

1. Which of the following Java code samples could be used to apply an XML layout resource to an Activity?

- A. `findViewById(R.id.help);`
- B. **`setContentView(R.layout.help);`** ✓
- C. `addContentView(R.layout.help);`
- D. `setVisible(true);`

2. Which of the following defines the padding for an XML View?

- A. `android:layout_padding`
- B. **`android:padding`** ✓
- C. `android:view_padding`
- D. `android:padding_width`

3. Which of these methods is called first when an Activity is initially launched?

- A. `onStart()`
- B. `onPause()`
- C. **`onCreate()`** ✓
- D. `onResume()`

4. Which of the following XML layout excerpts would allow you to identify a View in Java as follows: `findViewById(R.id.submit_btn);`

- A. `android:id="@id/submit_btn"`
- B. `android:id="@+name/submit_btn"`
- C. **`android:id="@+id/submit_btn"`** ✓
- D. `android:id="@string/submit_btn"`

5. Which of the following is not one of the pre-defined Android View Groups?

- A. `RelativeLayout`
- B. **`FixedLayout`** ✓
- C. `GridLayout`
- D. `TableLayout`

6. In which of the following locations would you include XML code to define a menu resource, including the menu items?

- A. `res/xml/main_menu.xml`
- B. **`res/menu/main_menu.xml`** ✓
- C. `res/menus/main_menu.xml`
- D. `res/layout/main_menu.xml`

7. In apps developed for Android 3.0 and higher, options menu items appear in the Action Bar.

- A. **True** ✓
- B. False

8. When does a context menu appear?

- A. on pressing
- B. **on long-pressing** ✓

- C. on pressing the menu button
- D. on swiping

9. What does the following specify when placed in the XML for a menu item:

`android:showAsAction="ifRoom"`

- A. the item must always appear in the Action Bar
- B. the item must always appear in the overflow menu
- C. the item should appear in the Action Bar if there is room, otherwise appearing in the overflow menu ✓
- D. the item should appear in the overflow menu if there is room, otherwise appearing in the Action Bar

10. Which of the following correctly assigns the content of a TextView using a value defined in the `res/values/strings.xml` file as follows: `<string name="info_text">Here is the info</string>`

- A. `android:text="@string/info_text"`
- B. `android:text="@+id/info_text"`
- C. `android:value="@strings/info_text"`
- D. `android:text="@+strings/info_text"` ✓

11. Which of the following must be supplied for each View in an Android XML layout?

- A. `id` and `layout_width`
- B. `width` and `height`
- C. `id` and `layout_weight`
- D. `layout_width` and `layout_height` ✓

12. Which of the following defines a row within the XML for a `TableLayout`?

- A. `<TableRow>` ✓
- B. `<Row>`
- C. `<TableRowView>`
- D. `<RowView>`

13. Which of the following is a typical example of gaining a reference to a View item, for example in order to attach an event listener to it?

- A. `Button infoBtn = (Button) findViewById(R.id.info_btn);` ✓
- B. `View infoView = new View(this);`
- C. `Button infoBtn = (Button) getView(R.id.info_btn);`
- D. `View infoView = (View) getContext();`

14. Which of the following defines a left margin of 10 pixels in an XML View item?

- A. `android:marginLeft="10px"`
- B. `android:layout_margin_left="10px"`
- C. `android:margin_left="10px"`
- D. `android:layout_marginLeft="10px"` ✓

15. Which of the following classes is used to display a message in an Android app only for a moment?

- A. Context
- B. **Toast**
- C. Alert
- D. Message

16. Which of the following is used to bind data to a layout View?

- A. Activity
- B. Context
- C. Intent
- D. **Adapter**

17. Which of the following correctly applies a style saved within an XML file in the res/values folder to a View in a layout file?

- A. **style="@style/MyStyle"**
- B. android:style="@style/MyStyle"
- C. android:style="@+style/MyStyle"
- D. android:style="MyStyle"

18. Which of the following is the correct outline for the "onClick" method in an OnClickListener?

- A. public void onClick(Activity a)
- B. **public void onClick(View v)**
- C. public void onClick(Context c)
- D. public View onClick(this)

19. Which of the following is a method of the View class which allows you to set the Layout Parameters, passing a ViewGroup as parameter?

- A. **setLayoutParams**
- B. setLayoutParameters
- C. setParameters
- D. layoutParams

20. Which of the following Java methods returns the actual width of a View when the layout has been applied and the View drawn on screen?

- A. getLayoutWidth
- B. **getWidth**
- C. getMeasuredWidth
- D. getViewWidth

21. Which operating systems support Android SDK Development?

- A. Windows
- B. Mac
- C. Linux
- D. **All of the above**

22. Which integrated development environment (IDE) is best suited for beginner Android developers?

- A. Microsoft Visual Studio
- B. **Android Studio**
- C. JBuilder
- D. AndroidDev

23. What tool can developers use to download Android SDK versions, samples, and other development tools for app development?

- A. The Android Tool Manager
- B. The Android Developer Plug-in
- C. And-Doc
- D. **The Android SDK Manager**

24. What is an AVD?

- A. An Android Virus Definition
- B. **A set of device configuration details for use with the Android emulator**
- C. An Android code versioning module
- D. A file used to define an Android app's name, icon, and other settings

25. True or False? You will only need to create one AVD in order to debug your applications

- A. True
- B. **False**

26. What is one way can you connect Android devices to your development environment for debugging?

- A. Port 411
- B. DebugLink technology
- C. **USB**
- D. None of the above. You can only debug your apps in the emulator, not devices.

27. If your app is building but is not installing properly on your device, what is the most likely reason?

- A. The app is built for an Android SDK version that is not compatible with your device.
- B. The device is not connected to your development machine properly.
- C. The device settings have not enabled USB debugging.
- D. **All of the above.**

28. What does ADB stand for?

- A. Android Data Bot
- B. **Android Debug Bridge**
- C. Android Device Bridge
- D. Active Digital Brain

29. \_\_\_\_\_ represents a single user interface screen
- a. Service
  - b. Broadcast Receiver
  - c. Content Provider
  - d. Activity
30. \_\_\_\_\_ is used to uniquely identify the framework API revision offered by a version of the Android platform
- a. Version Number
  - b. API Level
  - c. Code Name
31. Which among the following is not a member of Open Handset Alliance
- a. Google
  - b. Apple
  - c. Wipro
  - d. None of these
  - e. Samsung
32. Specify the directory name where the XML layout files are stored
- a. /assets
  - b. /res/values
  - c. /res/layout
  - d. /src
33. The root element of AndroidManifest.xml is :
- a. manifest
  - b. application
  - c. activity
  - d. action
34. What is the use of AndroidManifest.xml file?
- a. It declares the permissions of the application
  - b. It declares the minimum level of the Android API that the application requires
  - c. It facilitates to provide a unique name for the application by specifying package name
  - d. All
  - e. It describes the components of the application
35. Android is initially developed by
- a. Apple Inc
  - b. Android Inc
  - c. Open Handset Alliance
  - d. Google Inc
36. Which attribute of the element <uses-sdk> is used to specify the minimum API Level required for the application to run?  
Select one:
- a. android:minSdkVersion
  - b. android:maxSdkVersion

c. android.targetSdkVersion

37. What is the name of the class which is inherited to create a user interface screen?

- a. **Activity**
- b. None of these
- c. ViewGroup
- d. View

38. The basic building element of Android's user interface is called

- a. Activity
- b. ViewGroup
- c. **View**
- d. ContentProvider
- e. Layout

39. Select the method used to access a view element of a layout resource in an activity  
Select one:

- a. setContentView()
- b. None of these
- c. onCreate()
- d. **findViewById()**

40. A Layout where the positions of the children can be described in relation to each other or to the parent?

- a. Grid Layout
- b. Frame Layout
- c. Linear Layout
- d. **Relative Layout**

41. Select an attribute that specifies how to place the content of an object, both on the x- and y-axis, within the object itself

- a. layoutDirection
- b. layout\_width
- c. **gravity**
- d. layout\_centerHorizontal

42. All layout classes are direct subclass of

- a. **android.view.View**
- b. **android.view.ViewGroup**
- c. android.widget.AbsoluteLayout
- d. java.lang.Object

43. Select the unit, that is not recommended for specifying dimensions

- a. Density Independent Pixels
- b. **Pixels**
- c. Points
- d. Scale Independent Pixels

44. Choose the layout, that is deprecated

- a. Linear Layout
- b. Relative Layout
- c. **Absolute Layout**
- d. Frame Layout

45. A layout that arranges its children into rows and columns?

- a. Frame Layout
- b. Absolute Layout
- c. Linear Layout
- d. **Table Layout**

46. Select one:

a.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

**<TextView**

```
android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="@string/hello_world"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

b.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

**<TextView**

```
android:id="@+id/tv_helloworld"
android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="@string/hello_world"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

c.

