI and MIME type.

What is a MIME type?

MIME stands for Multi-purpose Internet Mail Extensions. MIME types form a standard way of classifying file types on the Internet.

similar to file extensions but more universally accepted, "MIME types" are used to identify the type of information that a file contains.

A MIME type has two parts: a type and a subtype. They are separated by a slash (/). For example, the MIME type for Microsoft Word

files is application and the subtype is msword. Together, the complete MIME type is application/msword.

following are some MIME types for different extension files-

Type Common File Extension Purpose

text/html .html Web Page

image/png .png PNG-format image

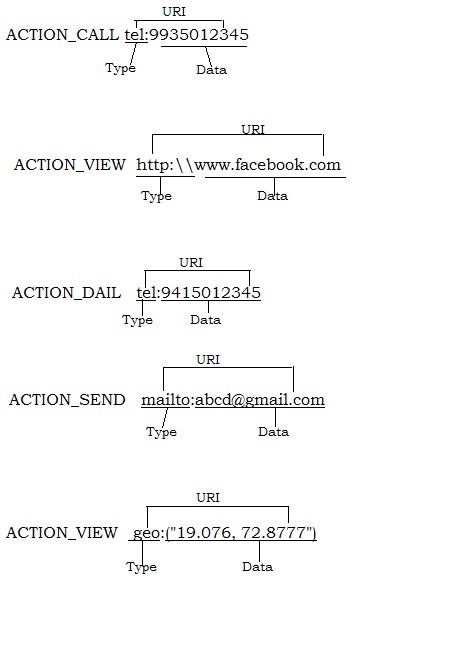
image/jpeg .jpeg JPEG-format image

audio/mpeg .mp3 MPEG Audio File

application/octet-stream .exe Best for downloads that should just be saved to disk

messgae/rfc822 .eml Email messages

Example:-



**Q.-How to create URI?**

A.- By Using static method *parse(String uriString)* of Uri class define in android.net package.

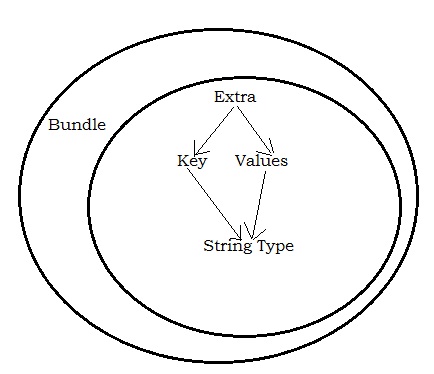
Example :- Uri uri=Uri.parse(“geo:19.076, 72.8777”);

MIME type- similar to file extension, however it tells what type data a file contains.

Note :- In addition to these primary attributes, there are a number of secondary attributes that you can also include with an intent are :-

1. **Extra:-**

This field of an Intent object is represented by Bundle Object which provides a facility to add some additional information in it required to accomplish the requested action in the form of key value pair. Just same as few action uses particular kind of data (Uri), few action also uses particular extras.

For example, if we have an action to send an e-mail message, we could also include extra pieces of data here to supply a subject, body, etc..

key is of String type and values are of any primitive type inside Bundle object as an extra field of intent object .

**Note:-** We have many predefine keys for extra field of an intent object in Intent class and we can also define custom keys for extra field of an object explicitly.

**Example-**

**Intent.EXTRA\_TEXT, Intent.EXTRA\_EMAIL etc..**

**Q.- How to set and get extra field?**

A.- By using putExtra() and getExtra() methods of Intent class.

**5. Categories: -** This String field of an intent object contains extra information needed to determine who should handle this Intent.

* Android defines a number of predefined categories and you can also define your categories which is rather uncommon.
* **Example Predefined Category:**
* **CATEGORY\_BROWSABLE**

The Activity that declares this category in its IntentFilter in the Manifest is treated as an Activity that lets itself be used by the Web browser to display an image or a link.

* **CATEGORY\_LAUNCHER**

The Activity that declares this category in its IntentFilter in the Manifest is treated as the initial Activity or Main Activity for that particular app and is displayed in the Launcher Screen where all the apps are placed.

