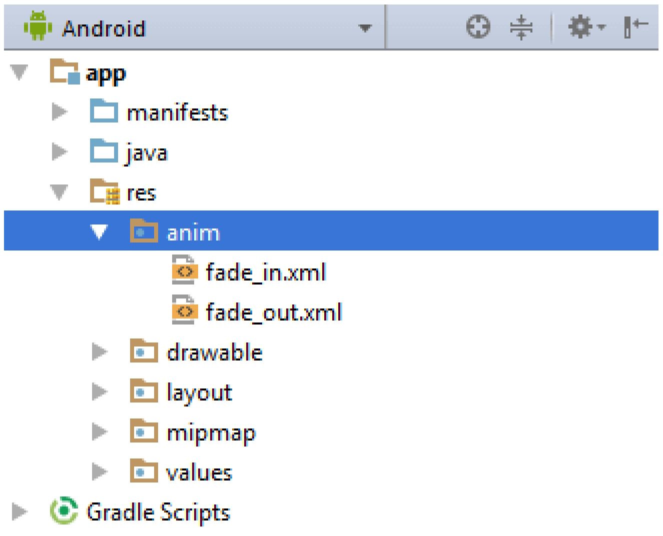
# Animation (Fade IN/Fade OUT)

## activity_main.xml

```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:in
terpolator="@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

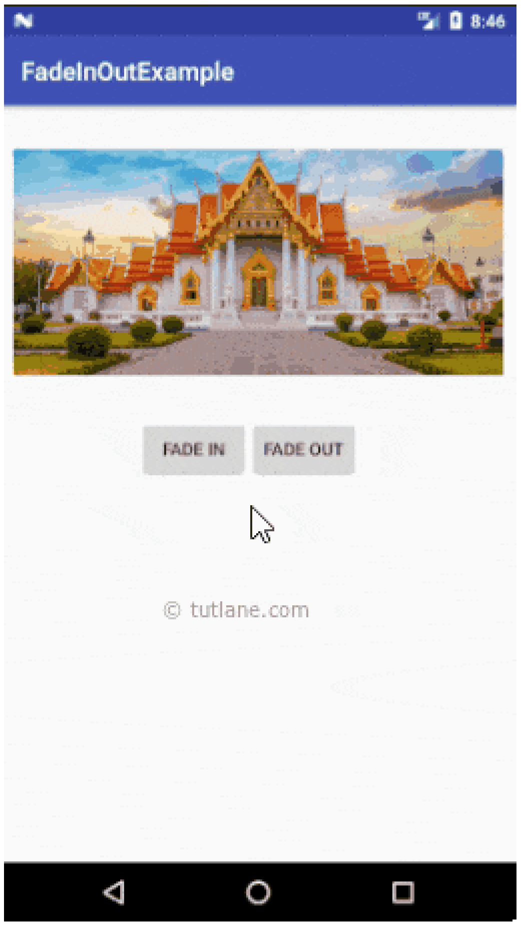
```
package com.tutlane.fadeinoutexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;

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(getApp
licationContext(),R.anim.fade_in);
                img.startAnimation(animFadeIn);
            }
        });
        btnfOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animFadeOut = AnimationUtils.loadAnimation(getAp
plicationContext(),R.anim.fade_out);
                img.startAnimation(animFadeOut);
            }
        });
    }
}
```

If you 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 you observe the 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.

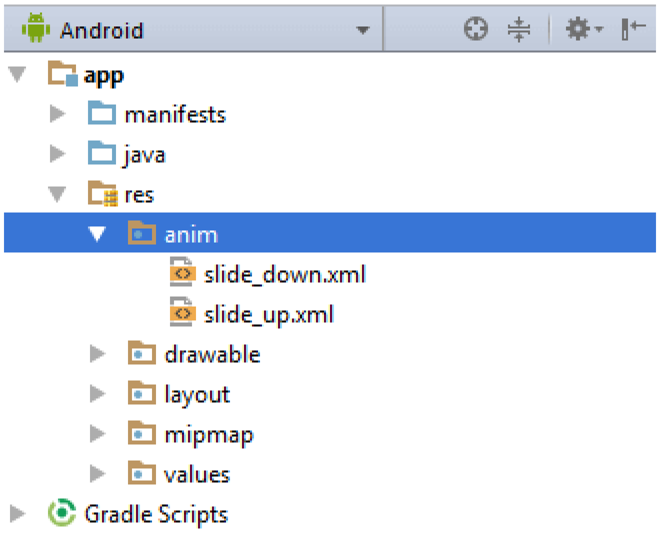
## Animation (Slide UP/Slide DOWN)

### 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/btnSlideDown"
        android:layout_below="@+id/imgvw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Slide Down" android:layout_marginLeft="100dp" />
    <Button
        android:id="@+id/btnSlideUp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnSlideDown"
        android:layout_toRightOf="@+id/btnSlideDown"
        android:text="Slide Up" />
</RelativeLayout>
```

As discussed, we need to create an xml files to define slide up and slide down animations in new folder **anim** under **res** directory (**res** → **anim** → **slide_up.xml**, **slide_down.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 (**slide_up.xml**, **slide_down.xml**) under **anim** folder to define slide up / down animation properties.



