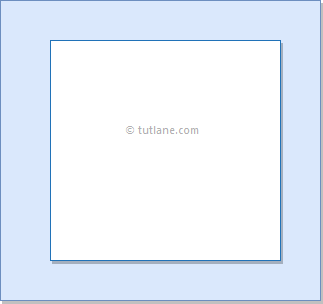
**6. Android FrameLayout with Examples**

In android, **Framelayout** is a **ViewGroup** subclass which is used to specify the position of **View** instances it contains on the top of each other to display only single **View** inside the FrameLayout.

In simple manner, we can say FrameLayout is designed to block out an area on the screen to display a single item.

Following is the pictorial representation of frame layout in android applications.



In android, **FrameLayout** will act as a placeholder on the screen and it is used to hold a single child view.

In FrameLayout, the child views are added in a stack and the most recently added child will show on the top. We can add multiple children views to FrameLayout and control their position by using gravity attributes in FrameLayout.

**Android FrameLayout Example**

Following is the example of creating a **FrameLayout** with different controls in android application.

Create a new android application using android studio and give names as **FrameLayout**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

Now open an **activity\_main.xml** file from **\res\layout** path and write the code like as shown below

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    android:orientation="vertical">  
    <ImageView  
        android:id="@+id/imgvw1"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:scaleType="centerCrop"  
        android:src="@drawable/flimg" />  
    <TextView  
        android:id="@+id/txtvw1"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="40dp"  
        android:background="#4C374A"  
        android:padding="10dp"  
        android:text="Grand Palace, Bangkok"  
        android:textColor="#FFFFFF"  
        android:textSize="20sp" />  
    <TextView  
        android:id="@+id/txtvw2"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_gravity="right|bottom"  
        android:background="#AA000000"  
        android:padding="10dp"  
        android:text="21/Aug/2017"  
        android:textColor="#FFFFFF"  
        android:textSize="18sp" />  
</FrameLayout>

If we observe above code we used **ImageView** to show the image (**flimg**) from drawable folder in framelayout. So add your image to **drawable** folder and replace **@drawable/flimg** path with your image path.

Once we are done with creation of layout, we need to load the XML layout resource from our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.framelayout** path and write the code like as shown below.

**MainActivity.java**

package com.tutlane.linearlayout;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
    }  
}

If we observe above code, we are calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main**.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



 This is how we can use frame layout in android applications based on our requirements.