Use addCategory() to specify category .

NOTE- Custom categories should use the package name as a prefix, to ensure that they are unique.

static final String MY\_CATEGORY = "com.example.category.UNIQUE";

### E:\documents\Study Stuff\Android snapshots\Intent\11.png

### 6. \*FLAGS:

* These flags are optional part of intent object and instruct Android OS how to launch an activity, and how to treat it after its launched etc.
* Act as metadata defining certain things such as
  + Under which task should the Activity be launched inside?

Example- FLAG\_ACTIVITY\_CLEAR\_TASK , FLAG\_ACTIVITY\_NEW\_TASK

* Use the **setFlags()**method to control such aspects.

**NOTE- An Intent Object can have any no of Actions, Categories, data and extra fields as per need of Intent building.**

Constructors of Intent class to create intent objects:-

1. Intent();
2. Intent(Intent i);
3. Intent(String ACTION);
4. Intent(String ACTION, Uri uri);
5. Intent(Context context, Class c);
6. Intent(String ACTION, Uri uri, Context context, Class c);

**Types of Intent:-**

There are basically two types of intents.

1. Explicit intent.
2. Implicit intent.
3. **Explicit intent:-**

An intent object will be treated as explicit intent when sender component is aware about which component to call to perform some action.

**Q.-How to create Explicit intent?**

**A.-** *Intent intent =new Intent(Context,ActivitySecond.class);*

For Example:- when an explicit intent sends to the Android system. If the class specified in the intent represents an activity, the Android system starts it.

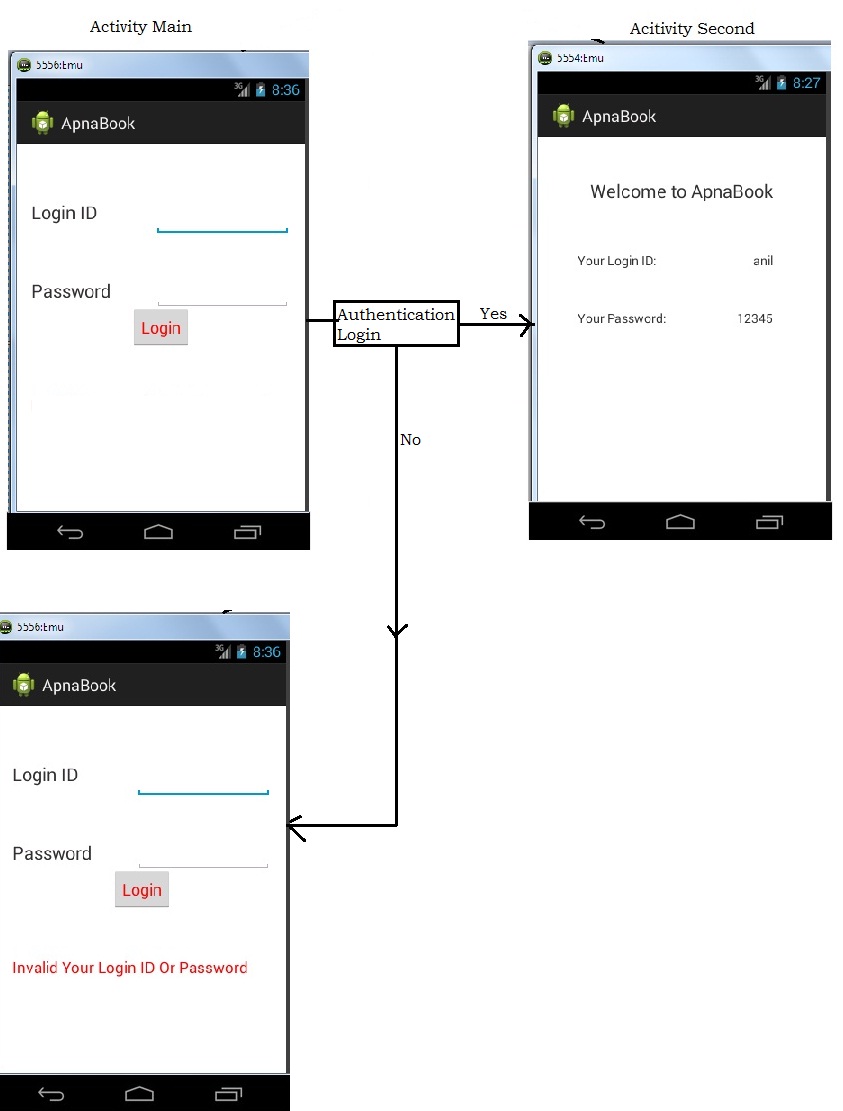
Intent i = **new** Intent(**this**, ActivityTwo.**class**);

