Animation tutorial

Animation attracts users. There are two types of animation in Android:

1. Drawable Animation:

   Drawable animation is a sequence of photos presented in a way that looks like a short clip.

2. View Animation:

   View animation performs transitions such as fading, rotating, transparency, moving and change of shapes. The following properties are used:

   - **Alpha** – It defines the transparency of an object. 0.0 is transparent and 1.0 is opaque. It is used to create fade-in and fade-out animation.
   - **Scale** – it is used to resize the image or text using x-axis and y-axis.
   - **Translate** – It is used to create a vertical or horizontal motion.
   - **Rotate** – It uses degree values to rotate where 0 degrees represents the starting point and 359 as a full rotation. 359 degrees rotation is clockwise, and -359 degrees is an anticlockwise rotation.
     pivotX and pivotY are used to determine the pivot point on the element through x and y axis. The values of pivot are in percentage form.
     E.g. pivotX= ”50%”;
     Another property of rotate is duration. Duration is the speed at which the element will rotate, and it is in milliseconds.

Animation Drawable

AnimationDrawable class is used to create a sequence of different images and play them in order. This gives it a movie kind of effect. This is done by creating an xml file in the drawable folder of the resources. It handles Drawable animation.

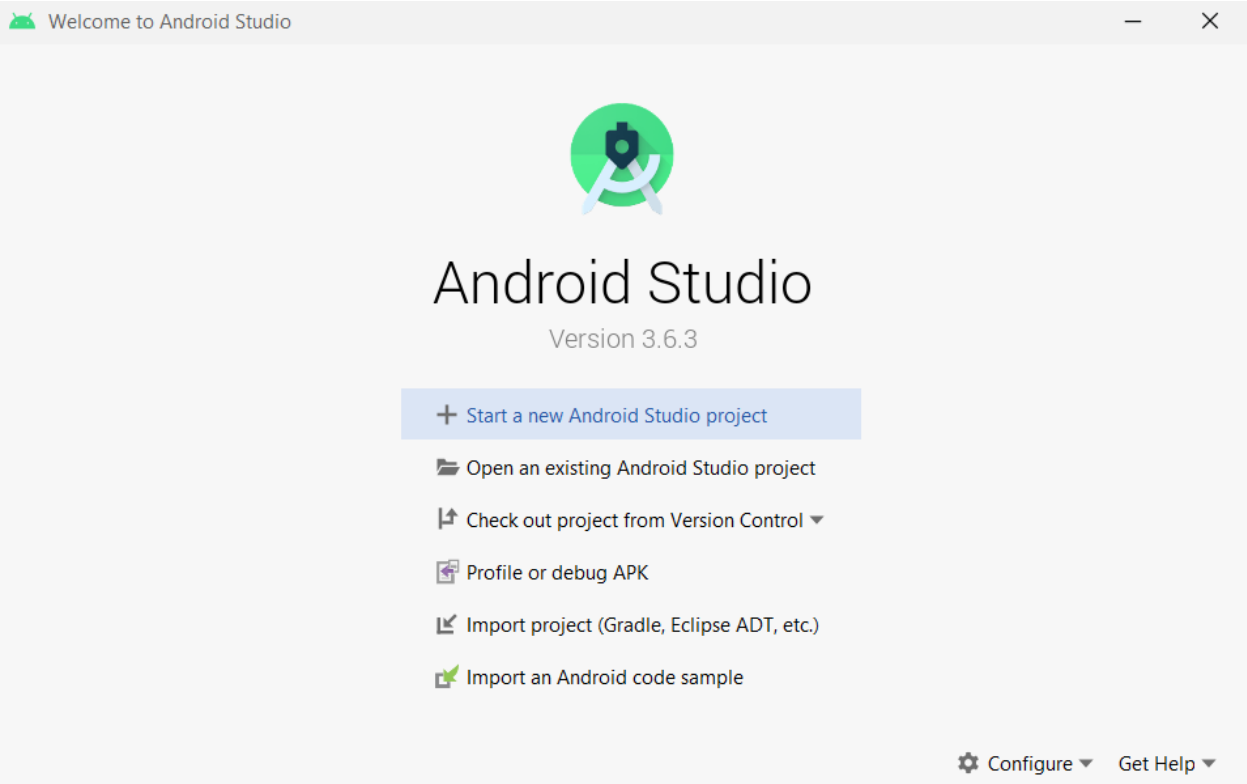
Animation

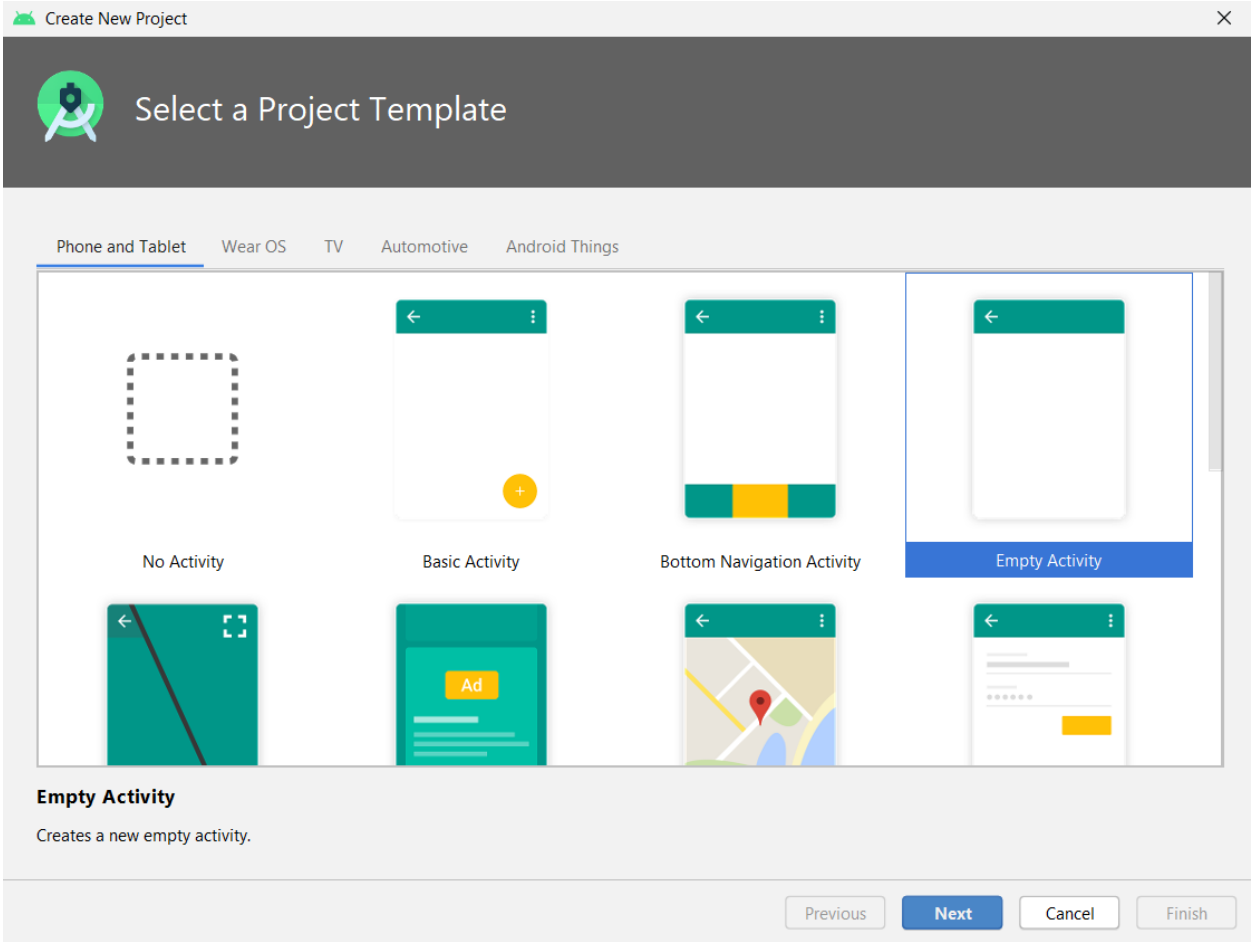
Animation package is used to inform Android to perform a series of simple transformations. It handles View Animation.

For the first part of the app, we create a Drawable Animation.

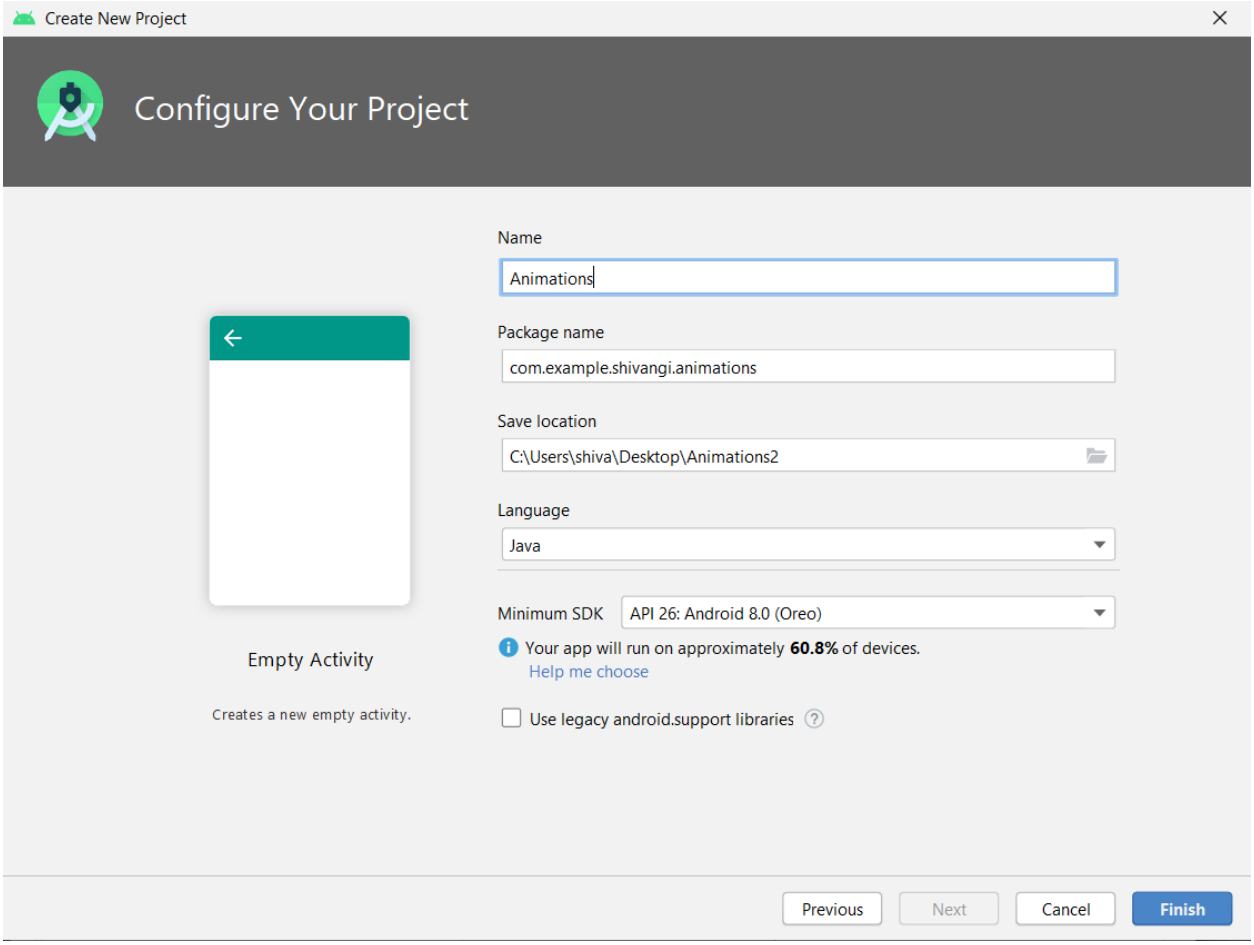
Let's Begin!