**<RelativeLayout**

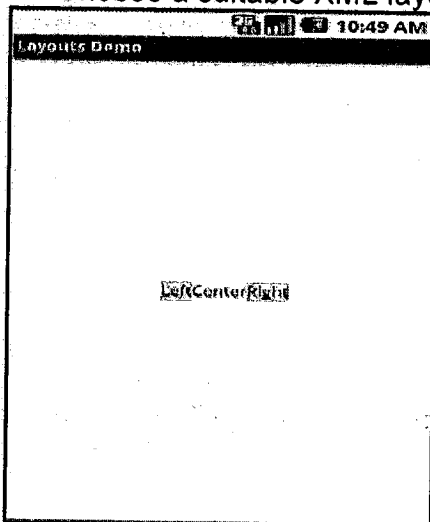
```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

**<TextView**

```
android:id="@+id/tv_helloworld"
android:layout_width="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="@string/hello_world"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

47. Choose a suitable XML layout file for the given below activity screen :



Select one:

a.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

**<TextView**

```
android:id="@+id/center"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/center"
```



```
android:background="@color/center"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
tools:context=".MainActivity" />
```

#### **<TextView**

```
android:id="@+id/left"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/left"
android:background="@color/left"
android:layout_toLeftOf="@id/center"
tools:context=".MainActivity" />
```

#### **<TextView**

```
android:id="@+id/right"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/right"
android:background="@color/right"
android:layout_toRightOf="@id/center"
tools:context=".MainActivity" />
```

#### **</RelativeLayout>**

b.

#### **<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

#### **<TextView**

```
android:id="@+id/center"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/center"
android:background="@color/center"
android:layout_centerInParent="true"
tools:context=".MainActivity" />
```

#### **<TextView**

```
android:id="@+id/left"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/left"
```

```
android:background="@color/left"
android:layout_toLeftOf="@id/center"
android:layout_centerVertical="true"
tools:context=".MainActivity" />
```

```
<TextView
android:id="@+id/right"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/right"
android:background="@color/right"
android:layout_toRightOf="@id/center"
android:layout_centerVertical="true"
tools:context=".MainActivity" />
```

```
</RelativeLayout>
```

c.

```
<RelativeLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

```
<TextView
```

```
android:id="@+id/center"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/center"
android:background="@color/center"
android:layout_centerInParent="true"
tools:context=".MainActivity" />
```

```
<TextView
```

```
android:id="@+id/left"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/left"
android:background="@color/left"
android:layout_toRightOf="@id/center"
android:layout_centerVertical="true"
tools:context=".MainActivity" />
```

```
<TextView
```

```
android:id="@+id/right"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:text="@string/right"
android:background="@color/right"
android:layout_toLeftOf="@id/center"
android:layout_centerVertical="true"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

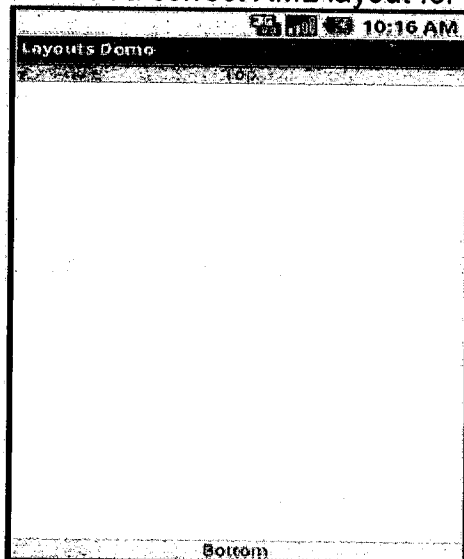
48. Select a mandatory attribute for any view inside of a containing layout manager

- a. contentDescription
- b. id
- c. text
- d. layout\_width

49. A Layout that arranges its children in a single column or a single row?

- a. Relative Layout
- b. Absolute Layout
- c. GridLayout
- d. Linear Layout

50. Pick a correct XML layout for the given below activity screen :



Select one:

a.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center_horizontal"
>
```

**<TextView**

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/top"
android:background="@color/top"
android:layout_alignParentTop="true"
tools:context=".MainActivity" />
```

**<TextView**

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/bottom"
android:background="@color/bottom"
android:layout_alignParentBottom="true"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

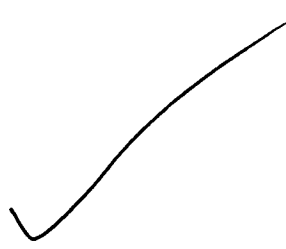
**b.**

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

**<TextView**

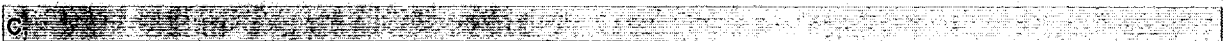
```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/top"
android:background="@color/top"
android:layout_alignParentTop="true"
android:gravity="center"
tools:context=".MainActivity" />
```



**<TextView**

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/bottom"
android:background="@color/bottom"
android:layout_alignParentBottom="true"
android:gravity="center"
tools:context=".MainActivity" />
```

**</RelativeLayout>**



### **<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:tools="http://schemas.android.com/tools"  
android:layout_width="match_parent"  
android:layout_height="match_parent" >
```

### **<TextView**

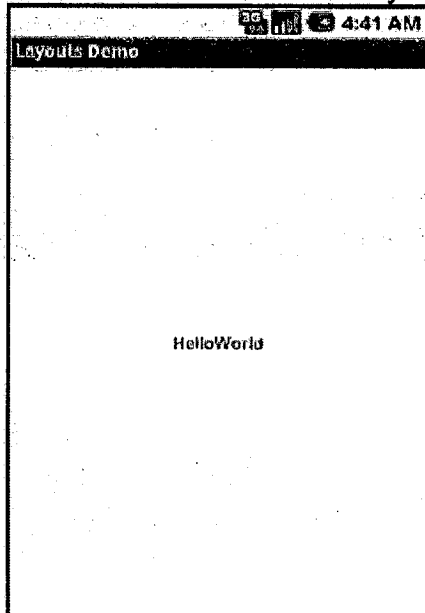
```
android:layout_width="fill_parent"  
android:layout_height="wrap_content"  
android:text="@string/top"  
android:background="@color/top"  
android:layout_alignParentTop="true"  
android:layout_centerHorizontal="true"  
tools:context=".MainActivity" />
```

### **<TextView**

```
android:layout_width="fill_parent"  
android:layout_height="wrap_content"  
android:text="@string/bottom"  
android:background="@color/bottom"  
android:layout_alignParentBottom="true"  
android:layout_centerHorizontal="true"  
tools:context=".MainActivity" />
```

### **</RelativeLayout>**

51. Select the correct xml layout for the below screen :



Select one:

a.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
>
```

**<TextView**

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:gravity="center"
android:text="HelloWorld"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

b.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

```
<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="HelloWorld"
tools:context=".MainActivity" />
```

**</RelativeLayout>**

c.

**<RelativeLayout**

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >
```

**<TextView**

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerInParent="false"
android:text="HelloWorld"
tools:context=".MainActivity" />
```

52. A layout that lets you specify exact locations (x/y coordinates) of its children

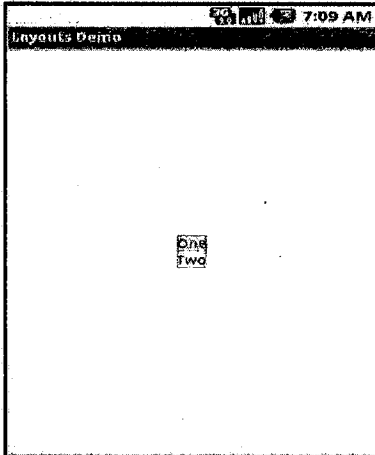
a. Frame Layout

b. **Absolute Layout**

c. Linear Layout

d. Relative Layout

53. Choose the correct XML layout for the given below activity screen



Select one:

a.

**<LinearLayout**

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center" >

**<TextView**

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/one"

android:background="@color/one"

tools:context=".MainActivity" />

**<TextView**

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/two"

android:background="@color/two"

tools:context=".MainActivity" />

**</LinearLayout>**

b.

**<LinearLayout**

xmlns:android="http://schemas.android.com/apk/res/android"

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
```

#### <TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/one"
android:background="@color/one"
tools:context=".MainActivity" />
```

#### <TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/two"
android:background="@color/two"
tools:context=".MainActivity" />
```

#### </LinearLayout>

c.

