Building blocks of Android:

there are 4 main components of android as follows:

* Activity
* Services
* Broadcast Receivers
* Content providers

**Activity:**

* An Android app contains one or more activities.
* An activity is a single screen with user Interface.
* A single Activity means a single screen contains two files java and xml file or layout file to represent user interface.
* A java file becomes an activity when it becomes sub class Activity class, it is as follows.

class MainActivity extends Activity

{

}

**Service:**

* Service is long running application component.
* It runs in the background without any user interface.
* Example for service play music, downloading data through internet etc.

**Broadcast Receivers:**

* A *broadcast receiver* is an Android component which allows you to register for system or application events.
* It also communication between OS and hardware for example whenever mobile battery goes down it notifies the user if not charged mobile will shut down.

**Content Providers:**

* An content provider is an Android component which allows to access the storage.
* Content providers is used to share the data between the applications.
* Example message app can access or share the contact numbers stored in the mobile to send the message.

7. login app

8. A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. You can declare a layout in two ways:

* **Declare UI elements in XML**. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
* **Instantiate layout elements at runtime**. Your application can create View and ViewGroup objects (and manipulate their properties) programmatically.

Types of Layouts

* Linear layout(vertical)
* Linear layout(horizontal)
* Relative layout
* Grid layout
* Table layout
* Frame layout

Linear layout(vertical):

* By default it place widgets one below the other



Linear layout(horizontal):

* By default it place widgets one after the other

Relative layout:

Relative Layout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent Relative Layout area (such as aligned to the bottom, left or center).



Grid layout:

GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid. The grid items are automatically inserted to the layout using a ListAdapter.

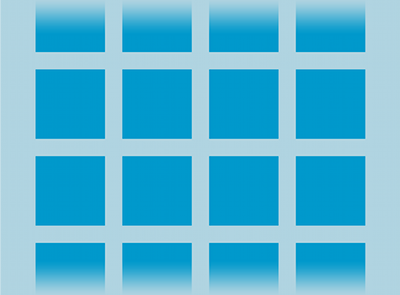


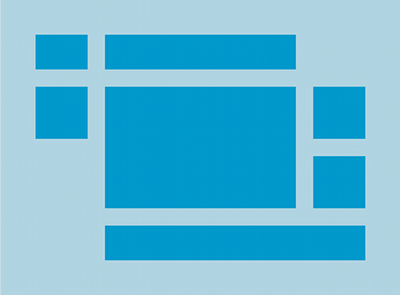
Table layout:

TableLayout is a ViewGroup that displays child View elements in rows and columns.

TableLayout positions its children into rows and columns. Table Layout containers do not display border lines for their rows, columns, or cells. The table will have as many columns as the row with the most cells. A table can leave cells empty, but cells cannot span columns, as they can in HTML.

TableRow objects are the child views of a TableLayout (each TableRow defines a single row in the table). Each row has zero or more cells, each of which is defined by any kind of other View. So, the cells of a row may be composed of a variety of View objects, like ImageView or TextView objects. A cell may also be a ViewGroup object (for example, you can nest another Table Layout as a cell).

Fig: TableLayout



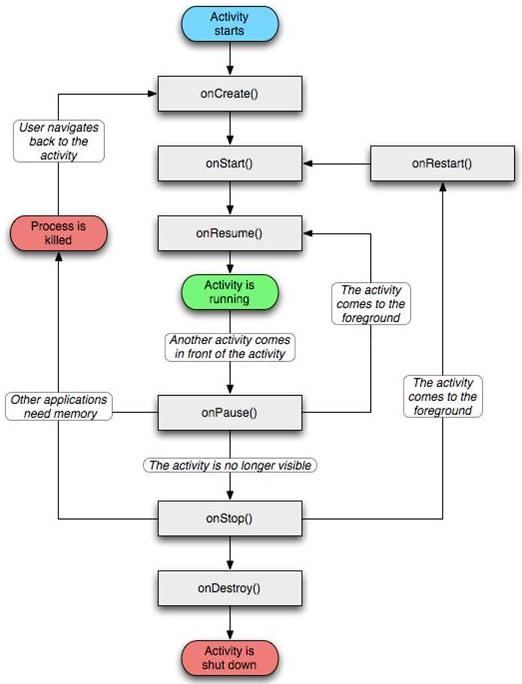
Frame Layout:

FrameLayout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other. You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the *android:layout\_gravity* attribute.

Child views are drawn in a stack, with the most recently added child on top. The size of the FrameLayout is the size of its largest child (plus padding), visible or not (if the FrameLayout's parent permits). Views that are *GONE* are used for sizing only if *setConsiderGoneChildrenWhenMeasuring()* is set to true.

**Activity & Activity Life Cycle:**

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window, the Activity instances in your app transition between different states in their lifecycle. For instance, when your activity starts for the first time, it comes to the foreground of the system and receives user focus. During this process, the Android system calls a series of lifecycle methods on the activity in which you set up the user interface and other components. If the user performs an action that starts another activity or switches to another app, the system calls another set of lifecycle methods on your activity as it moves into the background



there are several situations in which an activity transitions between different states that are illustrated in figure. 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.