First open android studio and create a new project. Use the Empty Activity template.

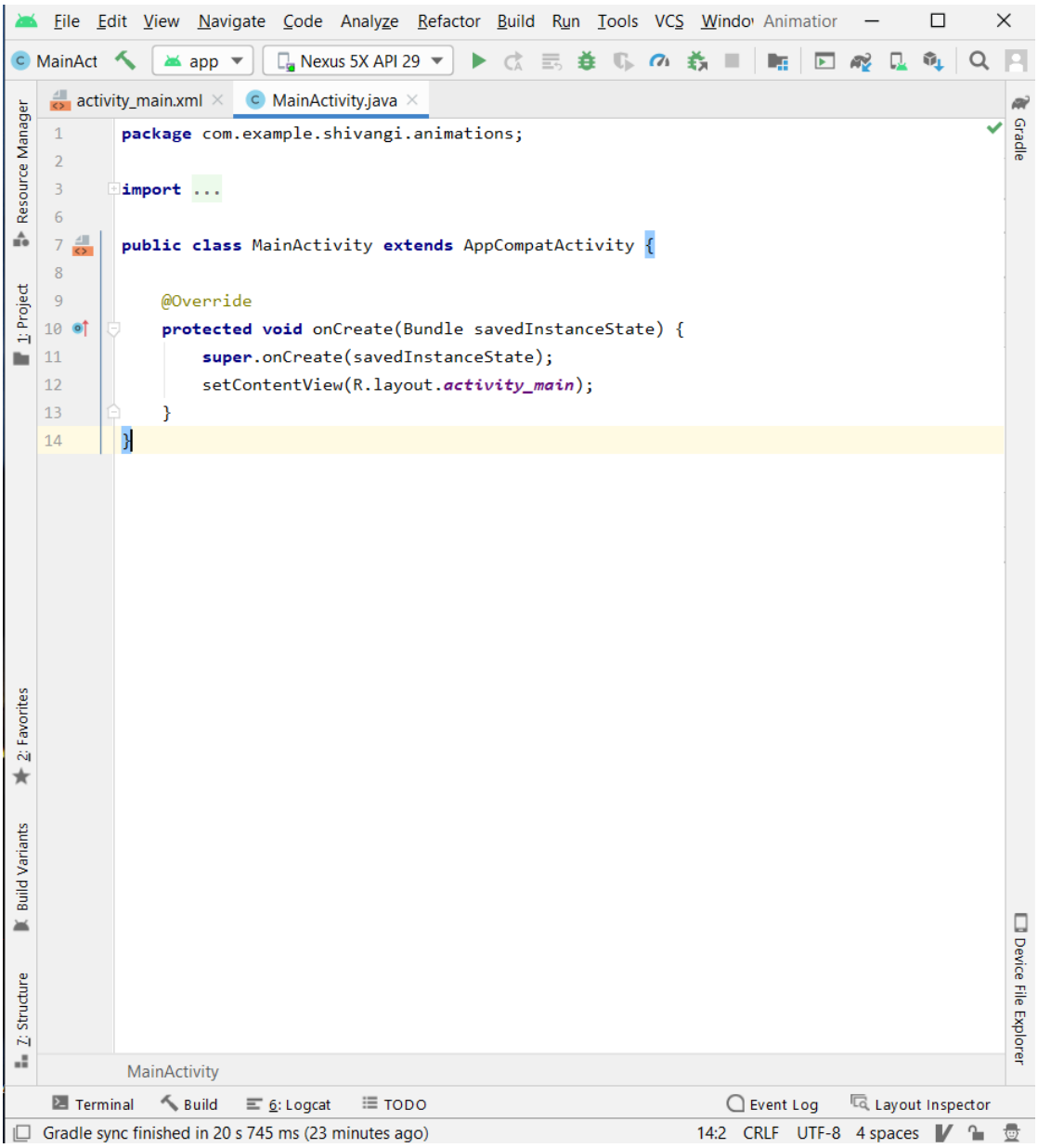



Create New Project
Select a Project Template
Phone and Tablet
Wear OS
TV
Automotive
Android Things
No Activity
Basic Activity
Bottom Navigation Activity
Empty Activity
Ad
Empty Activity
Creates a new empty activity.
Previous
Next
Cancel
Finish

Name the project Animations. You can select the location to save the project, set the minimum SDK to API 26 and click on the finish button.

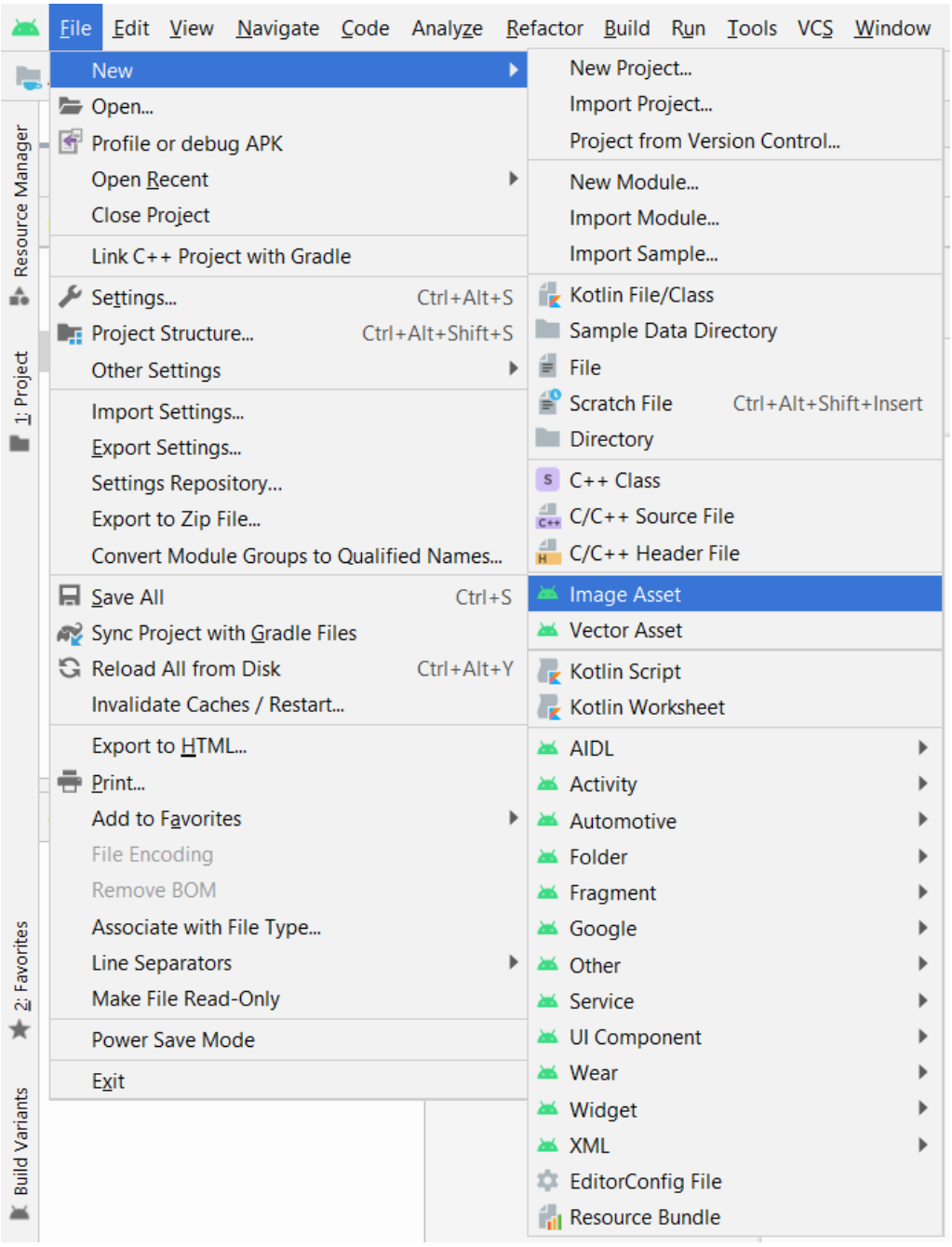


Once the project is synced, the following MainActivity.java file will appear.

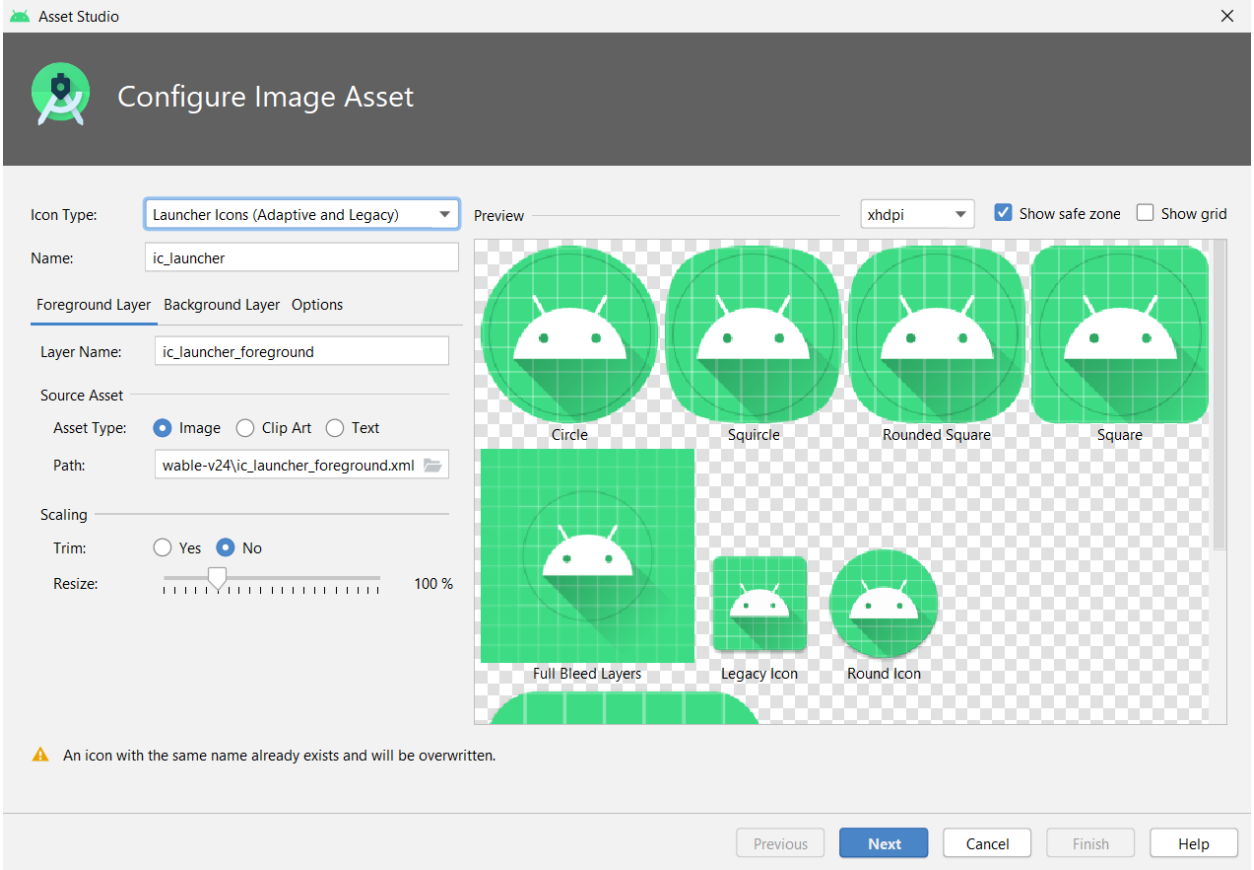


We will need two image assets that will be used in this app. So follow the steps bellow to create an image asset.