#### <LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" ✓
android:gravity="center" >
```

#### <TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/one"
android:background="@color/one"
tools:context=".MainActivity" />
```

#### <TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/two"
android:background="@color/two"
tools:context=".MainActivity" />
```

#### </LinearLayout>



**54. \_\_\_\_\_ tool is used to access database and manipulate it**

Ans1. sqlite3

Ans2. sqlplus

Ans3. ddms

Ans4. database

Correct Answer : 1

**55. In an explicit intent, which of the following is present?**

Ans1. Action

Ans2. Category

Ans3. URI

Ans4. Component

Correct Answer : 4

**56. In the life cycle of Activity, immediately after which event onPause event occurs?**

Ans1. onStop

Ans2. onCreate

Ans3. onRestart

Ans4. onResume

Correct Answer : 4

**57. Which of the following is used to make data in the database available to other applications for access and manipulations?**

Ans1. Activity

Ans2. Service

Ans3. Intent

Ans4. ContentProvider

Correct Answer : 4

**58. \_\_\_\_\_ method of SQLiteOpenHelper is called when database is created**

Ans1. create()

Ans2. onCreate()

Ans3. onUpdate()

Ans4. onStartup()

Correct Answer : 2

**59. \_\_\_\_\_ method returns list of all files from internal storage?**

Ans1. getFilesDir()

Ans2. getDir()

Ans3. fileList()

Ans4. listFiles()

Correct Answer : 3

**60. \_\_\_\_\_ method of AsyncTask is used to perform long running operation**

Ans1. doProcess()

Ans2. onProcess()

Ans3. doInBackground()

Ans4. onBackground()

Correct Answer : 3

**61. Which of the following viewgroups is considered if its contents are to be scrollable?**

Ans1. FrameLayout

Ans2. LinearLayout

Ans3. RelativeLayout

Ans4. ListView

Correct Answer : 4

**62. Which of the following is correct order for life cycle events of an Activity?**

Ans1. onCreate, onStop, onRestart

Ans2. onCreate, onStart, onResume

Ans3. onCreate, onStart, onPause

Ans4. onCreate, onPause, onResume

Correct Answer : 2

**63. \_\_\_\_\_ method is used to get data from a content provider**

Ans1. ContentResolver.query()

Ans2. ContentProvider.query()

Ans3. ContentProvider.managedQuery()

Ans4. ContentResolver.managedQuery()

Correct Answer : 1

**64. Which of the following is NOT a characteristic of Content Provider?**

Ans1. It is associated data

Ans2. It is accessed using URI

Ans3. It supports query, insert, delete and update operations

Ans4. It runs in a separate thread

Correct Answer : 4

**65. Which attribute is used to ensure a view occupies the rest of the area in LinearLayout**

Ans1. layout\_fill

Ans2. layout\_span

Ans3. layout\_weight

Ans4. layout\_anchor

Correct Answer : 3

\_\_\_\_\_ is used in case task is very lengthy or need to be done regularly.

- A. AndroidManifest.xml
- B. Window
- C. Service**
- D. Ctrl + Shift + O

An Android application is a collection of tasks, each of which is called a \_\_\_\_\_

- A. Uniform Resource Identifier
- B. versionCode
- C. Activity**
- D. versionName

How to access string resource in layout file ?

- A. manifest
- B. @string/mystr1**
- C. False
- D. default.properties

Specify the directory name where String resource are stored

- A. versionCode
- B. /res/values**
- C. True
- D. Android Development Tools

\_\_\_\_\_ file defines your application's capabilities and permissions and how it runs.

- A. Uniform Resource Identifier
- B. AndroidManifest.xml**
- C. /res/values
- D. onResume()

The Android operating system keeps track of all Activity objects running by placing them on an \_\_\_\_\_

- A. Activity Stack**
- B. Context
- C. getApplicationContext()
- D. True

How to access image in layout file ?

- A. versionCode
- B. True
- C. @drawable/image1**
- D. Android Development Tools

If your projects shows error while trying to run, so you want to clean your project, which menu you will use ?

- A. Project**
- B. Android Debug Bridge
- C. Service
- D. versionCode

\_\_\_\_\_ file defines your application's build target and other build system options, as required.

- A. versionName
- B. True
- C. versionName**

**D. default.properties**

**How to access images in .java file (source file) ?**

A. Service

**B. R.drawable.image1**

C. @drawable/image1

D. onPause()

**Full form of ADT is \_\_\_\_\_**

A. R.drawable.image1

B. Android Debug Bridge

C. Open Handset Alliance

**D. Android Development Tools**

**When an Activity first starts, the \_\_\_\_\_ method is called**

A. onCreate()

B. Uniform Resource Identifier

C. manifest

D. Uniform Resource Identifier

**Full form of DDMS is \_\_\_\_\_**

A. versionName

**B. Dalvik Debug Monitor Server**

C. Uniform Resource Identifier

D. /res/values

**\_\_\_\_\_ is used to Retrieve application context.**

A. True

B. Uniform Resource Identifier

C. manifest

D. **getApplicationContext()**

Passing Additional Information Using Intents which method is used ?

A. **versionName**

B. **AndroidManifest.xml**

C. **putExtra**

D. **Open Handset Alliance**

Action bar can be associated to

A. **Only activities**

B. **Only fragments**

C. **Both activities and fragments**

D. **None of the above**

Fragment is not a part of activity

A. **True**

B. **False**

View pager is used for

A. **Swiping Fragments**

B. **Swiping Activities**

C. **Paging down list items**

D. **View pager is not supported by android SDK**

In nine patch image, corners are

A. **Scalable**

B. **Not scalable**

C. **Can be either**

D. **None of the above**

Which one of the following is not included in API level 8 or lesser?

A. **Spinner**

B. **Fragment**

C. **List View**

D. **Progress Bar**

Select depreciated components from the list below

A. **Absolute Layout**

- B. Media Controller
- C. Image Button
- D. Grid Layout
- E. Horizontal Scroll View

Is this possible to add views in a layout dynamically?

- A. Yes**
- B. No

Nesting in layouts is not supported in android

- A. True
- B. False**

Rating bar can be resize

- A. Yes
- B. No**

Which of the following does not belong to transitions?

- A. ViewSwitcher
- B. ViewFlipper
- C. ViewAnimator
- D. ViewSlider**

Android is based on which kernel?

Options

- Linux kernel
- Windows kernel
- MAC kernel
- Hybrid Kernel

CORRECT ANSWER : Linux kernel

Web browser available in android is based on

Options

- Chrome
- Firefox
- Open-source Webkit
- Opera

CORRECT ANSWER : Open-source Webkit

Android doesn't support which format.

Options

- MP4
- MPEG
- AVI

- MIDI

CORRECT ANSWER : AVI

Android supports which features.

Options

- Multitasking
- Bluetooth
- Video calling
- All of the above



CORRECT ANSWER : All of the above

Android is based on which language.

Options

- C
- C++
- VC++
- Java



CORRECT ANSWER : Java

If Fragment and activity is running and activity is destroyed, what is the effect on the fragment?

Options

- Fragment is destroyed
- Fragment runs as it is
- Fragment goes in idle state
- none of these



CORRECT ANSWER : Fragment is destroyed

Which are the states in service life cycle? 1)starting 2)paused 3)running 4)destroyed 5) wait 6)yield

Options

- 1,2,3,5
- 1,3,4
- 2,5,6
- none of these

CORRECT ANSWER : 1,3,4

In \_\_\_\_\_, sender specifies type of receiver.

Options

- Implicit intent
- Explicit intent
- a and b
- none of these



CORRECT ANSWER : Implicit intent

In pause state

Options

- activity not in focus, but visible on screen
- activity not in focus, not visible on screen
- activity is focus also visible on screen
- activity is focus ,not visible on screen

CORRECT ANSWER : activity not in focus, but visible on screen

\_\_\_\_\_ database is automatically supplied to you by Android.

Options

- Apache
- Oracle
- SQLite
- MySql

CORRECT ANSWER : SQLite

For creating user interface in Android, you have to use

Options

- Eclipse
- java and XML
- java and SQL
- Java and PI/sql

CORRECT ANSWER : java and XML

Which of the following services are provided by Android operating system?

1)location

2)sensor reading

3)WiFi

4)cloud computing

Options

- 1,2,3 and more
- 2,3,4and more
- All of the above
- None of these

CORRECT ANSWER : 1,2,3 and more

\_\_\_\_\_makes appropriate list of application data for the other applications

Options

- service provider

- content provider
- application provider
- resource

CORRECT ANSWER : content provider

