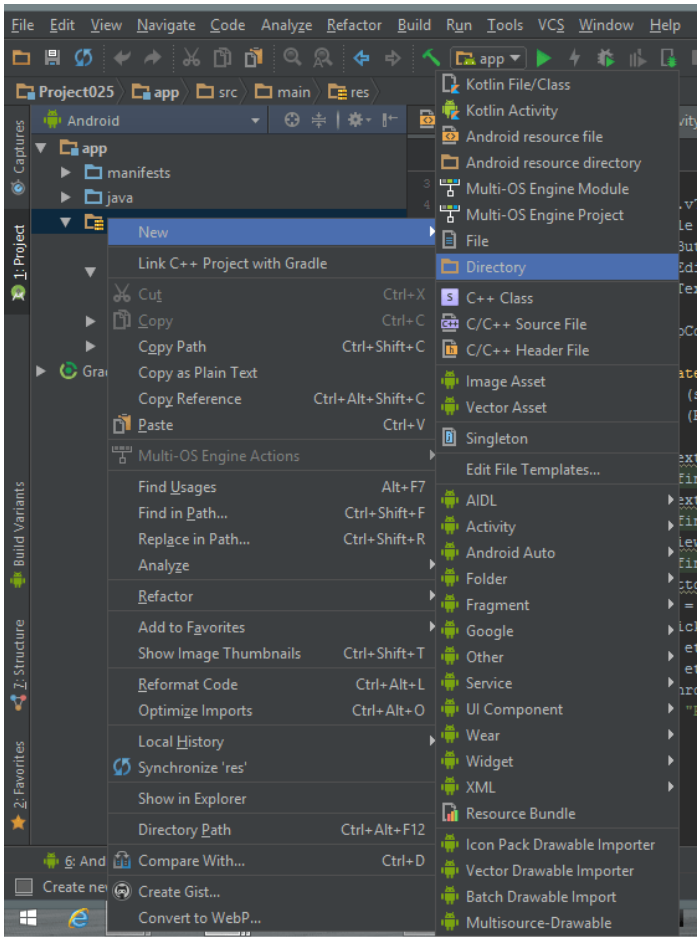
1-Insert audio in Android Studio App

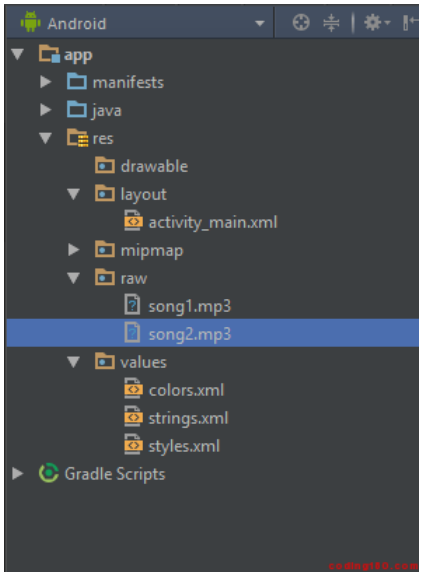
First, create a new project called Playsound.

Have 2 buttons with ID (button1 , button2), then when you press play the respective audio file, Sound files store them in the same application.

