**3. Android Fade In / Out Animations with Examples**

In android, **Fade In** and **Fade Out** animations are used to change the appearance and behaviour of the objects over a particular interval of time. The Fade In and Fade Out animations will provide a better look and feel for our applications.

Generally, the animations are useful when we want to notify users about the change’s happening in our app, such as new content loaded or new actions available, etc.

To create an animation effect to the objects in our android application, we need to follow below steps.

**Create XML File to Define Animation**

We need to create an xml file that defines the type of animation to perform in a new folder **anim** under res directory (**res**  **anim**  **fade\_in.xml**) with required properties. In case **anim** folder not exists in **res** directory, create a new one.

To use **Fade In** or **Fade Out** animations in our android applications, we need to define a new xml file with **<alpha>** tag like as shown below.

For **Fade In** animation, we need to increase the **alpha** value from **0** to **1** like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android" android:interpolator="@android:anim/linear\_interpolator">  
   <alpha  
        android:duration="2000"  
        android:fromAlpha="0.1"  
        android:toAlpha="1.0">  
    </alpha>  
</set>

For **Fade Out** animation, we need to decrease the **alpha** value from **1** to **0** like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"  
    android:interpolator="@android:anim/linear\_interpolator">  
    <alpha  
        android:duration="2000"  
        android:fromAlpha="1.0"  
        android:toAlpha="0.1" >  
    </alpha>  
</set>

Once we are done with creation of required animation XML files, we need to load those animation files using different properties.

**Android Load and Start the Animation**

In android, we can perform animations by using **AnimationUtils** component methods such as **loadAnimation()**. Following is the code snippet of loading and starting an animation using **loadAnimation()** and **startAnimation()**methods.

ImageView img = (ImageView)findViewById(R.id.imgvw);

Animation aniFade = AnimationUtils.loadAnimation(getApplicationContext(),R.anim.fade\_in);  
img.startAnimation(aniFade);

If we observe above code snippet, we are adding an animation to the image using **loadAnimation()** method. The second parameter in **loadAnimation()** method is the name of our animation xml file.

Here we used another method **startAnimation()** to apply the defined animation to imageview object.

Now we will see how to implement fade in and fade out animations for imageview on button click in android applications with examples.

**Android Fade In & Fade Out Animations Example**

Following is the example of implementing a fade in and fade out animations to fade in /out image on button click in android applications.

Create a new android application using android studio and give names as **FadeInOutExample**.

Once we create an application, open **activity\_main.xml** file from **\res\layout** folder path and write the code like as shown below.

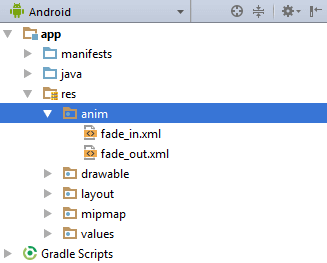
**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    android:paddingLeft="10dp"  
    android:paddingRight="10dp">  
    <ImageView android:id="@+id/imgvw"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="250dp"  
        android:src="@drawable/bangkok"/>  
    <Button  
        android:id="@+id/btnFadeIn"  
        android:layout\_below="@+id/imgvw"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Fade In"

android:layout\_marginLeft="100dp" />  
    <Button  
        android:id="@+id/btnFadeOut"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignBottom="@+id/btnFadeIn"  
        android:layout\_toRightOf="@+id/btnFadeIn"  
        android:text="Fade Out" />  
</RelativeLayout>

As discussed, we need to create an xml files to define fade in and fade out animations in new folder **anim** under **res**directory (**res**  **anim**  **fade\_in.xml**, **fade\_out.xml**) with required properties. In case **anim** folder not exists in **res**directory, create a new one.

Following is the example of creating an XML files (**fade\_in.xml**, **fade\_out.xml**) under **anim** folder to define fade in / out animation properties.



Now open **fade\_in.xml** file and write the code to set fade in animation properties like as shown below.

**fade\_in.xml**

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android" android:interpolator="@android:anim/linear\_interpolator">  
   <alpha  
        android:duration="2000"  
        android:fromAlpha="0.1"  
        android:toAlpha="1.0">  
    </alpha>  
</set>

Now open **fade\_out.xml** file and write the code to set fade out animation properties like as shown below

fade\_out.xml

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"  
    android:interpolator="@android:anim/linear\_interpolator">  
    <alpha  
        android:duration="2000"  
        android:fromAlpha="1.0"  
        android:toAlpha="0.1" >  
    </alpha>  
</set>

Now open your main activity file **MainActivity.java** from **\java\com.tutlane.fadeinoutexample** path and write the code like as shown below

**MainActivity.java**

package com.sarker.fadeinoutexample;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.view.animation.Animation;  
import android.view.animation.AnimationUtils;  
import android.widget.Button;  
import android.widget.ImageView;  
  
public class MainActivity extends AppCompatActivity {  
    private Button btnfIn;  
    private Button btnfOut;  
    private ImageView img;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        btnfIn = (Button)findViewById(R.id.btnFadeIn);  
        btnfOut = (Button)findViewById(R.id.btnFadeOut);  
        img = (ImageView)findViewById(R.id.imgvw);

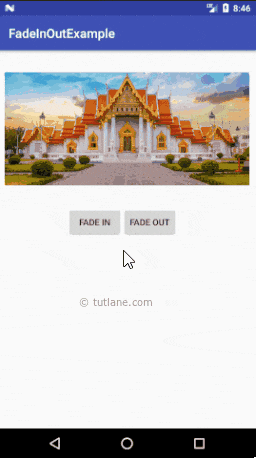
        btnfIn.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Animation animFadeIn = AnimationUtils.loadAnimation(getApplicationContext(),R.anim.fade\_in);  
                img.startAnimation(animFadeIn);  
            }  
        });

        btnfOut.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Animation animFadeOut = AnimationUtils.loadAnimation(getApplicationContext(),R.anim.fade\_out);  
                img.startAnimation(animFadeOut);  
            }  
        });  
    }  
}

If we observe above code, we are adding an animation to the image using **loadAnimation()** method and used **startAnimation()** method to apply the defined animation to imageview object.

**Output of Android Fade In / Out Animations Example**

When we run above program in android studio we will get the result like as shown below.



If we observe above result, whenever we are clicking on **Fade In** or **Fade Out** buttons, the image size varies based on our functionality.

This is how we can implement fade in and fade in animations for imageview in android applications based on our requirements.