Which of the following type(s) of notification is/are available in Android? 1) Toast notification  
2) Status bar notification 3) Dialog notification 4) alert notification

Options

- 1,2,3
- 2,3,4
- 1,3,4
- none of these

CORRECT ANSWER : 1,2,3

Android provides a few standard themes, listed in \_\_\_\_\_

Options

- R.style
- X.style
- manifest.XML
- application

CORRECT ANSWER : R.style

Is it possible to write Android code using c/c++?

Options

- yes
- no

CORRECT ANSWER : yes

For writing Android code using c/c++ , you need to use \_\_\_\_\_

Options

- SDK
- JDK
- NDK
- MDK

CORRECT ANSWER : NDK

Using a content provider, which of the following operations are able to perform? 1) create 2) read  
3) update 4) delete

Options

- 1,2,3
- 2,3,4
- all of the above

- none of these

CORRECT ANSWER : all of the above

To update contents of content provider using cursor and commit you need to call \_\_\_\_\_

**Options**

- commitUpdates()
- updates()
- commit()
- none of these

CORRECT ANSWER : commitUpdates()

To insert data into a content provider, you need to use \_\_\_\_\_ 1) insert() 2) bulkInsert() 3) getContentProvider() 4) update()

**Options**

- 1 and 2
- 3 and 4
- all of the above
- none of these

CORRECT ANSWER : 1 and 2

What do you mean by BLOB?

**Options**

- Bytes less object
- Binary large objects
- Binary low object
- bit low object

CORRECT ANSWER : Binary large objects

Android uses both the content, \_\_\_\_\_ and the \_\_\_\_\_ type as ways to identify content on the device.

**Options**

- Uri, MIME
- MIME, HTTP
- Uri, HTTP
- uri, FTP

CORRECT ANSWER : Uri, MIME

Requesting the use of other applications' data or services requires the user-permission element to be added to your \_\_\_\_\_ file

**Options**

- Manifest.xml
- Android.xml

- AndroidManifest.xml
- none of these

CORRECT ANSWER : AndroidManifest.xml

In Android, you can raise notifications via \_\_\_\_\_

**Options**

- Notification
- NotificationManager
- note
- Manager

CORRECT ANSWER : NotificationManager

The NotificationManager is a \_\_\_\_\_

**Options**

- system service
- user service
- interactive service
- none of these

CORRECT ANSWER : system service

Which is the base class for all android classes?

**Options**

- Object
- Class
- Android
- none of these

CORRECT ANSWER : Object

Which of the following resources can you use directly from available resources?

**Options**

- style
- styleable
- string
- raw

CORRECT ANSWER : style

**Which of the following file describes the application being built and what components – activities, services, etc. – are being supplied by that application?**

**Options**

- build.xml
- AndroidManifest.xml
- Manifest.xml
- Android.xml

**CORRECT ANSWER : AndroidManifest.xml**

**Which of the following scripts is used for building the application and installing it on the device?**

**Options**

- build.xml
- AndroidManifest.xml
- Manifest.xml
- Android.xml

**CORRECT ANSWER : build.xml**

**Which of the following holds Java source code for the application?**

**Options**

- res/
- assets/
- src/
- bin/

**CORRECT ANSWER : src/**

**"res/" holds**

**Options**

- resources
- Java source code
- application
- static files

**CORRECT ANSWER : resources**

**"assets/" holds**

**Options**

- resources
- Java source code
- application
- static files

**CORRECT ANSWER : static files**

**For building Android application we need**

**Options**

- JDK
- SDK
- ADK
- MDK

**CORRECT ANSWER : SDK**

**The simplest widget is the label, referred to in Android as a \_\_\_\_\_**

**Options**

- TextView
- grid view
- lableview
- none of these

**CORRECT ANSWER : TextView**

**Which of the following widgets helps you to embed images in your activities?**

**Options**

- ImageView
- ImageButton
- both a and b
- none of these

**CORRECT ANSWER : both a and b**

**\_\_\_\_\_ field provides automatic spelling assistance.**

**Options**

- Android:autoText
- Android:capitalize
- Android:digits
- Android:singleLine

**CORRECT ANSWER : Android:autoText**

**\_\_\_\_\_ field automatically capitalizes the first letter of entered text.**

**Options**

- Android:autoText
- Android:capitalize
- Android:digits
- Android:singleLine

CORRECT ANSWER : Android:capitalize

\_\_\_\_\_ field accepts only certain digits

**Options**

- Android:autoText
- Android:capitalize
- Android:digits
- Android:singleLine

CORRECT ANSWER : Android:digits

\_\_\_\_\_ field is used for single-line input or multiple-line input.

**Options**

- Android:autoText
- Android:capitalize
- Android:digits
- Android:singleLine

CORRECT ANSWER : Android:singleLine

Which of the following methods do we use to get the root of the tree?

**Options**

- findViewById()
- getParentOfType()
- getRootView()
- getParent()

CORRECT ANSWER : getRootView()

If you want to increase the whitespace between widgets, you will need to use the \_\_\_\_\_  
property

**Options**

- Android:padding
- Android:digits
- Android:capitalize
- Android:autoText

CORRECT ANSWER : Android:padding

Android:layout\_alignParentTop property takes a simple \_\_\_\_\_ value

**Options**

- integer

- character
- float
- Boolean

CORRECT ANSWER : Boolean

**APK package contains \_\_\_\_\_ files.**

**Options**

- .dex
- .XML
- .DOC
- .Xls

CORRECT ANSWER : .dex

**What do you mean by .dex?**

**Options**

- Dalvik expansion
- Dalvik extension
- Dalvik executables
- David executables

CORRECT ANSWER : Dalvik executables

**Which exception is thrown when a given package, application, or component name can not be found?**

**Options**

- PackageManager.NameNotFoundException
- Resources.NotFoundException
- ParseException
- FormatException

CORRECT ANSWER : PackageManager.NameNotFoundException

**For receiving an instance of Menu, we have to use**

**Options**

- setup()
- setIndicator()
- onCreate()
- onCreateOptionsMenu()

CORRECT ANSWER : onCreateOptionsMenu()



This example will take you through simple steps to show how to create your own Android Service. Follow the following steps to modify the Android application we created in *Hello World Example* chapter:

Step	Description
------	-------------

- |   |   |
|---|---|
| 1 | You will use Android Studio IDE to create an Android application and name it as <i>My Application</i> under a package <i>com.example.My Application</i> as explained in the <i>Hello World Example</i> chapter. |
| 2 | Modify main activity file <i>MainActivity.java</i> to add <i>startService()</i> and <i>stopService()</i> methods.   |
| 3 | Create a new java file <i>MyService.java</i> under the package <i>com.example.My Application</i> . This file will have implementation of Android service related methods.                                       |
| 4 | Define your service in <i>AndroidManifest.xml</i> file using <code>&lt;service.../&gt;</code> tag. An application can have one or more services without any restrictions.                                       |
| 5 | Modify the default content of <i>res/layout/activity_main.xml</i> file to include two buttons in linear layout.   |
| 6 | No need to change any constants in <i>res/values/strings.xml</i> file. Android studio take care of string values  |
| 7 | Run the application to launch Android emulator and verify the result of the changes done in the application.  |

Following is the content of the modified main activity file **src/com.example.My Application/MainActivity.java**. This file can include each of the fundamental life cycle methods. We have added *startService()* and *stopService()* methods to start and stop the service.

Sample code for Service

```
package com.example.My Application;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Intent;
import android.view.View;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    // Method to start the service
    public void startService(View view) {
        startService(new Intent(getApplicationContext(), MyService.class));
    }

    // Method to stop the service
    public void stopService(View view) {
```

```

        stopService(new Intent(getBaseContext(), MyService.class));
    }
}

```

Following is the content of **src/com.example.My Application/MyService.java**. This file can have implementation of one or more methods associated with Service based on requirements. For now we are going to implement only two methods *onStartCommand()* and *onDestroy()* -