Now open **slide_up.xml** file and write the code to set slide up animation properties like as shown below.

## slide_up.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:in
terpolator="@android:anim/linear_interpolator">
    <scale
        android:duration="500"
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:toXScale="1.0"
        android:toYScale="0.0" />
</set>
```

Now open **slide_down.xml** file and write the code to set slide down animation properties like as shown below.

## slide_down.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:in
terpolator="@android:anim/linear_interpolator">
    <scale
        android:duration="500"
        android:fromXScale="1.0"
        android:fromYScale="0.0"
        android:toXScale="1.0"
        android:toYScale="1.0" />
</set>
```

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

## MainActivity.java

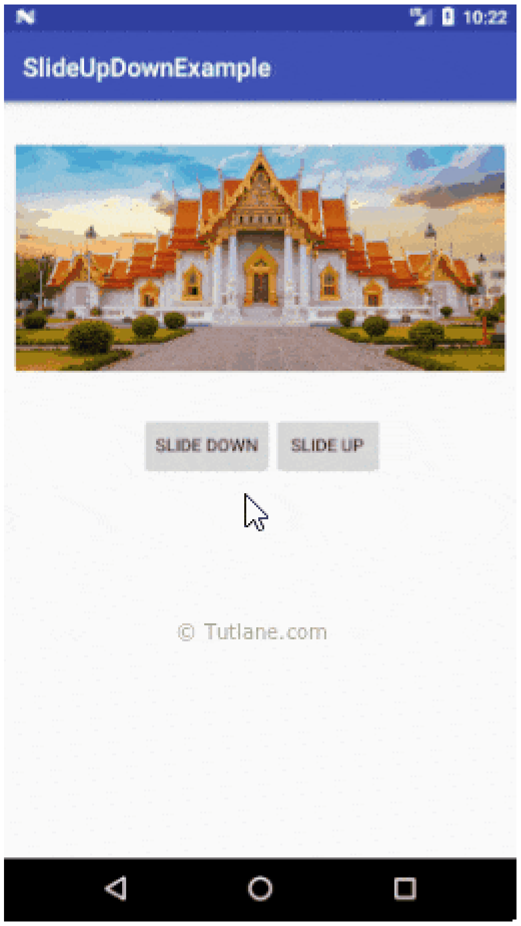
```
package com.tutlane.slideupdownexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {
    private Button btnSDown;
    private Button btnSUp;
    private ImageView img;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnSDown = (Button)findViewById(R.id.btnSlideDown);
        btnSUp = (Button)findViewById(R.id.btnSlideUp);
        img = (ImageView)findViewById(R.id.imgvw);
        btnSDown.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animSlideDown = AnimationUtils.loadAnimation(get
ApplicationContext(),R.anim.slide_down);
                img.startAnimation(animSlideDown);
            }
        });
        btnSUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animSlideUp = AnimationUtils.loadAnimation(getAp
plicationContext(),R.anim.slide_up);
                img.startAnimation(animSlideUp);
            }
        });
    }
}
```

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

## Output of Android Slide Up / Down Animation Example

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



If you observe the above result, whenever we are clicking on **Slide Up** or **Slide Down** buttons, the image size varies based on our functionality.

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

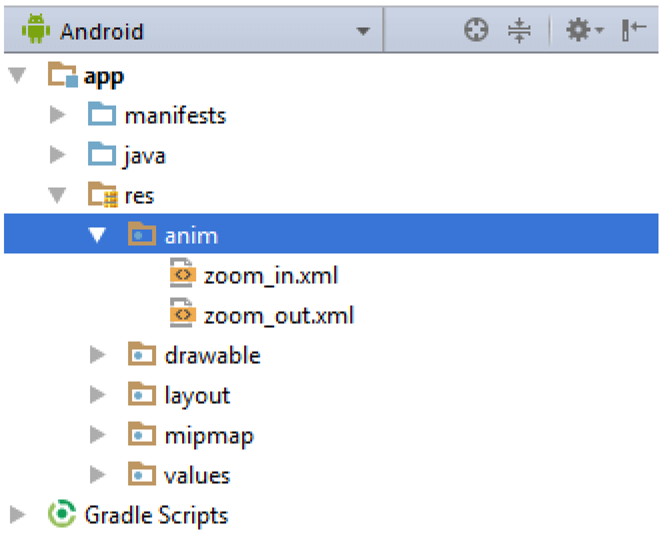
## Animation (Zoom In/Zoom Out)