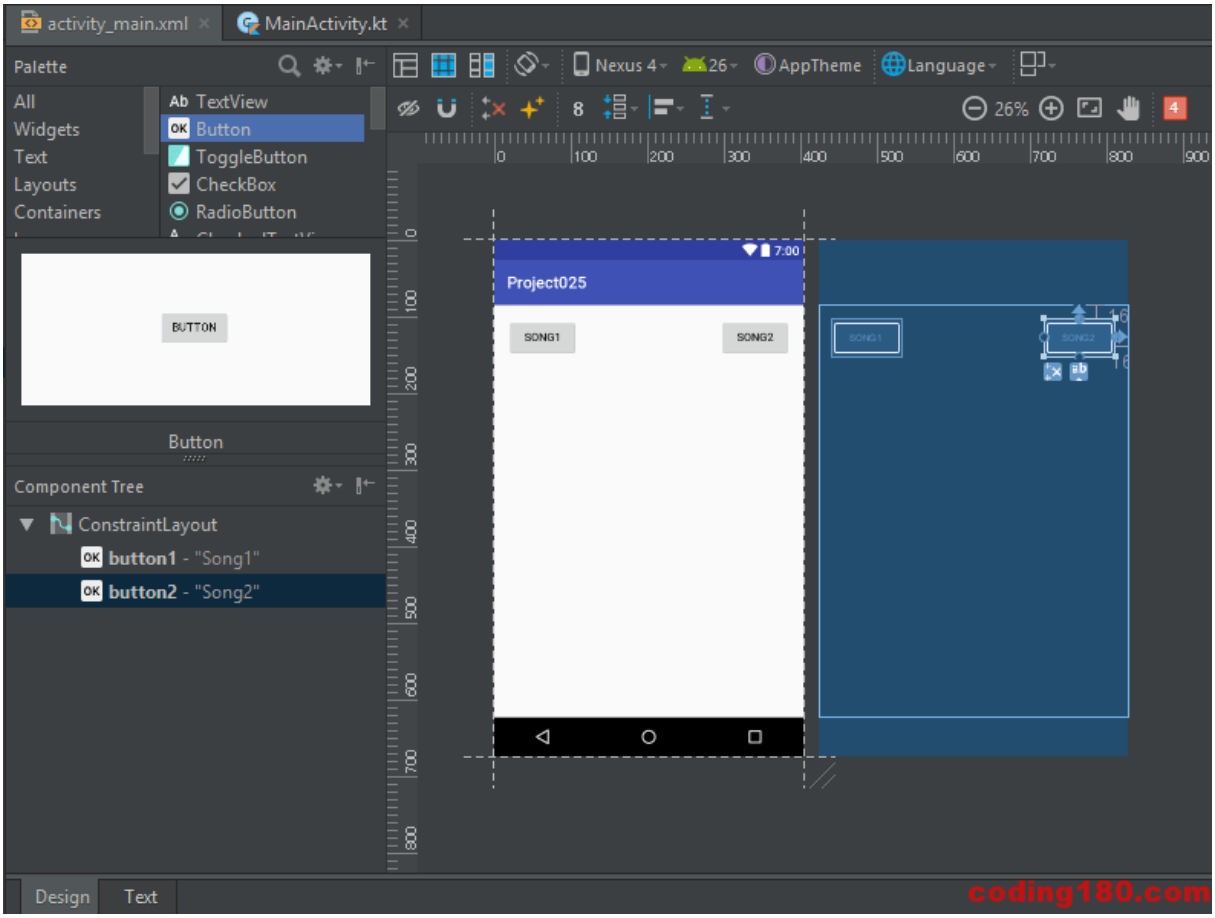
After creating the project we proceed to create a folder called raw that depends on the folder res, we store the two mp3 files in that folder (to create the folder we press the right mouse button on the folder res and select New -> Directory):



Then copy the files to the folder (in Android Studio Copy / Paste works from the file manager of the Windows operating system):



We create an interface with two buttons and initialize the properties text:



The HINT CODE is:

```
package com.coding180.project025


import android.media.MediaPlayer
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity: AppCompatActivity () {

    override fun onCreate (savedInstanceState: Bundle?) {
        super.onCreate (savedInstanceState)
        setContentView (R.layout.activity_main)
// coding180.com

        val button1 = findViewById (R.id.button1) as Button
        button1.setOnClickListener {
            val mp = MediaPlayer.create (this, R.raw.song1)
            mp.start ()
        }

        val button2 = findViewById (R.id.button2) as Button
        button2.setOnClickListener {
            val mp = MediaPlayer.create (this, R.raw.song2)
            mp.start ()
        }
    }
}
```