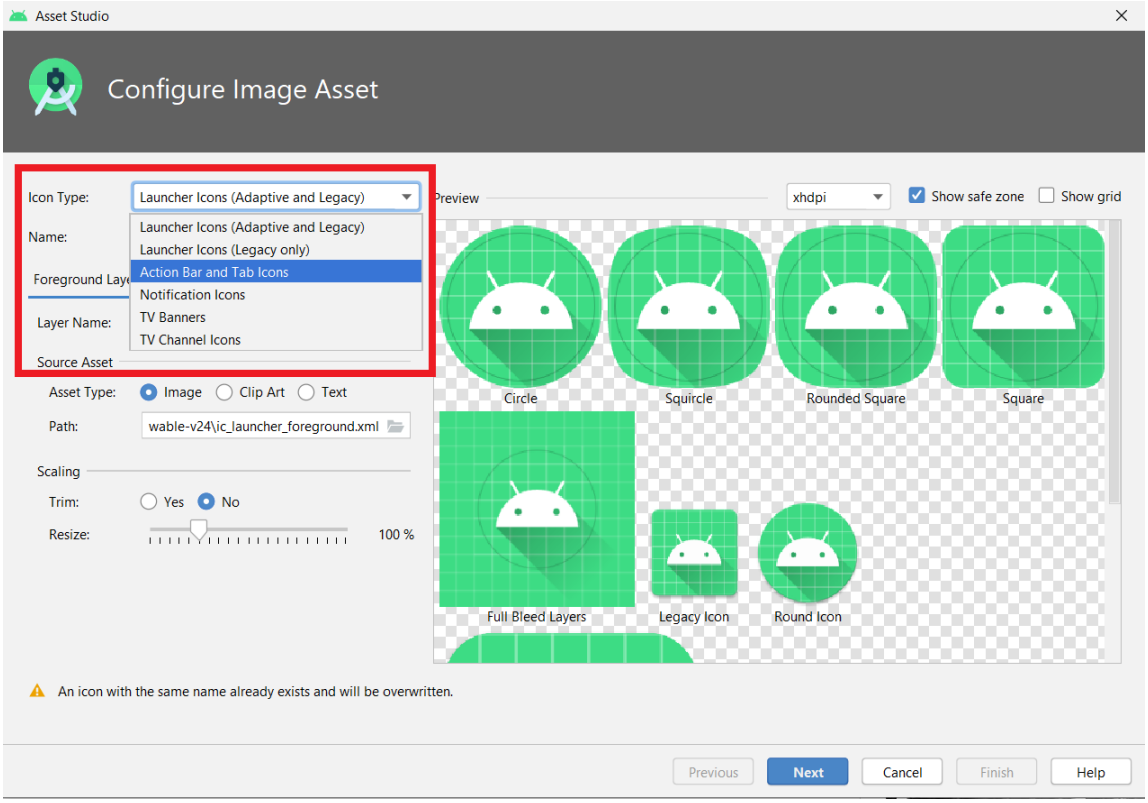
1) Click on the **File**. The go to **New > Image Asset.**

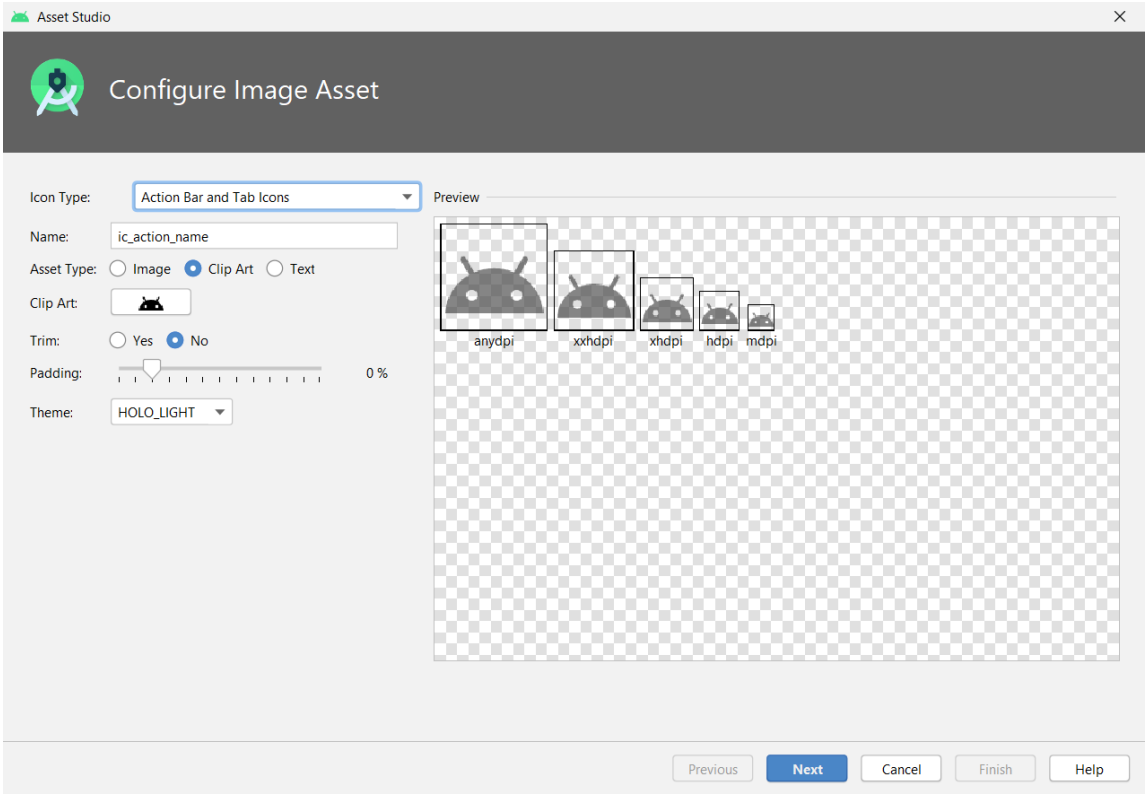

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window
New
Open...
Profile or debug APK
Open Recent
Close Project
Link C++ Project with Gradle
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
Other Settings
Import Settings...
Export Settings...
Settings Repository...
Export to Zip File...
Convert Module Groups to Qualified Names...
Save All Ctrl+S
Sync Project with Gradle Files
Reload All from Disk Ctrl+Alt+Y
Invalidate Caches / Restart...
Export to HTML...
Print...
Add to Favorites
File Encoding
Remove BOM
Associate with File Type...
Line Separators
Make File Read-Only
Power Save Mode
Exit
New Project...
Import Project...
Project from Version Control...
New Module...
Import Module...
Import Sample...
Kotlin File/Class
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Kotlin Worksheet
AIDL
Activity
Automotive
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
EditorConfig File
Resource Bundle
Resource Manager
1: Project
2: Favorites
Build Variants

The following tab will appear.



Click on the drop-down menu provided next to the Icon Type and select Action Bar and Tab Icons.

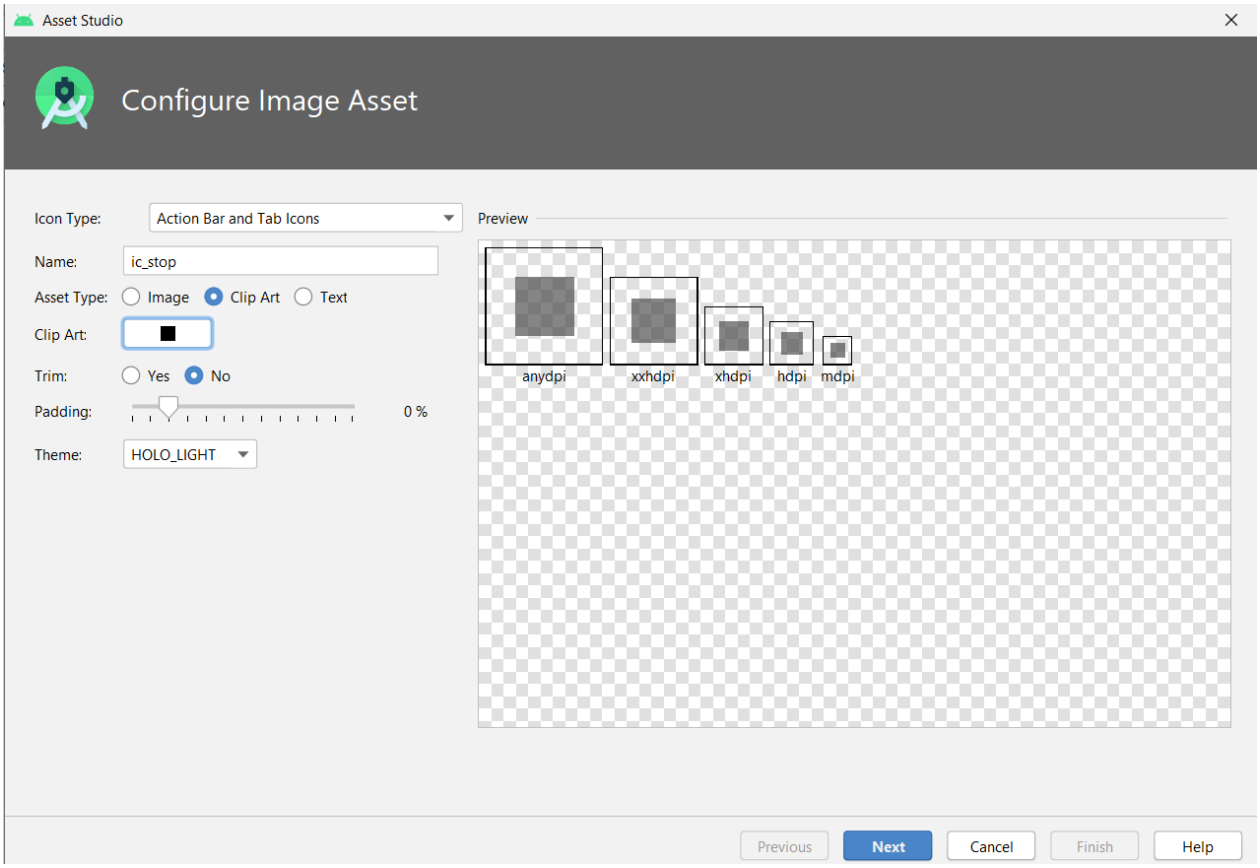

Asset Studio
Configure Image Asset
Icon Type:
Launcher Icons (Adaptive and Legacy)
Launcher Icons (Adaptive and Legacy)
Launcher Icons (Legacy only)
Action Bar and Tab Icons
Notification Icons
TV Banners
TV Channel Icons
Name:
Foreground Lay
Layer Name:
Source Asset
Asset Type:
Image
Clip Art
Text
Path:
wable-v24\ic_launcher_foreground.xml
Scaling
Trim:
Yes
No
Resize:
100 %
Preview
xhdpi
Show safe zone
Show grid
Circle
Squircle
Rounded Square
Square
Full Bleed Layers
Legacy Icon
Round Icon
An icon with the same name already exists and will be overwritten.
Previous
Next
Cancel
Finish
Help


Asset Studio
Configure Image Asset
Icon Type:
Action Bar and Tab Icons
Name:
ic_action_name
Asset Type:
Image
Clip Art
Text
Clip Art:
Trim:
Yes
No
Padding:
0 %
Theme:
HOLO_LIGHT
Preview
anydpi
xxhdpi
xhdpi
hdpi
mdpi
Previous
Next
Cancel
Finish
Help