```

package com.example.My Application;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service {

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // Let it continue running until it is stopped.
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
}

```

}

Following will be the modified content of *AndroidManifest.xml* file. Here we have added `<service.../>` tag to include our service:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.MyApplication"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="13"
        android:targetSdkVersion="22" />

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

        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>

        </activity>

        <service android:name=".MyService" />

    </application>
</manifest>
```

Following will be the content of **res/layout/activity\_main.xml** file to include two buttons:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
```

```
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Example of services"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />
```

```
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tutorials point "
        android:textColor="#ff87ff09"
        android:textSize="30dp"
        android:layout_above="@+id/imageButton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="40dp" />
```

```
    <ImageButton
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:src="@drawable/abc"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button2"
    android:text="Start Services"
    android:onClick="startService"
    android:layout_below="@+id/imageButton"
    android:layout_centerHorizontal="true" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Stop Services"
    android:id="@+id/button"
    android:onClick="stopService"
    android:layout_below="@+id/button2"
    android:layout_alignLeft="@+id/button2"
    android:layout_alignStart="@+id/button2"
    android:layout_alignRight="@+id/button2"
    android:layout_alignEnd="@+id/button2" />
```

```
</RelativeLayout>
```

**Broadcast Receivers** simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action

There are following two important steps to make BroadcastReceiver works for the system broadcasted intents –

- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

There is one additional steps in case you are going to implement your custom intents then you will have to create and broadcast those intents.

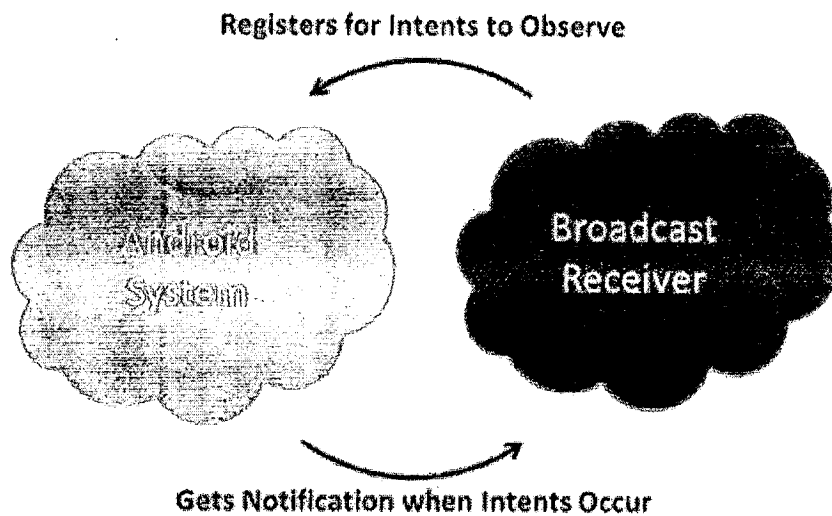
## Creating the Broadcast Receiver

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. Consider we are going to register *MyReceiver* for system generated event `ACTION_BOOT_COMPLETED` which is fired by the system once the Android system has completed the boot process.



### **BROADCAST-RECEIVER**

```
<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>
```

Now whenever your Android device gets booted, it will be intercepted by `BroadcastReceiver MyReceiver` and implemented logic inside `onReceive()` will be executed.

There are several system generated events defined as final static fields in the **Intent** class. The following table lists a few important system events.



Event Constant	Description
<code>android.intent.action.BATTERY_CHANGED</code>	Sticky broadcast containing the charging state, level, and other information about the battery.
<code>android.intent.action.BATTERY_LOW</code>	Indicates low battery condition on the device.
<code>android.intent.action.BATTERY_OKAY</code>	Indicates the battery is now okay after being low.
<code>android.intent.action.BOOT_COMPLETED</code>	This is broadcast once, after the system has finished booting.
<code>android.intent.action.BUG_REPORT</code>	Show activity for reporting a bug.
<code>android.intent.action.CALL</code>	Perform a call to someone specified by the data.
<code>android.intent.action.CALL_BUTTON</code>	The user pressed the "call" button to go to the dialer or other appropriate UI for placing a call.
<code>android.intent.action.DATE_CHANGED</code>	The date has changed.
<code>android.intent.action.REBOOT</code>	Have the device reboot.

## Broadcasting Custom Intents

If you want your application itself should generate and send custom intents then you will have to create and send those intents by using the `sendBroadcast()` method inside your activity class. If you use

the *sendStickyBroadcast(Intent)* method, the Intent is **sticky**, meaning the *Intent* you are sending stays around after the broadcast is complete.

```
public void broadcastIntent(View view)
{
    Intent intent = new Intent();
    intent.setAction("com.tutorialspoint.CUSTOM_INTENT");
    sendBroadcast(intent);
}
```

This intent *com.tutorialspoint.CUSTOM\_INTENT* can also be registered in similar way as we have registered system generated intent.

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

        <intent-filter>
            <action android:name="com.tutorialspoint.CUSTOM_INTENT">
            </action>
        </intent-filter>

    </receiver>
</application>
```

## Example

This example will explain you how to create *BroadcastReceiver* to intercept custom intent. Once you are familiar with custom intent, then you can program your application to intercept system generated intents. So let's follow the following steps to modify the Android application we created in *Hello World Example* chapter –

## Step    Description

- 1      You will use Android studio to create an Android application and name it as *My Application* under a package *com.example.My Application* as explained in the *Hello World Example* chapter.
- 2      Modify main activity file *MainActivity.java* to add *broadcastIntent()* method.
- 3      Create a new java file called *MyReceiver.java* under the package *com.example.My Application* to define a *BroadcastReceiver*.
- 4      An application can handle one or more custom and system intents without any restrictions. Every intent you want to intercept must be registered in your *AndroidManifest.xml* file using `<receiver.../>` tag
- 5      Modify the default content of *res/layout/activity\_main.xml* file to include a button to broadcast intent.
- 6      No need to modify the string file, Android studio take care of *string.xml* file.
- 7      Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.My Application/MainActivity.java**. This file can include each of the fundamental life cycle methods. We have added *broadcastIntent()* method to broadcast a custom intent.

```
package com.example.My Application;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Intent;
import android.view.View;
```

```

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    // broadcast a custom intent.
    public void broadcastIntent(View view){
        Intent intent = new Intent();
        intent.setAction("com.tutorialspoint.CUSTOM_INTENT");
        sendBroadcast(intent);
    }
}

```

Following is the content of **src/com.example.My Application/MyReceiver.java**:

```

package com.example.My Application;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {
    @Override

```

```

    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
    }
}

```

Following will be the modified content of *AndroidManifest.xml* file. Here we have added `<service.../>` tag to include our service:

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.My Application"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="22" />

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

        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>

        </activity>

        <receiver android:name="MyReceiver">

```

```

<intent-filter>
    <action android:name="com.tutorialspoint.CUSTOM_INTENT">
    </action>
</intent-filter>

```

```

</receiver>

```

```

</application>

```

```

</manifest>

```

Following will be the content of **res/layout/activity\_main.xml** file to include a button to broadcast our custom intent –

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

```

```

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Example of Broadcast"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />

```

```

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"

```

```

        android:layout_height="wrap_content"
        android:text="Tutorials point "
        android:textColor="#ff87ff09"
        android:textSize="30dp"
        android:layout_above="@+id/imageButton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="40dp" />

```

```
<ImageButton
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:src="@drawable/abc"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

```

```
<Button
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:text="Broadcast Intent"
        android:onClick="broadcastIntent"
        android:layout_below="@+id/imageButton"
        android:layout_centerHorizontal="true" />

```

```
</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants -

```
<resources>
```

```

    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">My Application</string>

```