### activity_main.xml

```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/btnZoomIn"
        android:layout_below="@+id/imgvw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Zoom In" android:layout_marginLeft="100dp" />
    <Button
        android:id="@+id/btnZoomOut"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnZoomIn"
        android:layout_toRightOf="@+id/btnZoomIn"
        android:text="Zoom Out" />
</RelativeLayout>
```

As discussed, we need to create an xml files to define zoom in and zoom out animations in new folder **anim** under **res** directory (**res → anim → zoom_in.xml**, **zoom_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 (**zoom_in.xml**, **zoom_out.zml**) under **anim** folder to define zoom in / out animation properties.



Now open **zoom_in.xml** file and write the code to set zoom in animation properties like as shown below

## zoom_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:fromXScale="2"
        android:fromYScale="2"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="4"
        android:toYScale="4" >
    </scale>
</set>
```

Now open **zoom_out.xml** file and write the code to set zoom out animation properties like as shown below

## zoom_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:duration="2500"
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale=".2"
        android:toYScale=".2" />
</set>
```

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

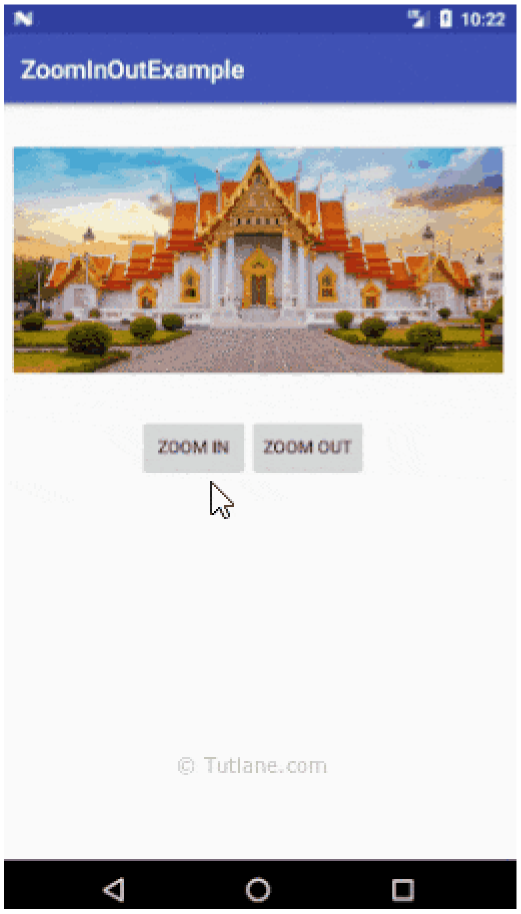
## MainActivity.java

```
package com.tutlane.zoominoutexample;
import android.media.Image;
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 btnzIn;
    private Button btnzOut;
    private ImageView img;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnzIn = (Button)findViewById(R.id.btnZoomIn);
        btnzOut = (Button)findViewById(R.id.btnZoomOut);
        img = (ImageView)findViewById(R.id.imgvw);
        btnzIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animZoomIn = AnimationUtils.loadAnimation(getApp
licationContext(),R.anim.zoom_in);
                img.startAnimation(animZoomIn);
            }
        });
        btnzOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animZoomOut = AnimationUtils.loadAnimation(getAp
plicationContext(),R.anim.zoom_out);
                img.startAnimation(animZoomOut);
            }
        });
    }
}
```
If you 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 Zoom In / Out Animations Example

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



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

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

Previous

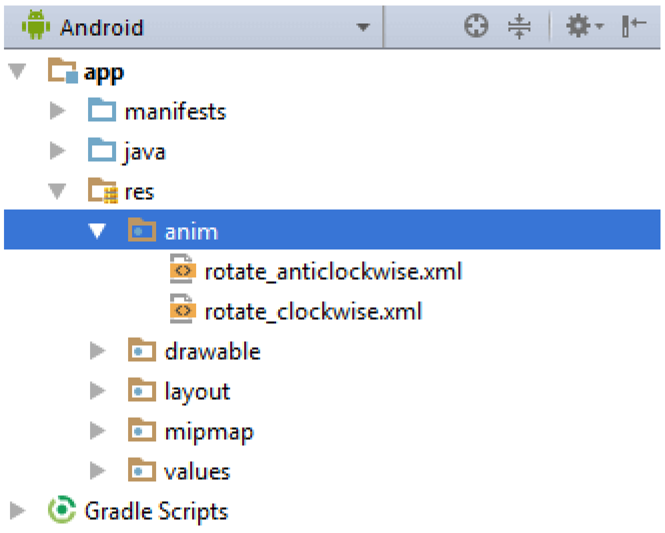
## Animation (Clockwise/AntiClockwise)