i.putExtra("key1", "This value one for ActivityTwo ");

i.putExtra("key2", "This value two for ActivityTwo");

NOTE-An explicit intent is typically used to start a component in your own app, because you know the class name of the activity or service you want to start.

### Sub Activity:- Activities which are started by other Android activities are called sub-activities. This wording makes it easier to describe which activity is meant.

**Q.- How to Broadcast an explicit intent?**

**A.-** By using following methods of an activity class.

startActivity(intent);

startActivities (intent);

startActivityForResult()

startServices()

startSendBroadcast()

1. **Implicit** **Intent: -** An intent object will be treated as implicit when sender component does not aware of which component is to be exactly performed your desired action.

In case of implicit intent, Android OS (package manager) chooses an appropriate component on the basis of action, data, and category fields of the intent object, to perform desired action.

For example: - If you want to display an URL then we have to create an intent object and set action, and data field with ACTION\_VIEW and TYPE:<http://www.facebook.com>, in intent object, then broadcast this intent, In this case Android OS will decide which component of which application (browser application) will perform desired action.

**Question: - How to create implicit intent?**

**Answer:** - Intent intent=new Intent (String action);

OR

Intent intent=new Intent (String action, Uri uri);

Example:-Intent intent=new Intent (ACTION\_VIEW, Uri.parse(“http:\\www.facebook.com”));

startActivity (intent);

**Question: - How to broadcast implicit intent?**

**Answer: -** By following method –

startActivity()

startActivities()

startActivityForResult()

startServices()

startSendBroadcast()

onCreate()

intent

intent

startActivity(intent);