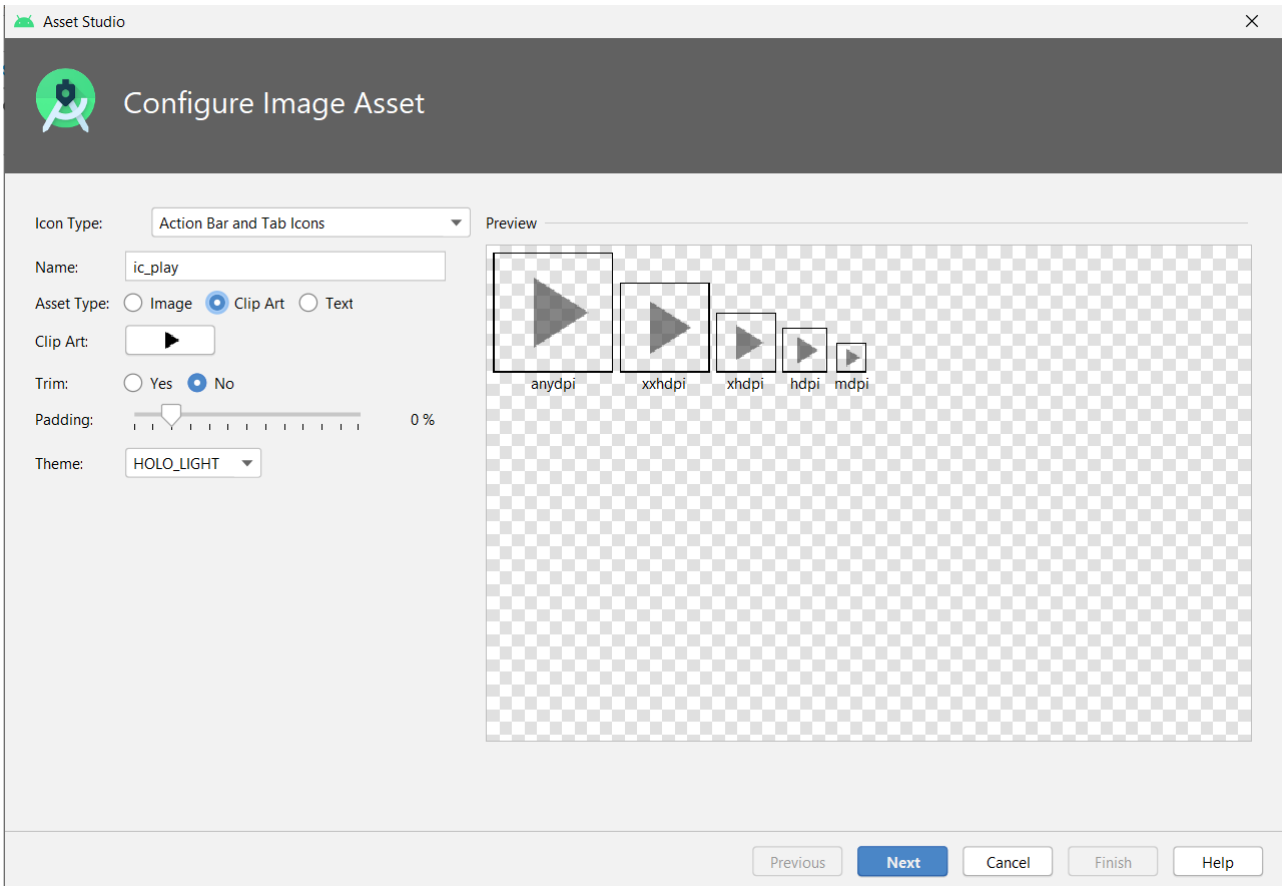
Click on the button provided next to the Clip Art Text field. The following window will appear.

In the search field, look and select the stop icon. Set the name to be *ic_stop*.

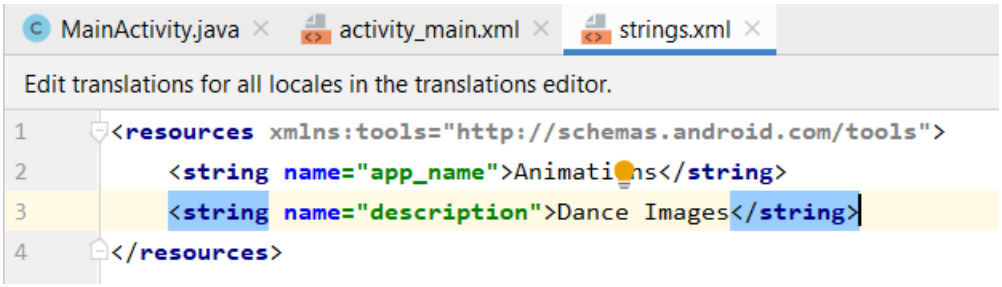
Select next and then click finish.



Follow the same steps for play arrow clip art. Set the name to be *ic_play*.



We will add a string in our string.xml file for the contentDescription required in the ImageView.



Add the following code to your activity_main.xml file. This code represents an ImageView and two ImageButtons which use the play and stop image assets we just created.

activity_main.xml | MainActivity.java

Code | Split | Design

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/danceImg"
        android:layout_width="200dp"
        android:layout_height="250dp"
        android:layout_marginTop="40dp"
        android:layout_marginBottom="39dp"
        android:contentDescription="@string/description"
        app:layout_constraintBottom_toTopOf="@+id/playFrame"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

```

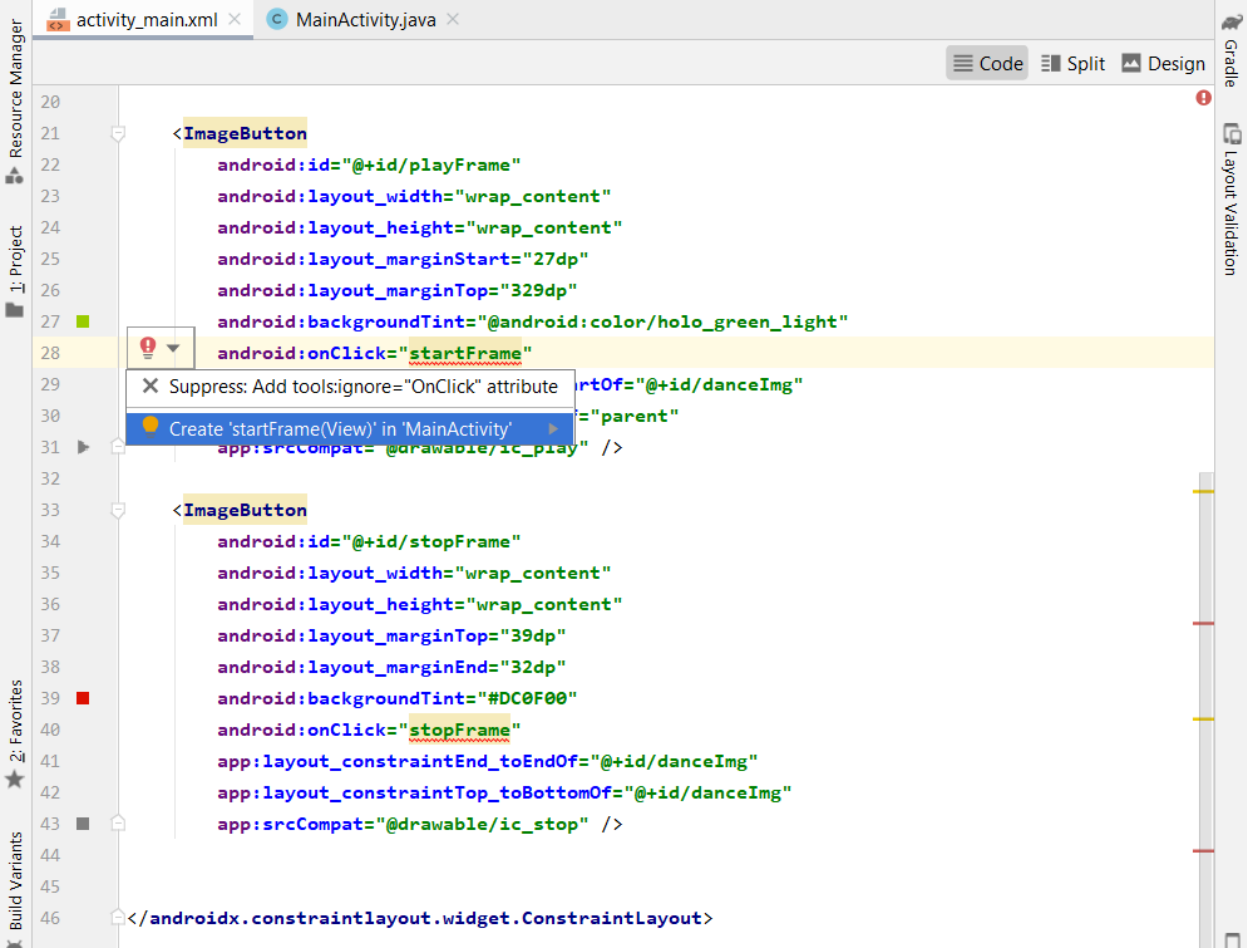
activity_main.xml | MainActivity.java

Code | Split | Design

```

    <ImageButton
        android:id="@+id/playFrame"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="27dp"
        android:layout_marginTop="329dp"
        android:backgroundTint="@android:color/holo_green_light"
        android:onClick="startFrame"
        app:layout_constraintStart_toStartOf="@+id/danceImg"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_play" />

    <ImageButton
        android:id="@+id/stopFrame"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="39dp"
        android:layout_marginEnd="32dp"
        android:backgroundTint="#DC0F00"
        android:onClick="stopFrame"
        app:layout_constraintEnd_toEndOf="@+id/danceImg"
        app:layout_constraintTop_toBottomOf="@+id/danceImg"
        app:srcCompat="@drawable/ic_stop" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

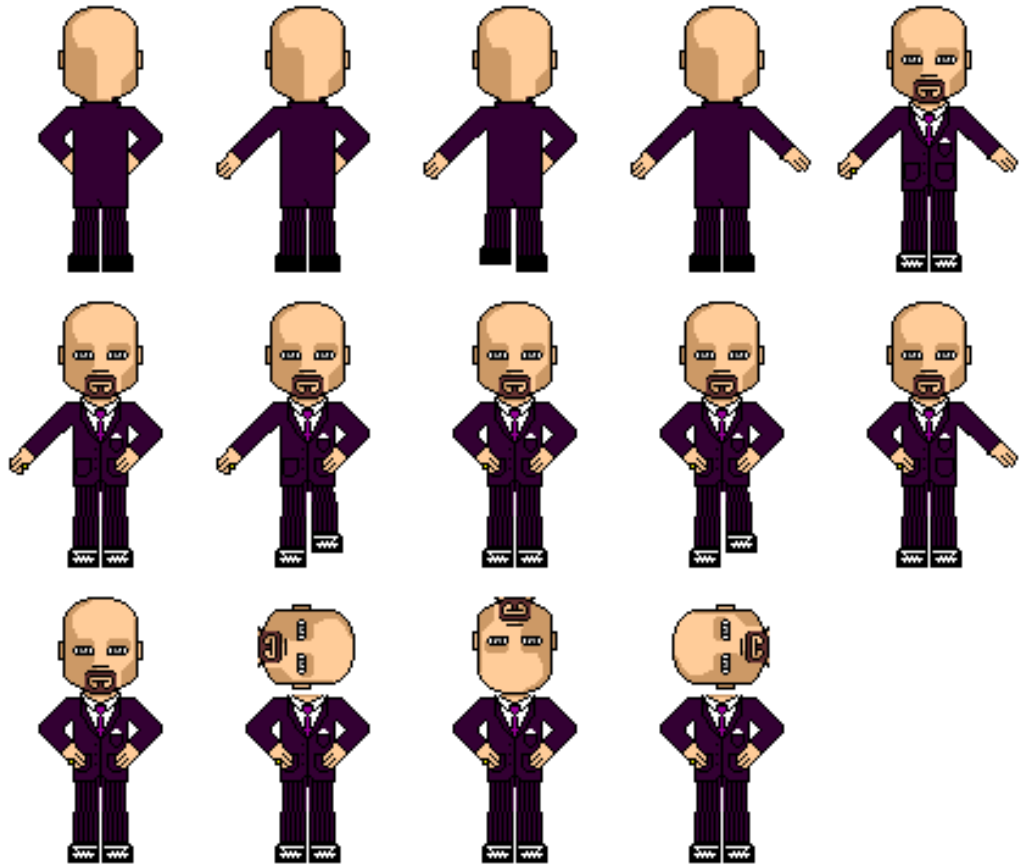
The red zigzag line underneath the onClick function represents a method that is not yet created in the MainActivity.java class.