When we copy the mp3 files then the reference to the two files is generated in the R class and later we can rescue them when we create an object of the **MediaPlayer** class:

```
val mp = MediaPlayer.create (this, R.raw.song1)
```

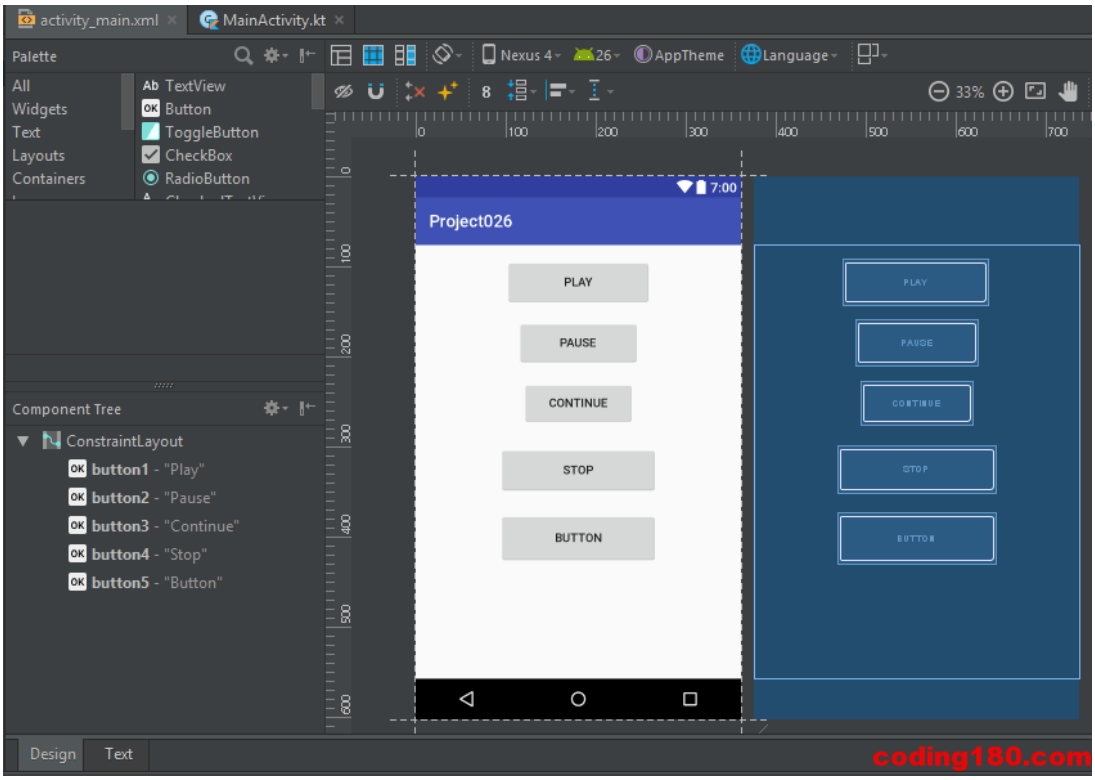
Next, we call the start method:

```
mp.start ()
```

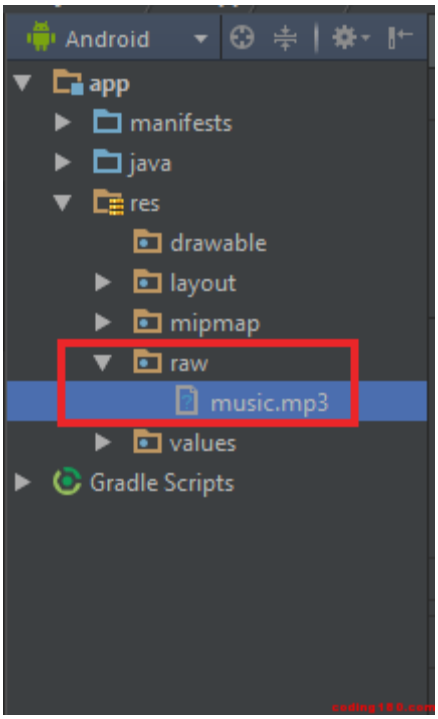
2-Insert audio (to play, stop, continue, permanently stop and not playback circularly)in Android Studio App

Create an application that allows you to start an mp3 file, stop, continue, permanently stop and activate or not playback in a circular way.