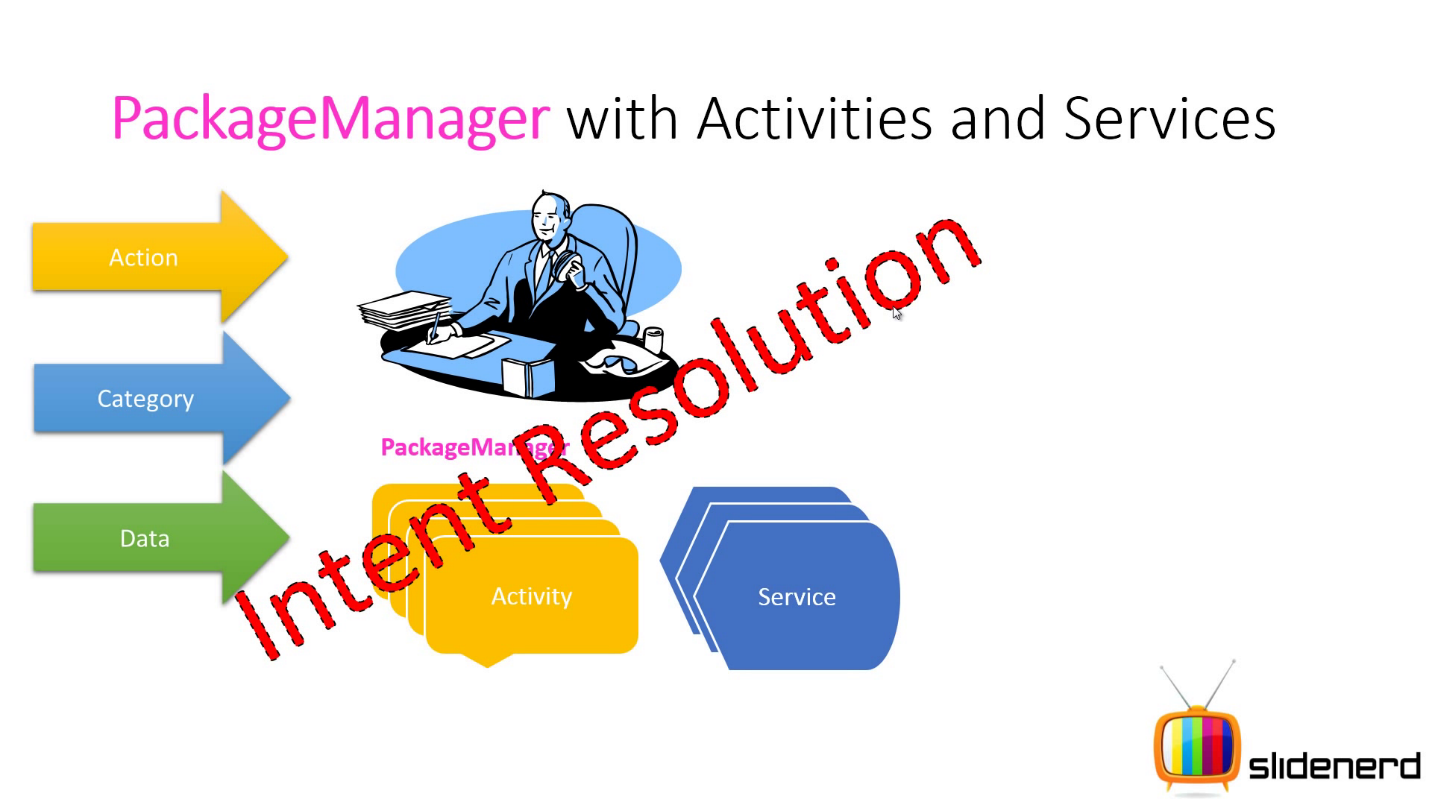