To implement the method, click on the red blub and select Create ‘startFrame(View)’ in MainActivity.java. This will create the onClick methods for you.

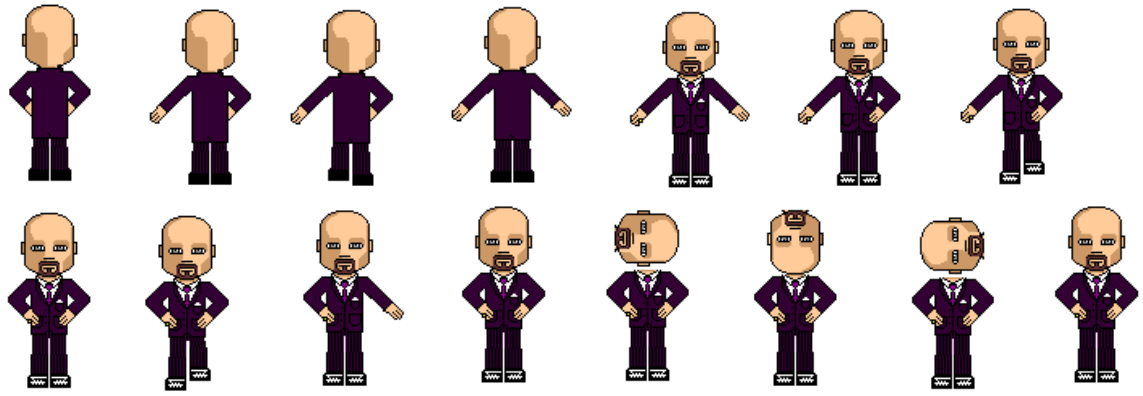


```java
package com.example.shivangi.animations;

import ...

public class MainActivity extends AppCompatActivity {

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


    public void startFrame(View view) {
    }

    public void stopFrame(View view) {
    }
}
```

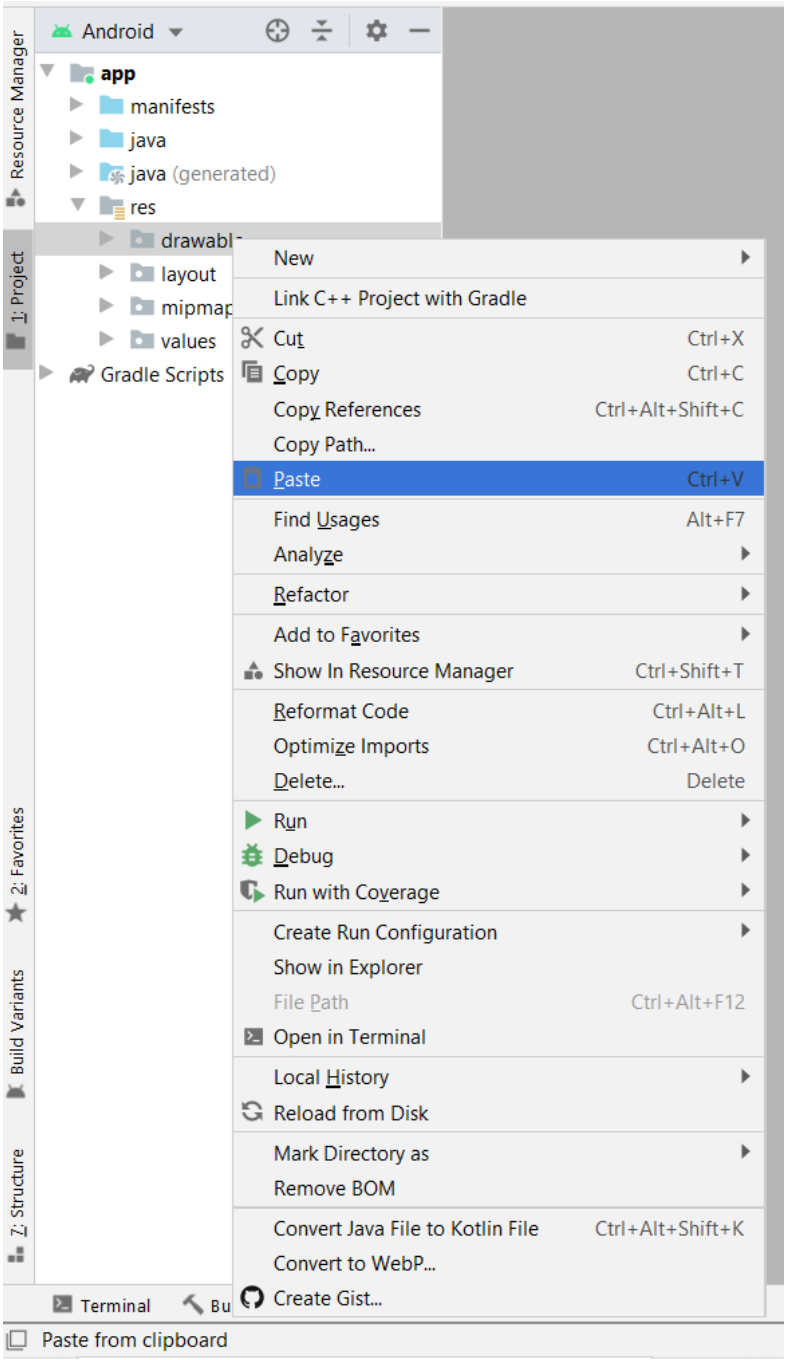
The next step is to add images to the drawable resource. These images are similar to each other as they are transitions of the same image. What I have done is that I downloaded the image below. I cut and saved each individual character from the image using the paint app on the desktop.

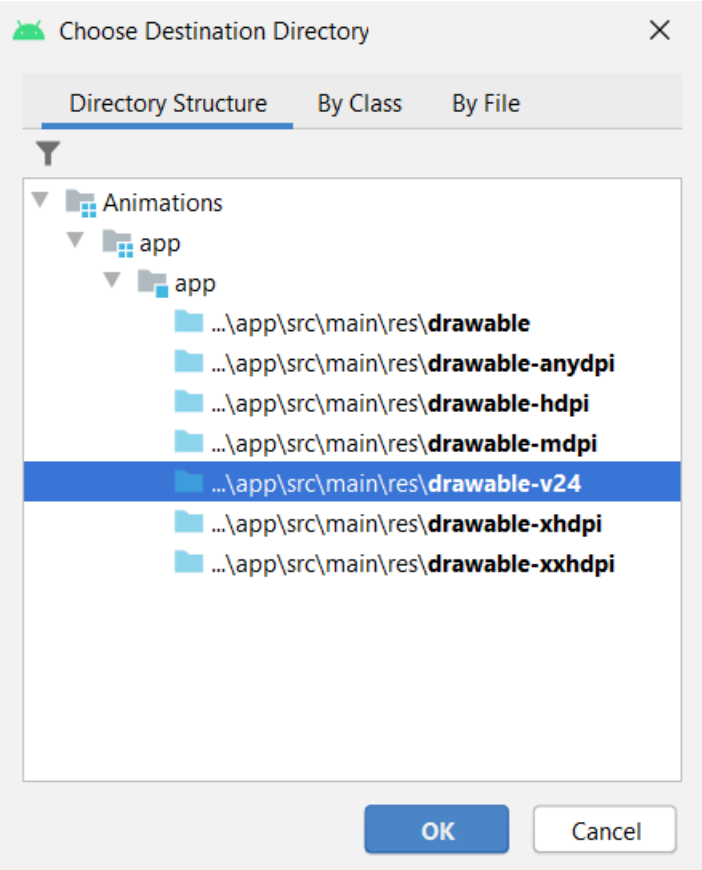


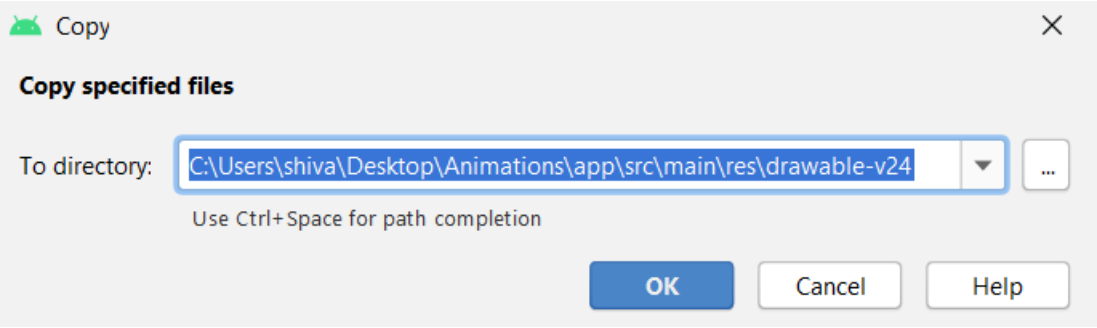
Below are the cut images.



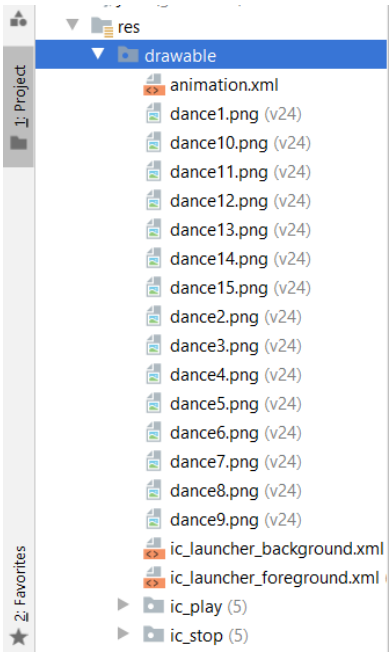
Copy all the 15 images and add the images to the drawable folder.