```
</resources>
```

A **Fragment** is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity**.

Following are important points about fragment –

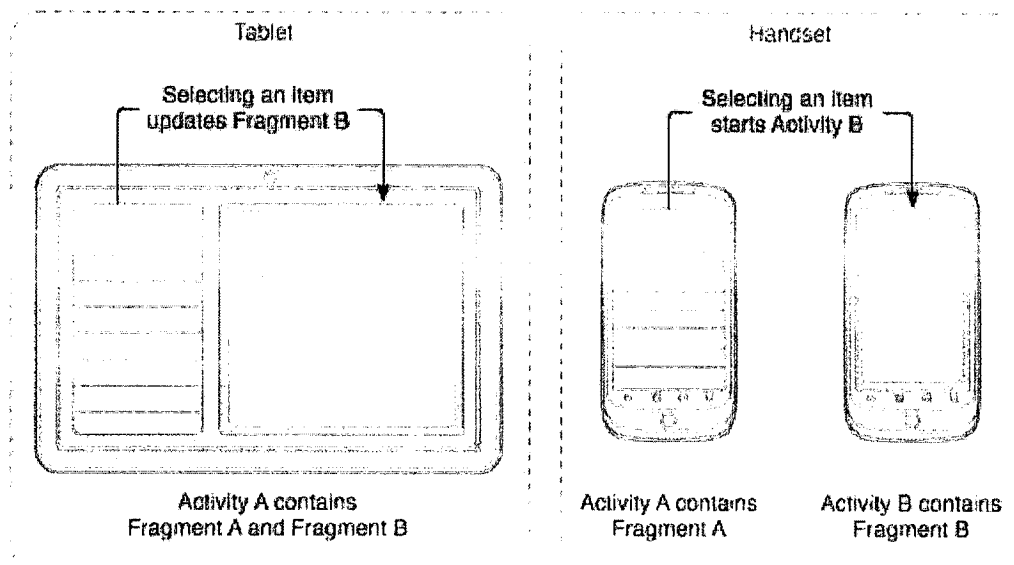
- A fragment has its own layout and its own behaviour with its own life cycle callbacks.
- You can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multi-plane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
- A fragment can implement a behaviour that has no user interface component.
- Fragments were added to the Android API in Honeycomb version of Android which API version 11.

You create fragments by extending **Fragment** class and You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a **<fragment>** element.

Prior to fragment introduction, we had a limitation because we can show only a single activity on the screen at one given point in time. So we were not able to divide device screen and control different parts separately. But with the introduction of fragment we got more flexibility and removed the limitation of having a single activity on the screen at a time. Now we can have a single activity but each activity can comprise of multiple fragments which will have their own layout, events and complete life cycle.

Following is a typical example of how two UI modules defined by fragments can be combined into one activity for a tablet design, but separated for a handset design.

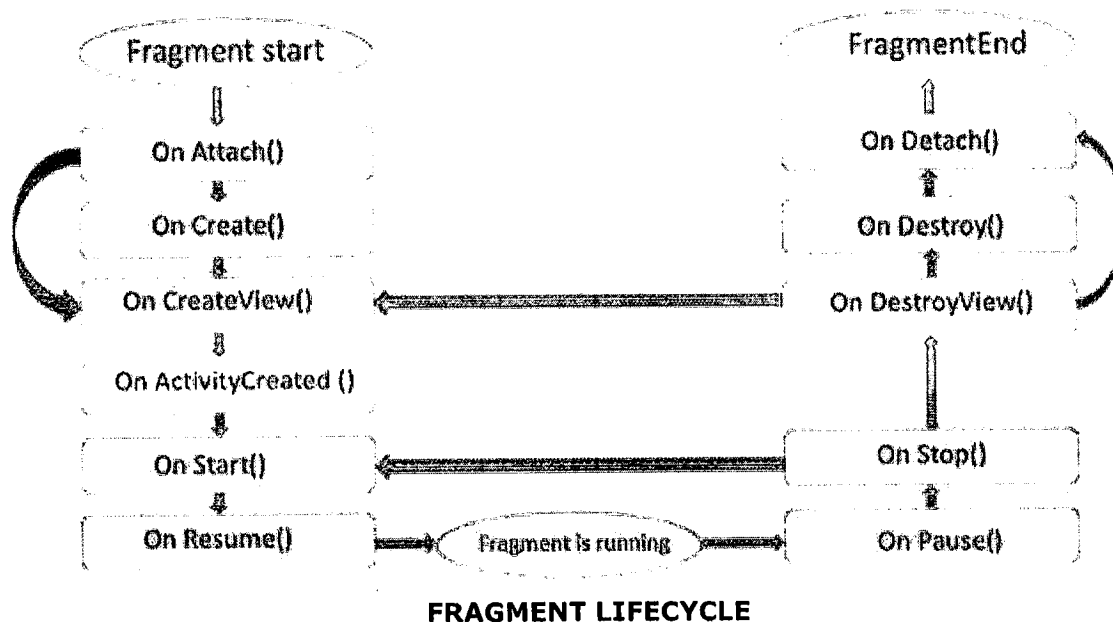




The application can embed two fragments in Activity A, when running on a tablet-sized device. However, on a handset-sized screen, there's not enough room for both fragments, so Activity A includes only the fragment for the list of articles, and when the user selects an article, it starts Activity B, which includes the second fragment to read the article.

## Fragment Life Cycle

Android fragments have their own life cycle very similar to an android activity. This section briefs different stages of its life cycle.



Here is the list of methods which you can to override in your fragment class

- **onAttach()** The fragment instance is associated with an activity instance. The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.
- **onCreate()** The system calls this method when creating the fragment. You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.
- **onCreateView()** The system calls this callback when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a **View** component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.
- **onActivityCreated()** The `onActivityCreated()` is called after the `onCreateView()` method when the host activity is created. Activity and fragment instance have been created as well as the view hierarchy of the activity. At this point, view can be accessed with the `findViewById()` method. example. In this method you can instantiate objects which require a Context object

- **onStart()** The onStart() method is called once the fragment gets visible.
- **onResume()** Fragment becomes active.
- **onPause()** The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.
- **onStop()** Fragment going to be stopped by calling onStop()
- **onDestroyView()** Fragment view will destroy after call this method
- **onDestroy()** onDestroy() called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

## How to use Fragments?

This involves number of simple steps to create Fragments.

- First of all decide how many fragments you want to use in an activity. For example let's we want to use two fragments to handle landscape and portrait modes of the device.
- Next based on number of fragments, create classes which will extend the *Fragment* class. The *Fragment* class has above mentioned callback functions. You can override any of the functions based on your requirements.
- Corresponding to each fragment, you will need to create layout files in XML file. These files will have layout for the defined fragments.
- Finally modify activity file to define the actual logic of replacing fragments based on your requirement.

## 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.

## Single Frame Fragment

Single frame fragment is designed for small screen devices such as hand hold devices(mobiles) and it should be above android 3.0 version.

## Example

This example will explain you how to create your own *Fragments*. Here we will create two fragments and one of them will be used when device is in landscape mode and another fragment will be used in case of portrait mode. So let's follow the following steps to similar to what we followed while creating *Hello World Example* –

Step	Description
1	You will use Android Studio IDE to create an Android application and name it as <i>MyFragments</i> under a package <i>com.example.myfragments</i> , with blank Activity.
2	Modify main activity file <i>MainActivity.java</i> as shown below in the code. Here we will check orientation of the device and accordingly we will switch between different fragments.
3	Create a two java files <i>PM_Fragment.java</i> and <i>LM_Fragment.java</i> under the package <i>com.example.myfragments</i> to define your fragments and associated methods.
4	Create layouts files <i>res/layout/lm_fragment.xml</i> and <i>res/layout/pm_fragment.xml</i> and define your layouts for both the fragments.
5	Modify the default content of <i>res/layout/activity_main.xml</i> file to include both the fragments.
6	Define required constants in <i>res/values/strings.xml</i> file
7	Run the application to launch Android emulator and verify the result of the

changes done in the application.

Following is the content of the modified main activity file **src/com.example.mycontentprovider/MainActivity.java** –

```
package com.example.myfragments;

import android.os.Bundle;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.res.Configuration;
import android.view.WindowManager;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Configuration config = getResources().getConfiguration();

        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
            fragmentManager.beginTransaction();

        /**
         * Check the device orientation and act accordingly
         */
        if (config.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            /**
             * Landscape mode of the device
             */
        }
    }
}
```

```

        LM_Fragment ls_fragment = new LM_Fragment();
        fragmentTransaction.replace(android.R.id.content, ls_fragment);
    }else{
        /**
         * Portrait mode of the device
         */
        PM_Fragment pm_fragment = new PM_Fragment();
        fragmentTransaction.replace(android.R.id.content, pm_fragment);
    }
    fragmentTransaction.commit();
}
}

```

Create two fragment files **LM\_Fragment.java** and **PM\_Fragment.java** under *com.example.mycontentprovider* package.

Following is the content of **LM\_Fragment.java** file –

```

package com.example.myfragments;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class LM_Fragment extends Fragment{
    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
        /**
         * Inflate the layout for this fragment

```