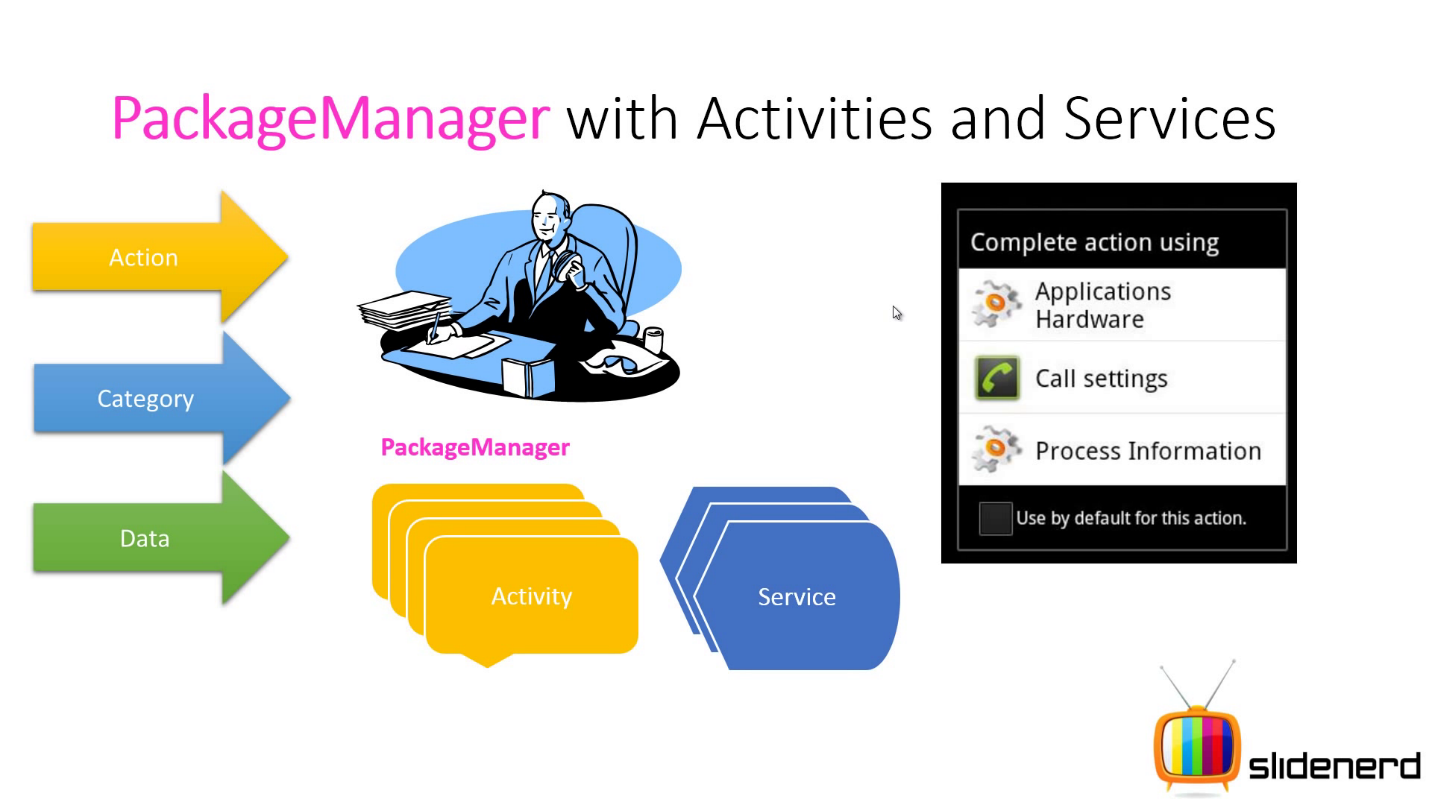
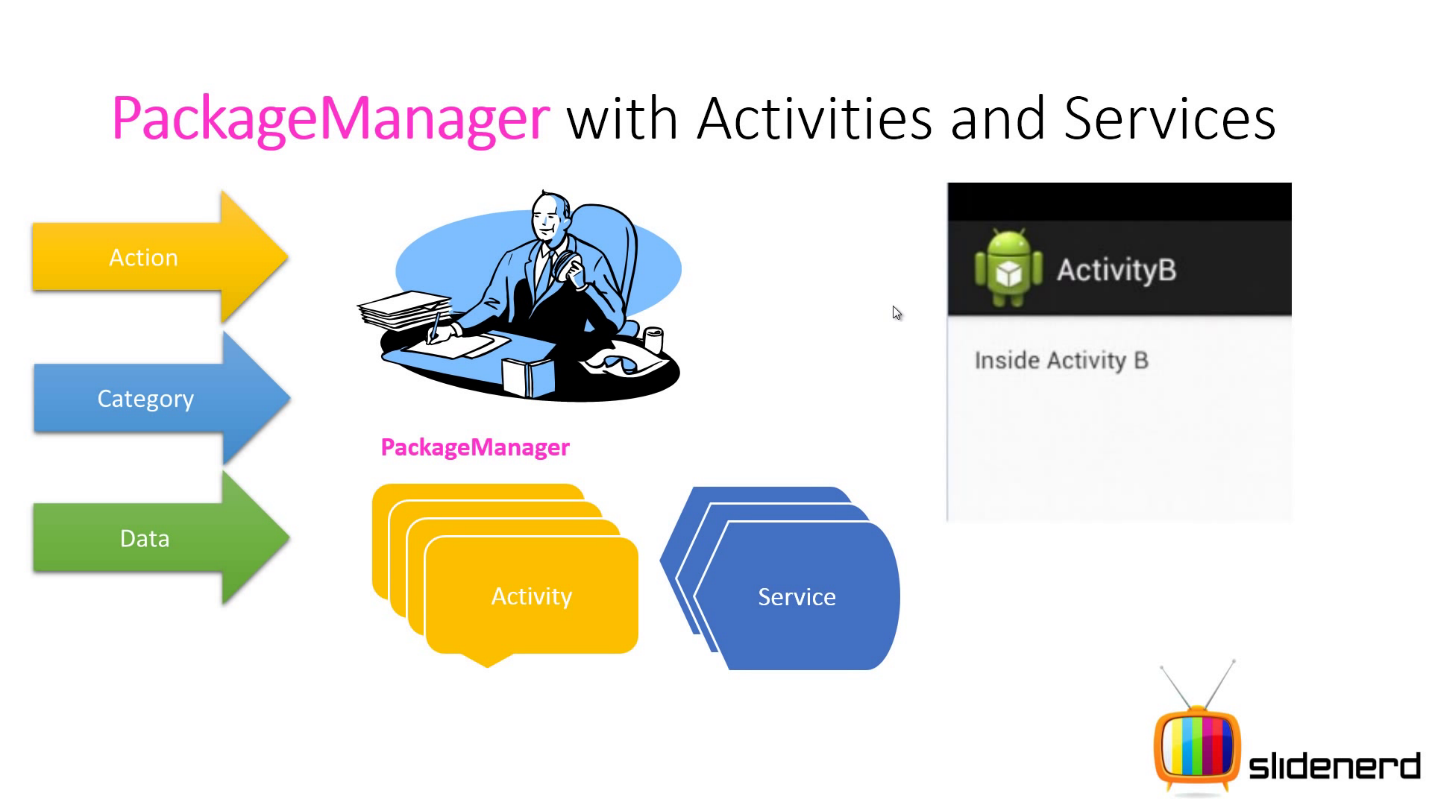