Resource Manager
1: Project
2: Favorites
Build Variants
7: Structure
Android
app
manifests
java
java (generated)
res
drawabl
layout
mipmap
values
Gradle Scripts
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy References Ctrl+Alt+Shift+C
Copy Path...
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Show In Resource Manager Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run
Debug
Run with Coverage
Create Run Configuration
Show in Explorer
File Path Ctrl+Alt+F12
Open in Terminal
Local History
Reload from Disk
Mark Directory as
Remove BOM
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Convert to WebP...
Create Gist...
Terminal
Paste from clipboard


Choose Destination Directory
Directory Structure
By Class
By File
Animations
app
app
...\app\src\main\res\drawable
...\app\src\main\res\drawable-anydpi
...\app\src\main\res\drawable-hdpi
...\app\src\main\res\drawable-mdpi
...\app\src\main\res\drawable-v24
...\app\src\main\res\drawable-xhdpi
...\app\src\main\res\drawable-xxhdpi
OK
Cancel

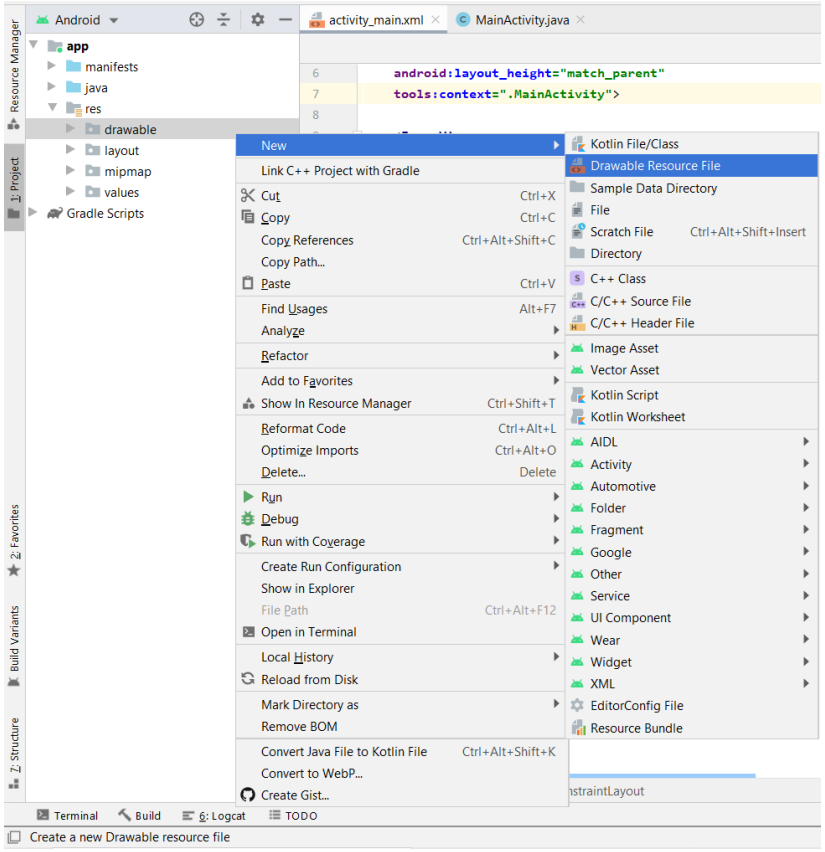

Copy
Copy specified files
To directory:
C:\Users\shiva\Desktop\Animations\app\src\main\res\drawable-v24
...
Use Ctrl+Space for path completion
OK
Cancel
Help

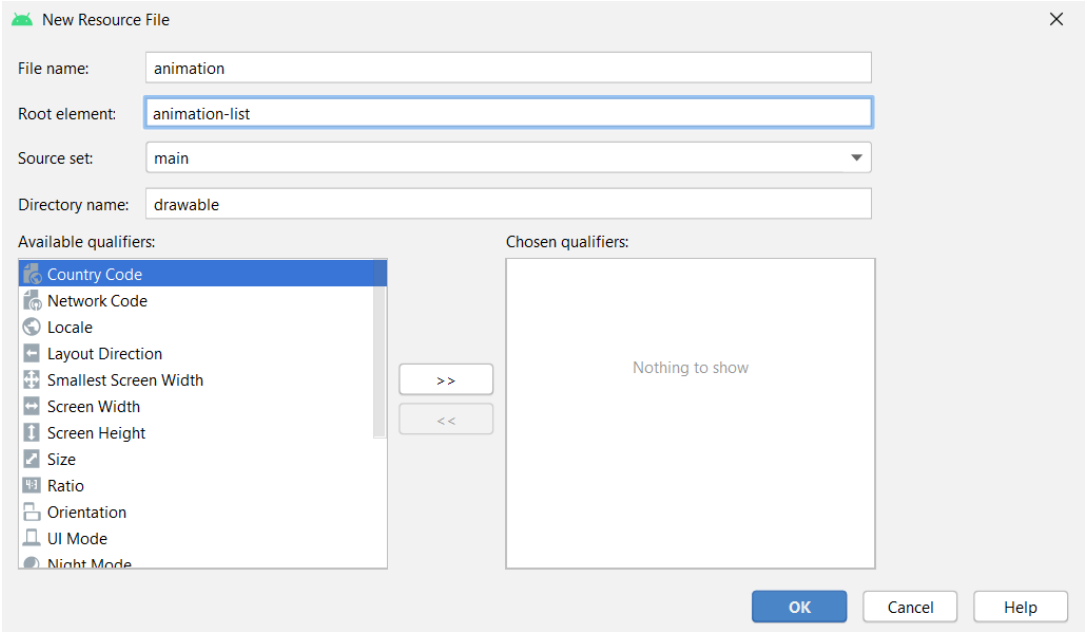
Your drawable folder will look like the one below.

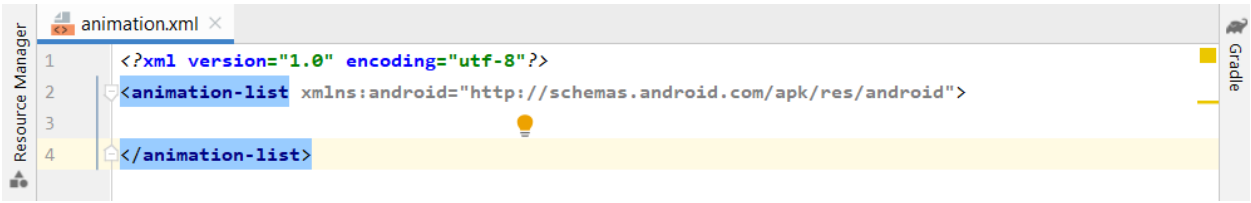


Now we will create an animation.xml to perform the transitions for every image.

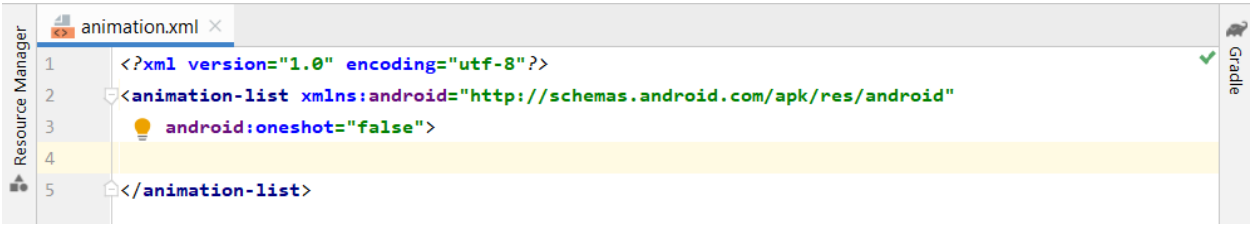
Create a new drawable resource file and name it animation and set the root-element as animation-list.







In the animation-list tag add android:oneshot property to be false. This ensures that the animation will play continuously.



Now we will add each image as an item and set the duration of each image to 100 milliseconds. This will display each images just for 100 milliseconds.

```xml
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/dance1" android:duration="100"/>
    <item android:drawable="@drawable/dance2" android:duration="100"/>
    <item android:drawable="@drawable/dance3" android:duration="100"/>
    <item android:drawable="@drawable/dance4" android:duration="100"/>
    <item android:drawable="@drawable/dance5" android:duration="100"/>
    <item android:drawable="@drawable/dance6" android:duration="100"/>
    <item android:drawable="@drawable/dance7" android:duration="100"/>
    <item android:drawable="@drawable/dance8" android:duration="100"/>
    <item android:drawable="@drawable/dance9" android:duration="100"/>
    <item android:drawable="@drawable/dance10" android:duration="100"/>
    <item android:drawable="@drawable/dance11" android:duration="100"/>
    <item android:drawable="@drawable/dance12" android:duration="100"/>
    <item android:drawable="@drawable/dance13" android:duration="100"/>
    <item android:drawable="@drawable/dance14" android:duration="100"/>
    <item android:drawable="@drawable/dance15" android:duration="100"/>

</animation-list>
```

In your MainActivity.java file, instantiate the AnimationDrawable instance as a class variable.

```java
package com.example.shivangi.animations;

import ...

public class MainActivity extends AppCompatActivity {

    AnimationDrawable animationDrawable;

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


    public void startFrame(View view) {
    }

    public void stopFrame(View view) {
    }
}
```

Now we will get access to the ImageView created in the activity_main.xml file. In the onCreate() method, add the following code.

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

    ImageView frameImg = findViewById(R.id.danceImg);

}
```

After getting access to the ImageView container, we will set the background by using the setBackgroundResource() method provided to the 15 images set in the animation.xml file. AnimationdDrawer is used to get the images and display the animation to the user. The AnimationDrawer controls and displays each image accordingly to the duration.

Add the code below to your onCreate() method.

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

    ImageView frameImg = findViewById(R.id.danceImg);
    frameImg.setBackgroundResource(R.drawable.animation);
    animationDrawable = (AnimationDrawable) frameImg.getBackground();

}
```

The AnimationDrawer provides start and stop methods that will be used when clicking the start and stop buttons. Since each button is linked to its own onClick method, we will add the following code in the respective methods.

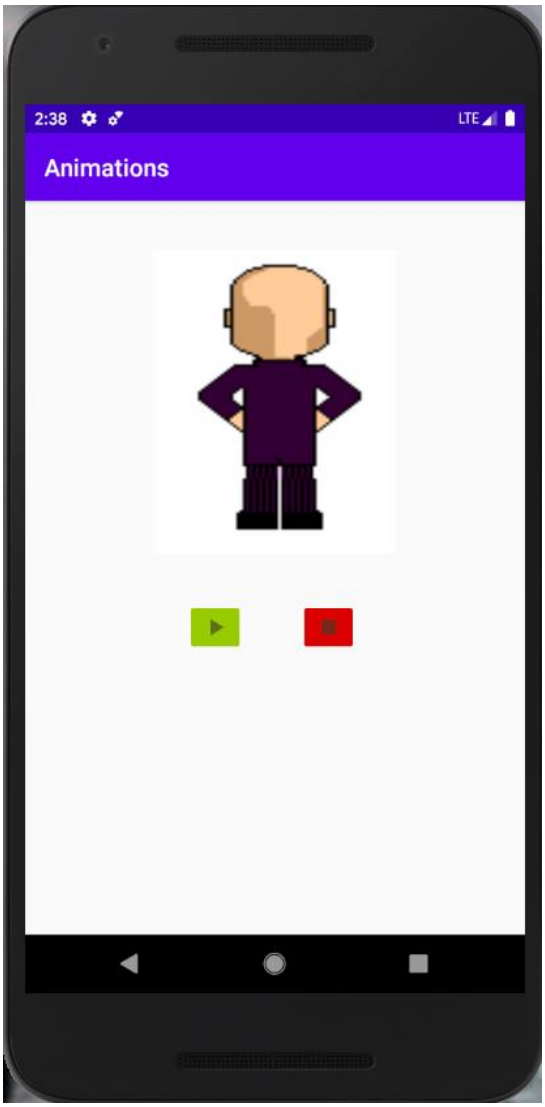
```
public void startFrame(View view) {
    animationDrawable.start();
}

public void stopFrame(View view) {
    animationDrawable.stop();
}
```

Your final MainActivity.java file will look as follows:

MainActivity.java

```java
package com.example.shivangi.animations;

import ...

public class MainActivity extends AppCompatActivity {

    AnimationDrawable animationDrawable;

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

        ImageView frameImg = findViewById(R.id.danceImg);
        frameImg.setBackgroundResource(R.drawable.animation);
        animationDrawable = (AnimationDrawable) frameImg.getBackground();

    }


    public void startFrame(View view) {
        animationDrawable.start();
    }

    public void stopFrame(View view) {
        animationDrawable.stop();
    }
}
```

Run the app. The following are snapshots of the app.