```

        */
        return inflater.inflate(
            R.layout.lm_fragment, container, false);
    }
}

```

Following is the content of **PM\_Fragment.java** file –

```

package com.example.myfragments;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PM_Fragment extends Fragment{
    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
        /**
         * Inflate the layout for this fragment
         */
        return inflater.inflate(
            R.layout.pm_fragment, container, false);
    }
}

```

Create two layout files **lm\_fragments.xml** and **pm\_fragment.xml** under *res/layout* directory

Following is the content of **lm\_fragments.xml** file –



```

<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#7bae16">

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/landscape_message"
            android:textColor="#000000"
            android:textSize="20px" />

        <!-- More GUI components go here -->

    </LinearLayout>

```

Following is the content of **pm\_fragment.xml** file –

```

<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#666666">

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/portrait_message"
            android:textColor="#000000"

```

```
        android:textSize="20px" />
```

```
<!-- More GUI components go here -->
```

```
</LinearLayout>
```

Following will be the content of **res/layout/activity\_main.xml** file which includes your fragments –

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:orientation="horizontal">
```

```
    <fragment
```

```
        android:name="com.example.fragments"
```

```
        android:id="@+id/lm_fragment"
```

```
        android:layout_weight="1"
```

```
        android:layout_width="0dp"
```

```
        android:layout_height="match_parent" />
```

```
    <fragment
```

```
        android:name="com.example.fragments"
```

```
        android:id="@+id/pm_fragment"
```

```
        android:layout_weight="2"
```

```
        android:layout_width="0dp"
```

```
        android:layout_height="match_parent" />
```

```
</LinearLayout>
```

Make sure you have following content of **res/values/strings.xml** file –

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
```

```
  <string name="app_name">MyFragments</string>
```

```
  <string name="action_settings">Settings</string>
```

```
  <string name="hello_world">Hello world!</string>
```

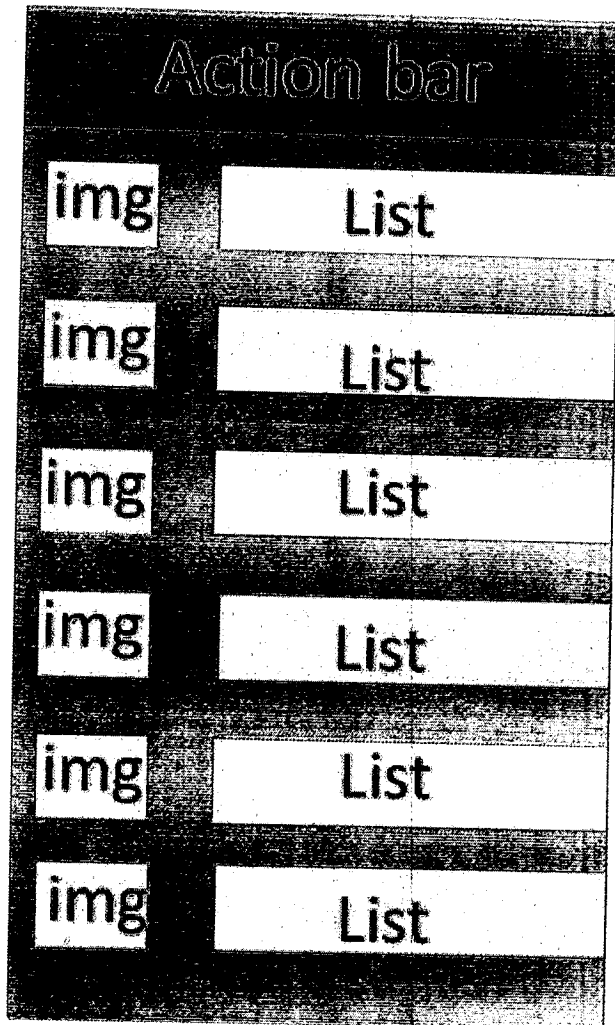
```
  <string name="landscape_message">This is Landscape mode fragment</string>
```

```
  <string name="portrait_message">This is Portrait mode fragment<</string>
```

```
</resources>
```

Static library support version of the framework's ListFragment. Used to write apps that run on platforms prior to Android 3.0. When running on Android 3.0 or above, this implementation is still used.

**The basic implementation of list fragment is for creating list of items in fragments**



**LIST IN FRAGMENTS**

## Example

This example will explain you how to create your own list fragment based on arrayAdapter. So let's follow the following steps to similar to what we followed while creating Hello World Example:

## Step Description

- 1 You will use Android Studio to create an Android application and name it as *SimpleListFragment* under a package *com.example.listfragmentdemo*, with blank Activity.
- 2 Modify the string file, which has placed at *res/values/string.xml* to add new string constants
- 3 Create a layout called *list\_fragment.xml* under the directory *res/layout* to define your list fragments. and add fragment tag (<fragment>) to your *activity\_main.xml*
- 4 Create a *myListFragment.java*, which is placed at *java/myListFragment.java* and it contained *onCreateView()*, *onActivityCreated()* and *OnItemClickListener()*
- 7 Run the application to launch Android emulator and verify the result of the changes done in the application.

Before start coding i will initialize of the string constants inside *string.xml* file under *res/values* directory

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ListFragmentDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="imgdesc">imgdesc</string>

    <string-array name="Planets">
        <item>Sun</item>
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

```
<item>Jupiter</item>
<item>Saturn</item>
<item>Uranus</item>
<item>Neptune</item>
</string-array>
```

```
</resources>
```

Following will be the content of **res/layout/activity\_main.xml** file. it contained linear layout and fragment tag

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <fragment
        android:id="@+id/fragment1"
        android:name="com.pavan.listfragmentdemo.MyListFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Following will be the content of **res/layout/list\_fragment.xml** file. it contained linear layout, list view and text view

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
```

```

<ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</ListView>

<TextView
    android:id="@android:id/empty"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</TextView>

</LinearLayout>

```

following will be the content of **src/main/java/myListFragment.java** file. before writing to code, need to follow few steps as shown below

- Create a class **MyListFragment** and extend it to **ListFragment** .
- Inside the **onCreateView()** method , inflate the view with above defined **list\_fragment** xml layout.
- Inside the **onActivityCreated()** method , create a arrayadapter from resource ie using **String** array **R.array.planet** which you can find inside the **string.xml** and set this adapter to listview and also set the **onItem** click Listener.
- Inside the **OnItemClickListener()** method , display a toast message with Item name which is being clicked.

```

package com.example.listfragmentdemo;

import android.annotation.SuppressLint;
import android.app.ListFragment;
import android.os.Bundle;

```

```

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Toast;

public class MyListFragment extends ListFragment implements OnItemClickListener {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.list_fragment, container, false);
        return view;
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);

        ArrayAdapter adapter = ArrayAdapter.createFromResource(getActivity(),
R.array.Planets, android.R.layout.simple_list_item_1);
        setListAdapter(adapter);
        getListView().setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(getActivity(), "Item: " + position, Toast.LENGTH_SHORT).show();
    }
}

```

Following code will be the content of MainActivity.java



```

package com.example.listfragmentdemo;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

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

```

following code will be the content of manifest.xml, which has placed at res/AndroidManifest.xml

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.listfragmentdemo"
    android:versionCode="1"
    android:versionName="1.0" >

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

        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>


```

```
</intent-filter>

</activity>

</application>
</manifest>
```

## Running the Application

Let's try to run our **SimpleListFragment** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android Studio, open one of your project's activity files and click Run  icon from the toolbar. Android installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window –

Sun

Mercury

Venus

Earth

Mars

Jupiter

Saturn

Uranus

Neptune

## What is a Transition?

Activity and Fragment transitions in Lollipop are built on top of a relatively new feature in Android called Transitions. Introduced in KitKat, the transition framework provides a convenient API for animating between different UI states in an application. The framework is built around two key concepts: scenes and transitions. A scene defines a given state of an application's UI, whereas a transition defines the animated change between two scenes.

When a scene changes, a Transition has two main responsibilities –

- Capture the state of each view in both the start and end scenes.
- Create an Animator based on the differences that will animate the views from one scene to the other.

## Example

This example will explain you how to create your custom animation with fragment transition . So let's follow the following steps to similar to what we followed while creating Hello World Example –

Step	Description
1	You will use Android Studio to create an Android application and name it as <i>fragmentcustomanimations</i> under a package <i>com.example.fragmentcustomanimations</i> , with blank Activity.
2	Modify the <i>activity_main.xml</i> , which has placed at <i>res/layout/activity_main.xml</i> to add a Text View
3	Create a layout called <i>fragment_stack.xml.xml</i> under the directory <i>res/layout</i> to define your fragment tag and button tag
4	Create a folder, which is placed at <i>res/</i> and name it as <i>animation</i> and add <i>fragment_slide_left_enter.xml</i> <i>fragment_slide_left_exit.xml</i> , <i>fragment_slide_right_exit.xml</i> and <i>fragment_slide_left_enter.xml</i>

- 7 In MainActivity.java, need to add fragment stack, fragment manager, and onCreateView()
- 8 Run the application to launch Android emulator and verify the result of the changes done in the application.

following will be the content of *res.layout/activity\_main.xml* it contained TextView

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical|center_horizontal"
    android:text="@string/hello_world"
    android:textAppearance="?android:attr/textAppearanceMedium" />
```