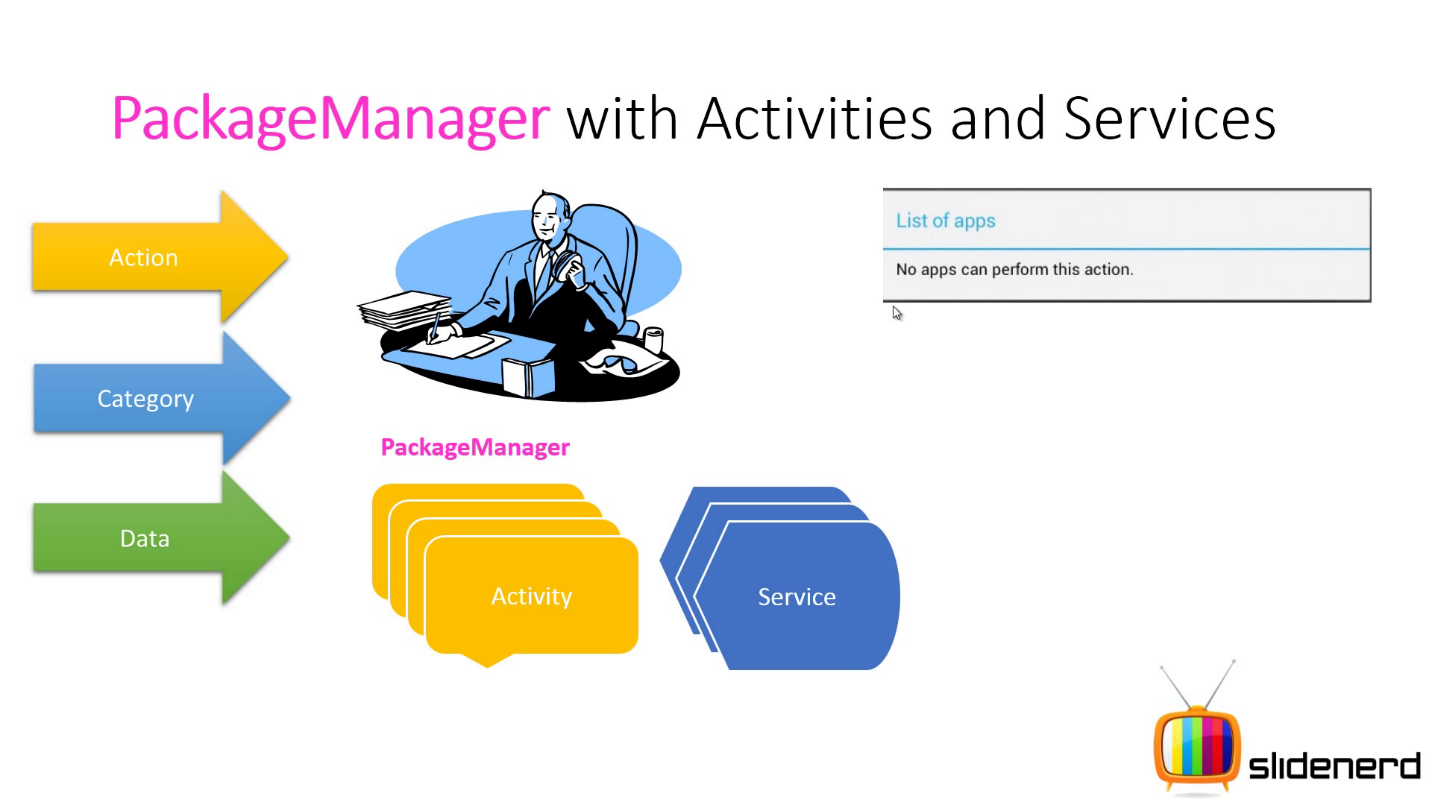
Activity B