We will now create a View Animation using the rotation property.

First, add one more string to the string.xml file. This string is for the textView we will create in the next step.

MainActivity.java | activity_main.xml | strings.xml

Edit translations for all locales in the translations editor.

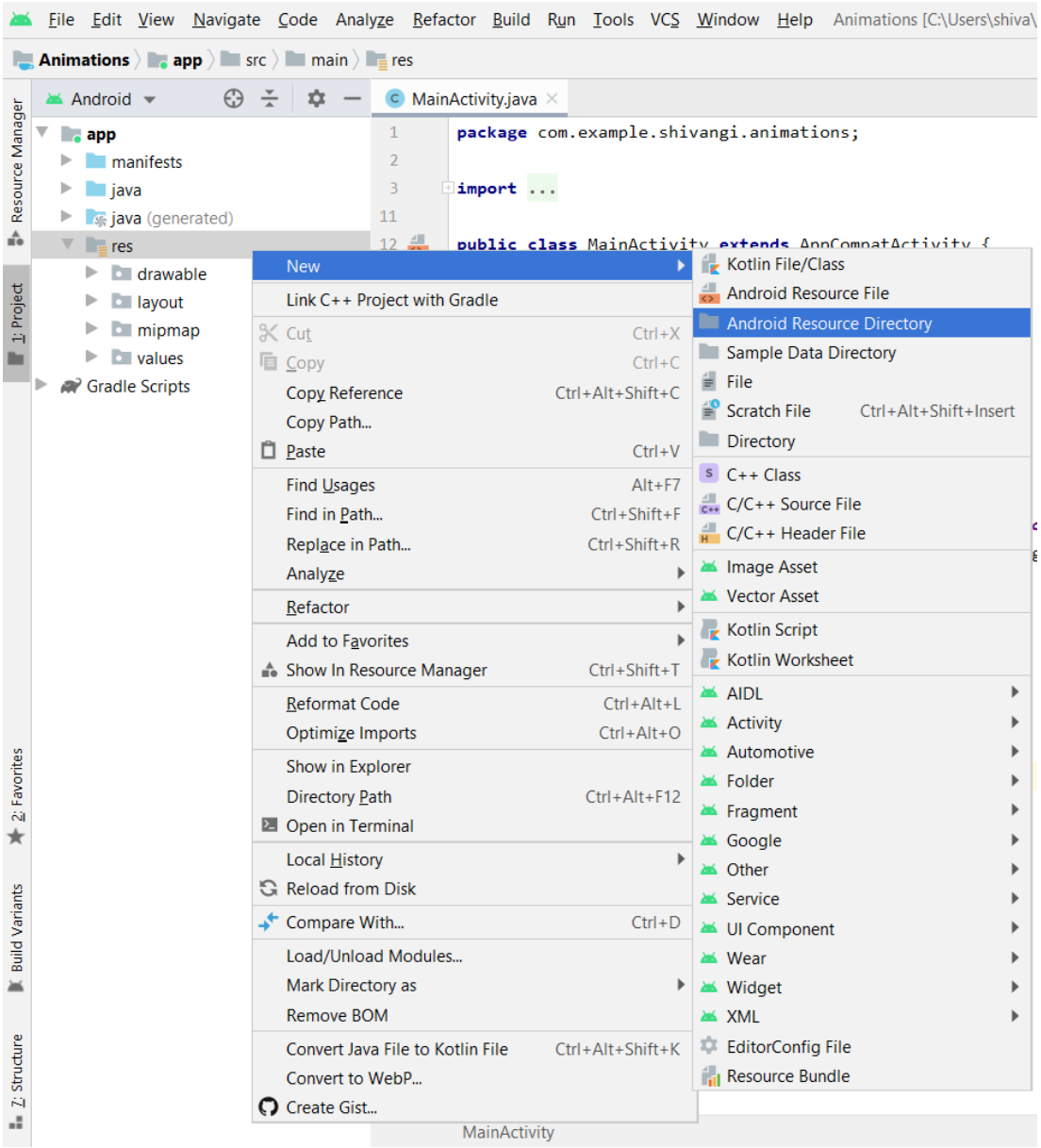
```
1  <resources xmlns:tools="http://schemas.android.com/tools">
2      <string name="app_name">Animations</string>
3      <string name="description">Dance Images</string>
4      <string name="text">DANCE!</string>
5  </resources>
```

Add a textView after the last ImageButton in the activity_main.xml file.

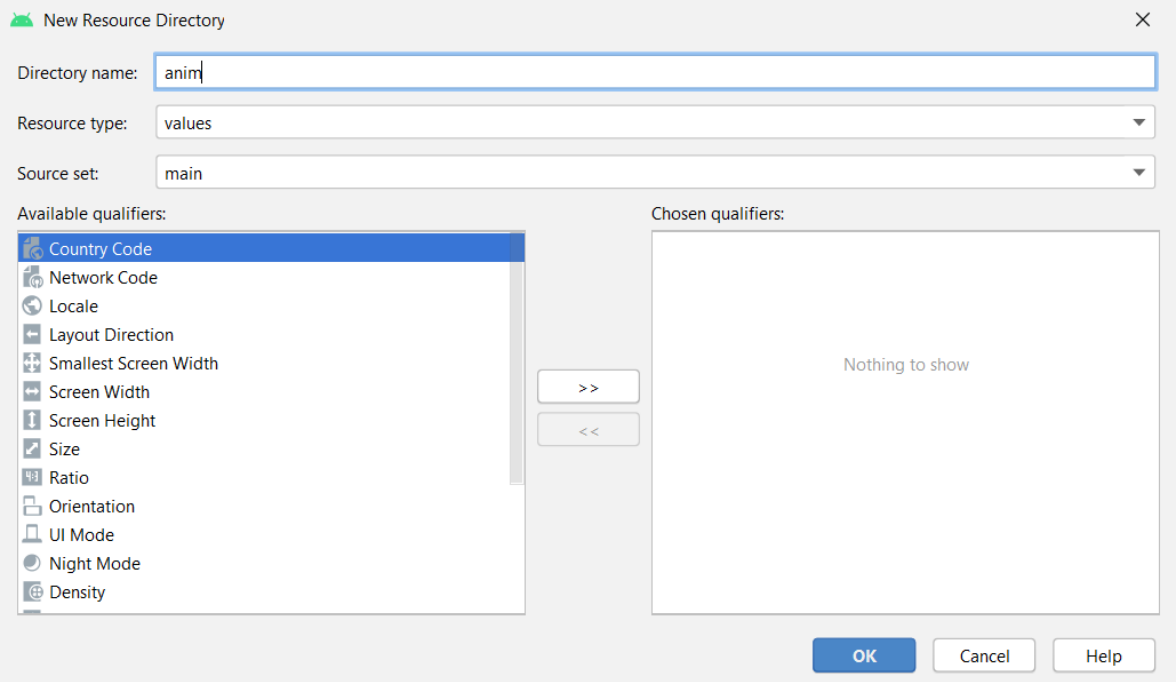
```
45     <TextView
46         android:id="@+id/textView"
47         android:layout_width="wrap_content"
48         android:layout_height="wrap_content"
49         android:layout_marginBottom="221dp"
50         android:text="@string/text"
51         android:textColor="#00BCD4"
52         android:textSize="40sp"
53         android:textStyle="bold|italic"
54         app:layout_constraintBottom_toBottomOf="parent"
55         app:layout_constraintEnd_toEndOf="parent"
56         app:layout_constraintStart_toStartOf="parent"
57         app:layout_constraintTop_toBottomOf="@+id/danceImg" />
```

Now create an anim directory under the res directory. Follow the steps below to create a directory in the res folder.

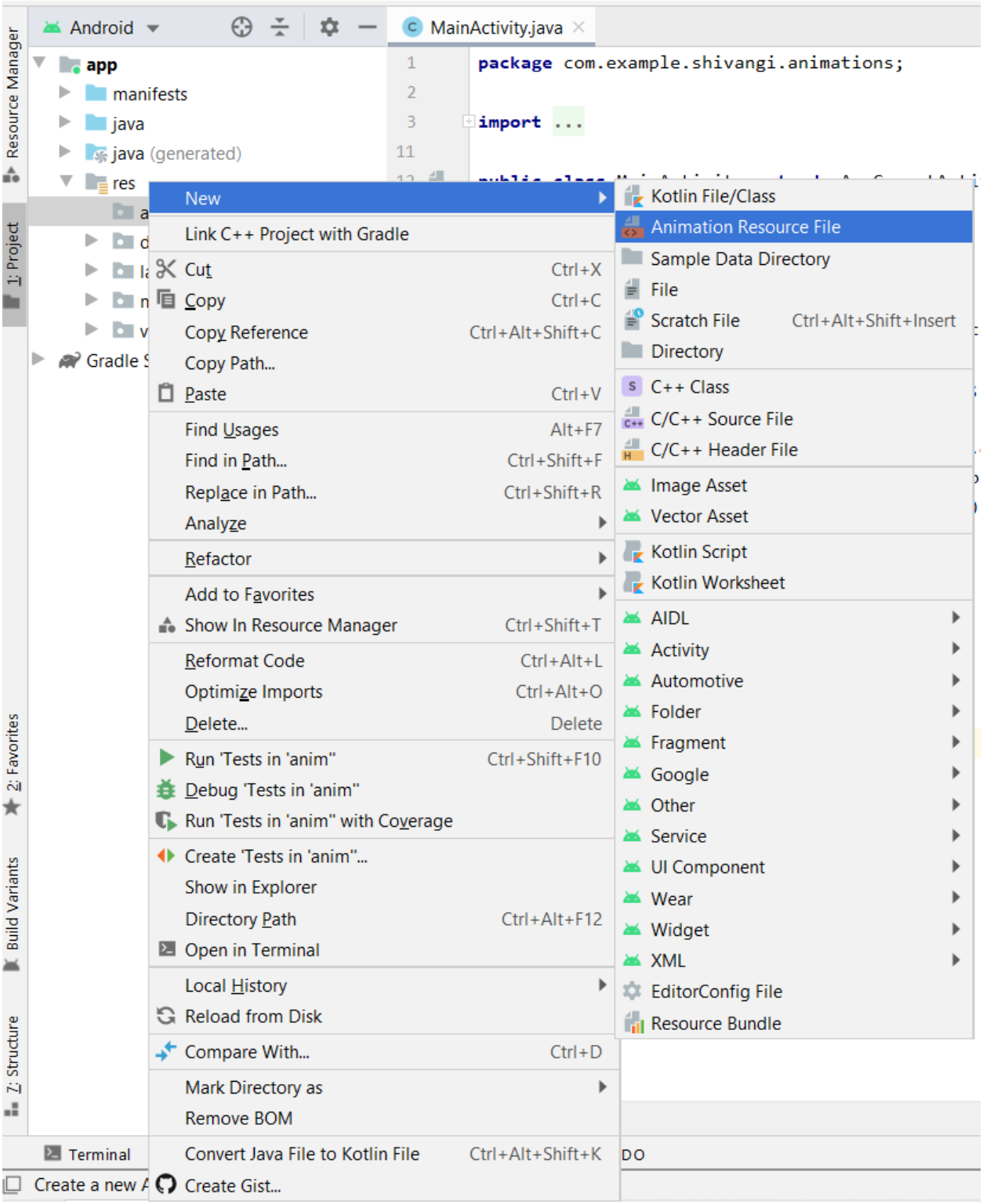
Right-click on the res folder, then select **New > Android Resource Directory.**