Following will be the content of **res/animation/fragment\_stack.xml** file. it contained frame layout and button

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <fragment
        android:id="@+id/fragment1"
        android:name="com.pavan.listfragmentdemo.MyListFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Following will be the content of **res/animation/fragment\_slide\_left\_enter.xml** file. it contained set method and object animator

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="100dp" android:valueTo="0dp"
        android:valueType="floatType"
        android:propertyName="translationX"
        android:duration="@android:integer/config_mediumAnimTime" />

    <objectAnimator
        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="0.0" android:valueTo="1.0"
        android:valueType="floatType"
        android:propertyName="alpha"
        android:duration="@android:integer/config_mediumAnimTime" />
</set>
```

following will be the content of **res/animation/fragment\_slide\_left\_exit.xml** file. it contained set and object animator tags

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="0dp" android:valueTo="-100dp"
        android:valueType="floatType"
        android:propertyName="translationX"
        android:duration="@android:integer/config_mediumAnimTime" />

    <objectAnimator
```

```

        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="1.0" android:valueTo="0.0"
        android:valueType="floatType"
        android:propertyName="alpha"
        android:duration="@android:integer/config_mediumAnimTime" />
</set>

```

Following code will be the content of **res/animation/fragment\_slide\_right\_enter.xml** file. it contained set and object animator tags

```

<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="-100dp" android:valueTo="0dp"
        android:valueType="floatType"
        android:propertyName="translationX"
        android:duration="@android:integer/config_mediumAnimTime" />

    <objectAnimator
        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="0.0" android:valueTo="1.0"
        android:valueType="floatType"
        android:propertyName="alpha"
        android:duration="@android:integer/config_mediumAnimTime" />
</set>

```

following code will be the content of **res/animation/fragment\_slide\_right\_exit.xml** file, it contained set and object animator tags

```

<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator

```

```

        android:interpolator="@android:interpolator/decelerate_quint"
        android:valueFrom="0dp" android:valueTo="100dp"
        android:valueType="floatType"
        android:propertyName="translationX"
        android:duration="@android:integer/config_mediumAnimTime" />

```

```

<objectAnimator

```

```

    android:interpolator="@android:interpolator/decelerate_quint"
    android:valueFrom="1.0" android:valueTo="0.0"
    android:valueType="floatType"
    android:propertyName="alpha"
    android:duration="@android:integer/config_mediumAnimTime" />

```

```

</set>

```

following code will be the content of **src/main/java/MainActivity.java** file. it contained button listener, stack fragment and onCreateView

```

package com.example.fragmentcustomanimations;

```

```

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.os.Bundle;

```

```

import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;

```

```

import android.widget.Button;
import android.widget.TextView;

```

```

/**

```



```
* Demonstrates the use of custom animations in a FragmentTransaction when  
* pushing and popping a stack.
```

```
*/
```

```
public class FragmentCustomAnimations extends Activity {
```

```
    int mStackLevel = 1;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.fragment_stack);
```

```
        // Watch for button clicks.
```

```
        Button button = (Button)findViewById(R.id.new_fragment);
```

```
        button.setOnClickListener(new OnClickListener() {
```

```
            public void onClick(View v) {
```

```
                addFragmentToStack();
```

```
            }
```

```
        });
```

```
        if (savedInstanceState == null) {
```

```
            // Do first time initialization -- add initial fragment.
```

```
            Fragment newFragment = CountingFragment.newInstance(mStackLevel);
```

```
            FragmentTransaction ft = getFragmentManager().beginTransaction();
```

```
            ft.add(R.id.simple_fragment, newFragment).commit();
```

```
        }
```

```
        else
```

```
        {
```

```
            mStackLevel = savedInstanceState.getInt("level");
```

```
        }
```

```
    }
```

```
    @Override
```

```

public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt("level", mStackLevel);
}

void addFragmentToStack() {
    mStackLevel++;

    // Instantiate a new fragment.
    Fragment newFragment = CountingFragment.newInstance(mStackLevel);

    // Add the fragment to the activity, pushing this transaction
    // on to the back stack.
    FragmentTransaction ft = getFragmentManager().beginTransaction();
    ft.setCustomAnimations(R.animator.fragment_slide_left_enter,
        R.animator.fragment_slide_left_exit,
        R.animator.fragment_slide_right_enter,
        R.animator.fragment_slide_right_exit);
    ft.replace(R.id.simple_fragment, newFragment);
    ft.addToBackStack(null);
    ft.commit();
}

public static class CountingFragment extends Fragment {
    int mNum;
    /**
     * Create a new instance of CountingFragment, providing "num"
     * as an argument.
     */
    static CountingFragment newInstance(int num) {
        CountingFragment f = new CountingFragment();

        // Supply num input as an argument.

```

```

        Bundle args = new Bundle();
        args.putInt("num", num);
        f.setArguments(args);
        return f;
    }

    /**
     * When creating, retrieve this instance's number from its arguments.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mNum = getArguments() != null ? getArguments().getInt("num") : 1;
    }

    /**
     * The Fragment's UI is just a simple text view showing its
     * instance number.
     */
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.hello_world, container, false);
        View tv = v.findViewById(R.id.text);
        ((TextView)tv).setText("Fragment #" + mNum);

        tv.setBackgroundDrawable(getResources().getDrawable(android.R.drawable.gallery_thumb));
    }

    return v;
}
}
}

```

following will be the content of **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.fragmentcustomanimations"
android:versionCode="1"
android:versionName="1.0" >

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
        android:name="com.example.fragmentcustomanimations.MainActivity"
        android:label="@string/app_name" >


        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

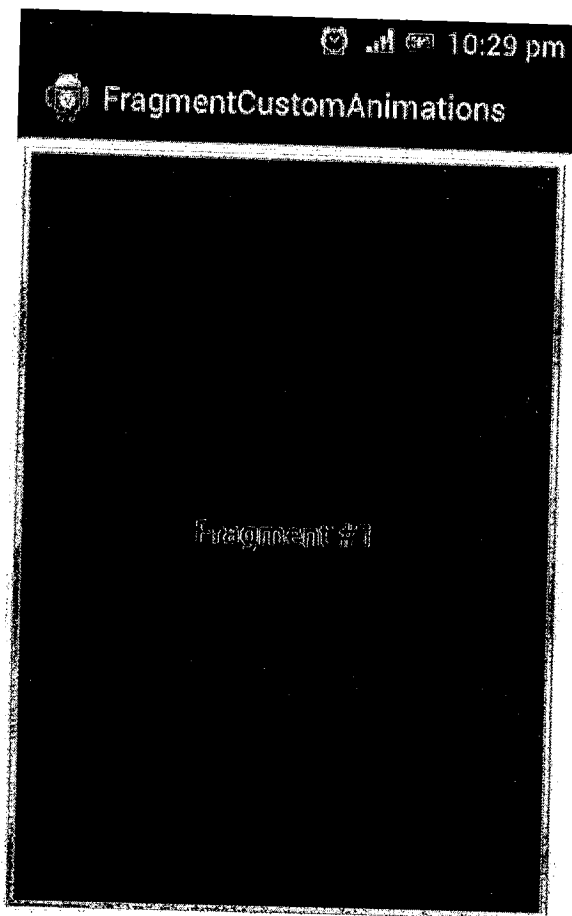
    </activity>

</application>
</manifest>

```

## Running the Application

Let's try to run our **Fragment Transitions** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android Studio, open one of your project's activity files and click Run  icon from the toolbar. Android installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window:



New Fragment

If click on new fragment, it going to be changed the first fragment to second fragment as shown below

10:29 pm

FragmentCustomAnimations

Fragment 72

New Fragment