Android OS

ActivityA

If an implicit intent is sent to the Android system, it searches for all components which are registered to Package Manager for the specific action and the fitting data type.

If only one component is found, Android starts this component directly. If several components are identified by the Android system, the user will get a selection dialog and can decide which component should be used for the intent.



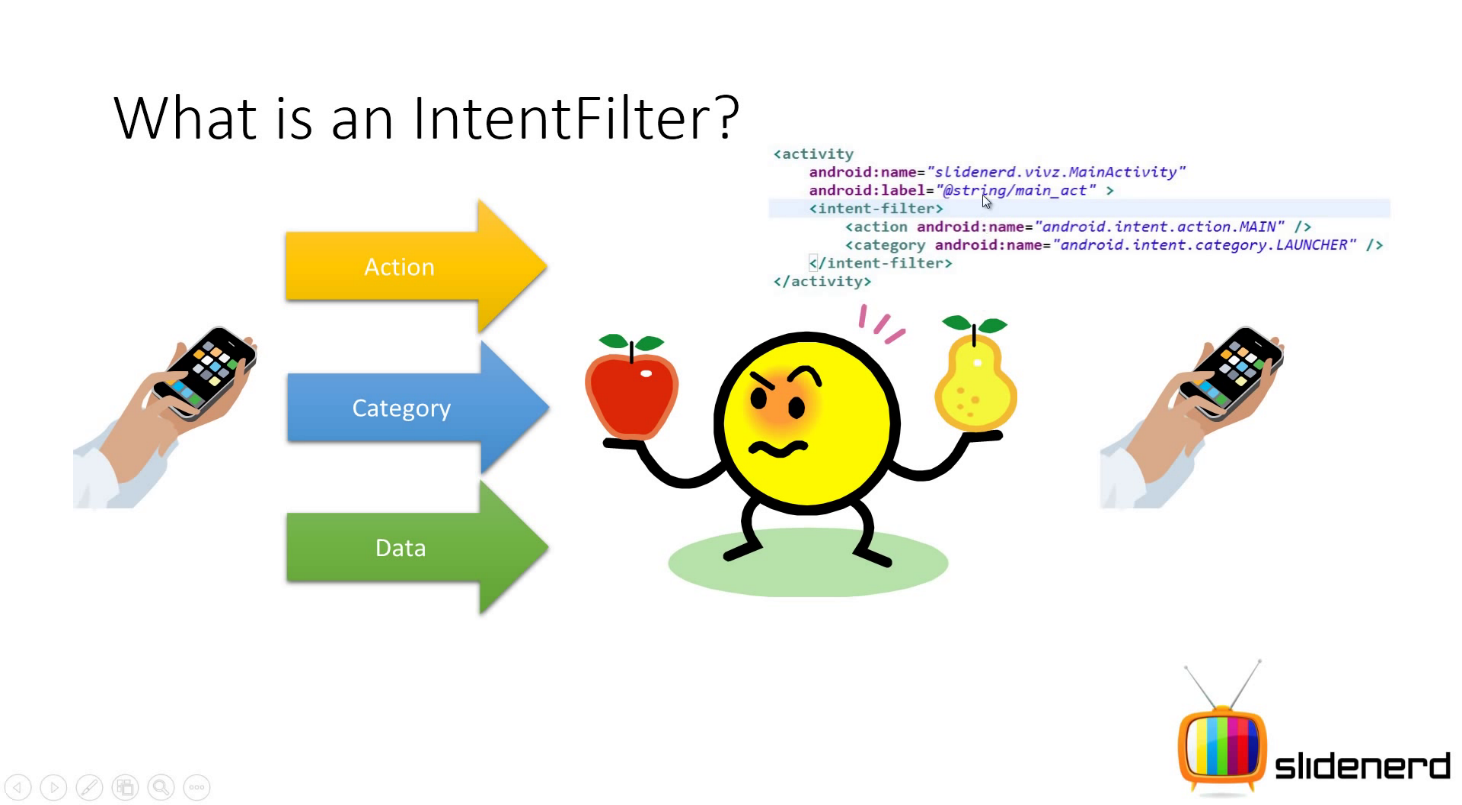
Note:- Component which is capable to handle an implicit intent, must be registered with <intent-filter> in manifest xml file