## activity_main.xml

```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/btnRClk"
        android:layout_below="@+id/imgvw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clockwise" android:layout_marginLeft="100dp" />
    <Button
        android:id="@+id/btnRAClk"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnRClk"
        android:layout_toRightOf="@+id/btnRClk"
        android:text="Anti Clockwise" />
</RelativeLayout>
```

As discussed, we need to create an xml files to define rotate animation either in clockwise or anti clockwise in new folder **anim** under **res** directory (**res → anim → rotate_clockwise.xml**, **rotate_anticlockwise.xml**) with required properties. In case **anim** folder not exists in **res** directory, create a new one.

Following is the example of creating XML files (**rotate_clockwise.xml**, **rotate_anticlockwise.xml**) under **anim** folder to define rotate animation in clockwise and anti-clockwise properties.



Now open **rotate_clockwise.xml** file and write the code to set rotate animation properties to rotate the object in clockwise like as shown below.

## rotate_clockwise.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:in
terpolator="@android:anim/cycle_interpolator">
    <rotate android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="5000" />
</set>
```

Now open **rotate_anticlockwise.xml** file and write the code to set rotate animation properties to rotate the object in anti-clockwise like as shown below

## rotate_anticlockwise.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:in
terpolator="@android:anim/cycle_interpolator">
    <rotate android:fromDegrees="360"
        android:toDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="5000" />
</set>
```

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

## MainActivity.java

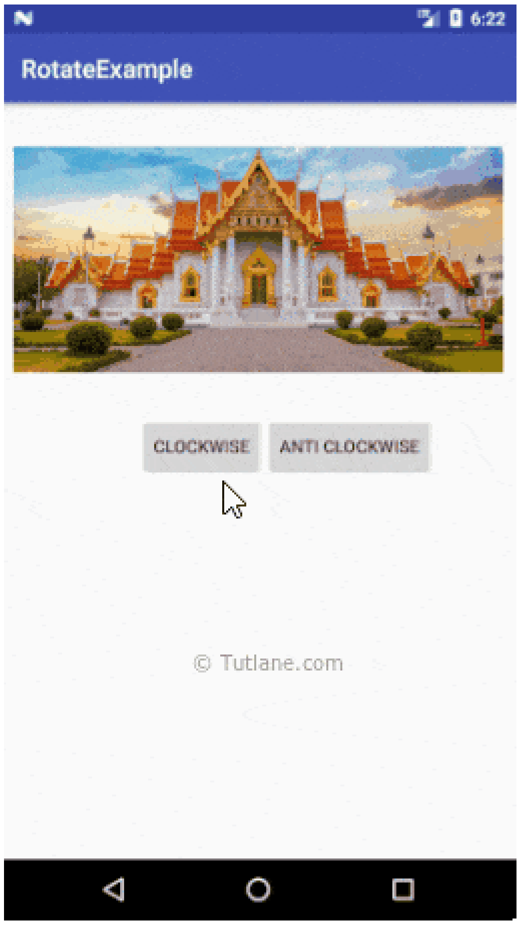
```
package com.tutlane.rotateexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {
    private Button btnrclock;
    private Button btnrantick;
    private ImageView img;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnrclock = (Button)findViewById(R.id.btnRClk);
        btnrantick = (Button)findViewById(R.id.btnRAClk);
        img = (ImageView)findViewById(R.id.imgvw);
        btnrclock.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation aniRotateClk = AnimationUtils.loadAnimation(getA
pplicationContext(),R.anim.rotate_clockwise);
                img.startAnimation(aniRotateClk);
            }
        });
        btnrantick.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animRotateAclk = AnimationUtils.loadAnimation(ge
tApplicationContext(),R.anim.rotate_anticlockwise);
                img.startAnimation(animRotateAclk);
            }
        });
    }
}
```

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

## Output of Android Rotate Animation Example

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



If you observe above result, whenever we are clicking on **Clockwise** or **Anti Clockwise** buttons, the image will rotate either in clockwise or anti-clockwise based on our functionality.

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