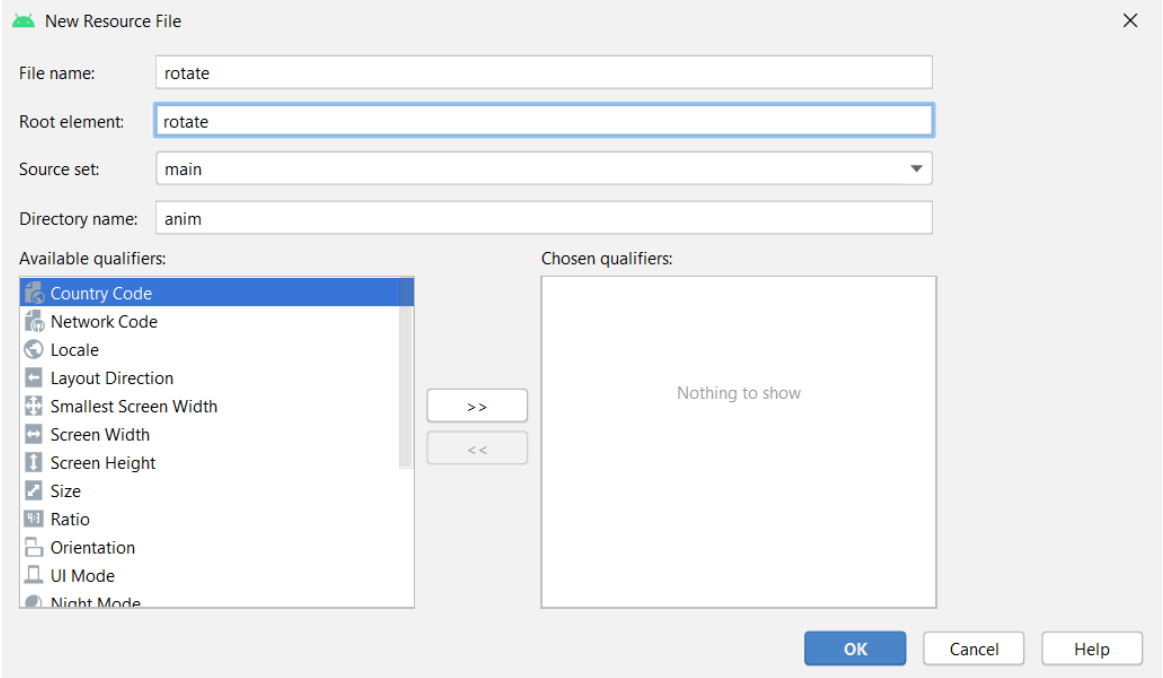
Set the directory name as anim and click ok.



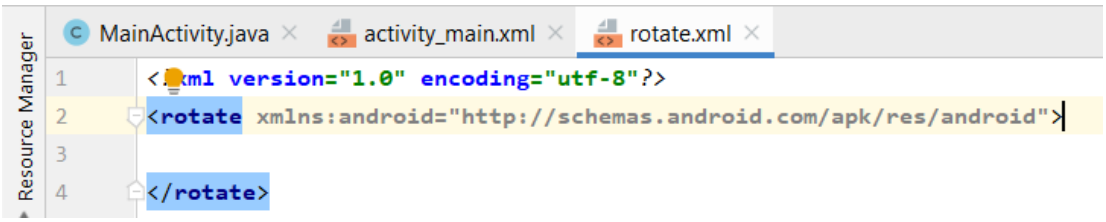
We will now create a rotate.xml file in the anim folder we just created above. Right click on the anim folder. Then select New > Animation resource file.



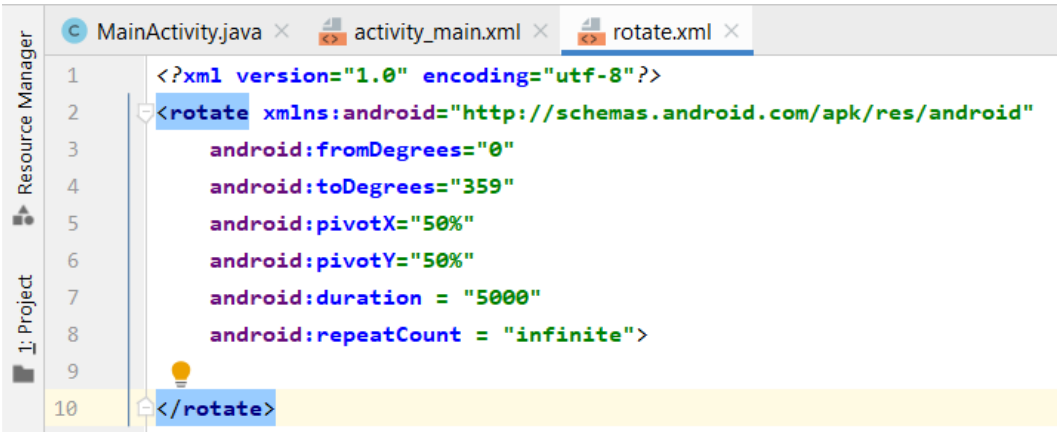
Name the file rotate and select rotate as the root element. Click ok.



After creating the rotate.xml file, it will look like below.



Add the following code inside the first tag.



In the above code, we set the repeatCount to infinite as we want it to continue rotating throughout.

Now we will gain access to the textView by implementing the following code in the onCreate() method of MainActivity.java.

```
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_main);
21
22          ImageView frameImg = findViewById(R.id.danceImg);
23          frameImg.setBackgroundResource(R.drawable.animation);
24          animationDrawable = (AnimationDrawable) frameImg.getBackground();
25
26          TextView dance = findViewById(R.id.textView);
27
28      }
```

Declare Animation as a class variable.

```
15  public class MainActivity extends AppCompatActivity {
16
17      AnimationDrawable animationDrawable;
18      Animation animation;
```

Animation package provides you with different classes that can be used. One of the main classes is the AnimationUtils that defines the common utilities for working with animations.

We will use this class to load and start the animation.

```
20      @Override
21      protected void onCreate(Bundle savedInstanceState) {
22          super.onCreate(savedInstanceState);
23          setContentView(R.layout.activity_main);
24
25          ImageView frameImg = findViewById(R.id.danceImg);
26          frameImg.setBackgroundResource(R.drawable.animation);
27          animationDrawable = (AnimationDrawable) frameImg.getBackground();
28
29          TextView dance = findViewById(R.id.textView);
30          animation = AnimationUtils.loadAnimation( context: this,R.anim.rotate);
31          dance.startAnimation(animation);
32
33      }
34
```

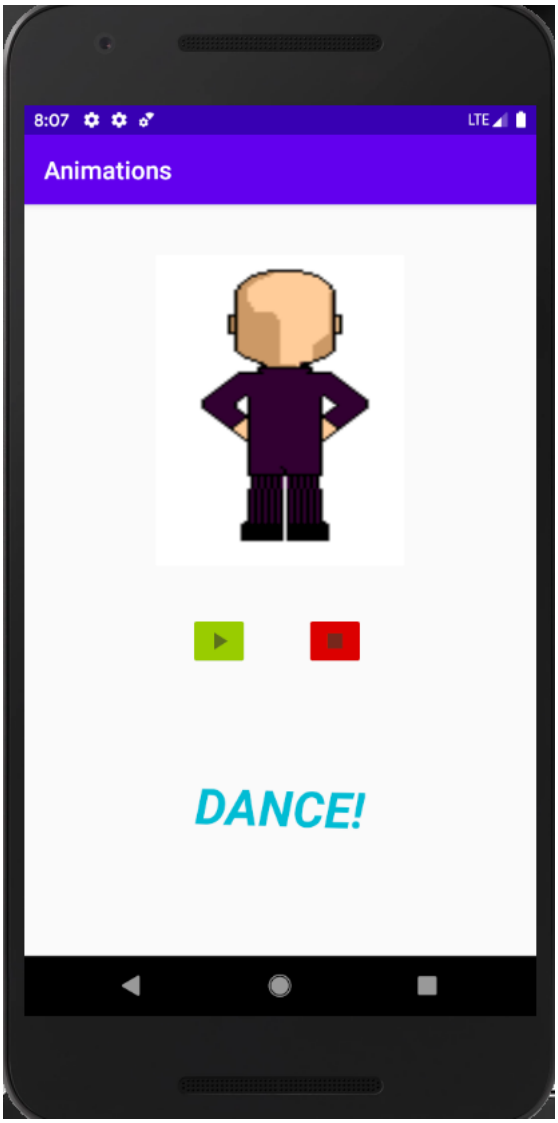
Your MainActivity.java class will look as follows:

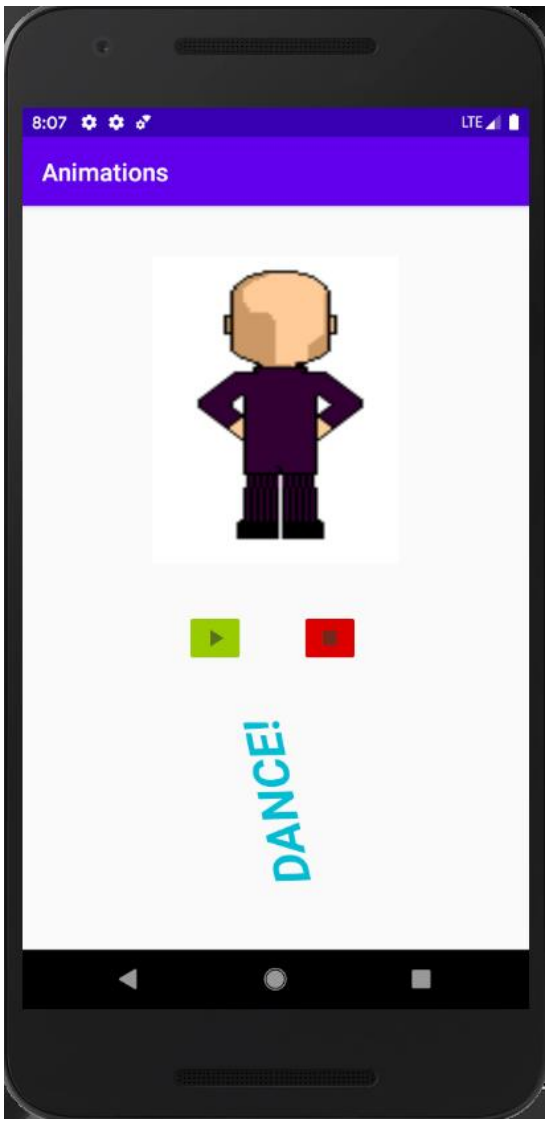
MainActivity.java

```java
package com.example.shivangi.animations;

import ...

public class MainActivity extends AppCompatActivity {

    AnimationDrawable animationDrawable;
    Animation animation;

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

        ImageView frameImg = findViewById(R.id.danceImg);
        frameImg.setBackgroundResource(R.drawable.animation);
        animationDrawable = (AnimationDrawable) frameImg.getBackground();

        TextView dance = findViewById(R.id.textView);
        animation = AnimationUtils.loadAnimation( context: this,R.anim.rotate);
        dance.startAnimation(animation);

    }


    public void startFrame(View view) {
        animationDrawable.start();
    }

    public void stopFrame(View view) {
        animationDrawable.stop();
    }
}
```

MainActivity › startFrame()

Run the app. The following are snapshots of the app.





RESOURCES:

https://developer.android.com/reference/android/graphics/drawable/AnimationDrawable

https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/graphics/drawable-animation.html

https://developer.android.com/guide/topics/resources/animation-resource

https://developer.android.com/guide/topics/graphics/view-animation

https://developer.android.com/reference/android/view/animation/package-summary