**Question: - What is intent filter?**

**Answer: -** Intent filters are simply advertisers about any component to the package manager by saying what action, category or data of an intent object, the particular component can handle.

To advertise which implicit intent, your application can receive, declare intent filters for each of your application components with an <Intent-Filter> element in manifest file.

The system will deliver an implicit intent to your application component only if the intent can pass through one of your intent filter by Package Manager.

****

Launch any component after best match

**<Intent-Filter>** must be defined for implicit intent because an explicit intent always delivered to its target, regardless of any Intent-Filter declare for any component.

**Question: - Where do we define <intent-filter> element?**

**Answer: -** We can define <intent-filter> element in any of following element of <application> element in manifest file.

<activity> element

<service> element

<receiver> element

**Question: - What do we write inside an <intent-filter> element**

**Answer: -** We can define –

1. One or more <action> element to indicate what are different and possible actions can be handled by any component.

<intent-filter>  
    <action android:name="android.intent.action.EDIT" />  
    <action android:name="android.intent.action.VIEW" />  
    ...  
</intent-filter>

1. One or more <category> element to indicate what grouping does your component supports.

<intent-filter>  
    <category android:name="android.intent.category.DEFAULT" />  
    <category android:name="android.intent.category.BROWSABLE" />  
    ...  
</intent-filter>

**Note:** In order to receive implicit intents, you must include the [CATEGORY\_DEFAULT](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_DEFAULT) category in the intent filter. The methods [startActivity()](http://developer.android.com/reference/android/app/Activity.html#startActivity(android.content.Intent)) and [startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult(android.content.Intent, int)) treat all intents as if they declared the [CATEGORY\_DEFAULT](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_DEFAULT) category. If you do not declare it in your intent filter, no implicit intents will resolve to your activity.

1. One or more <data> element to indicate what type of data can be handled by any component.

<intent-filter>  
    <data android:mimeType="video/mpeg" android:scheme="http" ... />  
    <data android:mimeType="audio/mpeg" android:scheme="http" ... />  
    ...  
</intent-filter>

Where-

ATTRIBUTES:

android:scheme

The scheme part of a URI. This is the minimal essential attribute for specifying a URI; at least one scheme attribute must be set for the filter, or none of the other URI attributes are meaningful.

A scheme is specified without the trailing colon (for example, http, rather than http:).

\*NOTE- If the filter has a data type set (the [mimeType](http://developer.android.com/guide/topics/manifest/data-element.html#mime) attribute) but no scheme, the content: and file:schemes are assumed.

android:mimeType

A MIME media type, such as image/jpeg or audio/mpeg4-generic. The subtype can be the asterisk wildcard (\*) to indicate that any subtype matches.