First, create a project and define the **5** respective buttons:



We create the **raw folder** and store in it the mp3 file previously created:



The HINT CODE is:

```
package com.coding180.project025


import android.media.MediaPlayer
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity: AppCompatActivity () {



    private lateinit var mp: MediaPlayer

    override fun onCreate (savedInstanceState: Bundle?) {
        super.onCreate (savedInstanceState)
        setContentView (R.layout.activity_main)

        mp = MediaPlayer.create (this, R.raw.music)
        var position = 0

        val button1 = findViewById (R.id.button1) as Button
        val button2 = findViewById (R.id.button2) as Button
        val button3 = findViewById (R.id.button3) as Button
        val button4 = findViewById (R.id.button4) as Button
        val button5 = findViewById (R.id.button5) as Button
```

```
        button1.setOnClickListener {
            mp.start ()
            if (button5.text == "Do not play in a circular way")
                mp.isLooping = false
            else
                mp.isLooping = true
        }

        button2.setOnClickListener {
            if (mp.isPlaying ()) {
                position = mp.getCurrentPosition ()
                mp.pause ()
            }
        }

        button3.setOnClickListener {
            if (mp.isPlaying () == false) {
                mp.seekTo (position)
                mp.start ()
            }
        }

        button4.setOnClickListener {
            mp.pause ()
            position = 0
            mp.seekTo (0)
        }

        button5.setOnClickListener {
            if (button5.text == "Do not play in a circular way")
                button5.setText ("Play in circular form")
            else
                button5.setText ("Do not play in circular form")
        }
    }

    override fun onDestroy () {
        super.onDestroy ()
        mp.release ()
    }
}
```

We define a property of the **MediaPlayer** class and by means of the lateinit keyword we indicate that it is a late start property (in the onCreate method it is loaded):

```
private lateinit var mp: MediaPlayer
```

The variable **mp** must be defined as property because we will access it in two methods (**onCreate** and **onDestroy**)

First, in the onCreate method we create the MediaPlayer class object to manage the mp3 file, and define an integer where the current playback position is stored in milliseconds (to be able to continue in the future).

To create an object of the MediaPlayer class by calling create method (in this we refer to the file that we copy to the raw folder)

```
mp = MediaPlayer.create (this, R.raw.music)
var position = 0
```

In the lambda of the button1, we call the method start of the MediaPlayer with which it begins to reproduce the sound file, we also determine if the sound should be executed in circular form depending on the text that has the button 5:

```
button1.setOnClickListener {
    mp.start ()
    if (botton5.text == "Do not play in a circular way")
        mp.isLooping = false
    else
        mp.isLooping = true
}
```

The **second** button checks if the sound is playing, if so we retrieve the current playback position and then call the pause method:

```
button2.setOnClickListener {
    if (mp.isPlaying ()) {
        position = mp.getCurrentPosition ()
        mp.pause ()
    }
}
```

The **third** button verifies that the object of the MediaPlayer class is paused and proceed to position in which millisecond continue the reproduction:

```
button3.setOnClickListener {
    if (mp.isPlaying () == false) {
        mp.seekTo (position)
        mp.start ()
    }
}
```

The **fourth** button stops playing the mp3 and initializes the position attribute with zero:

```
button4.setOnClickListener {
    mp.pause ()
    position = 0
```

```
            mp.seekTo (0)
        }
```

When the **fifth** button is pressed it changes if the reproduction is done in circular form or not proceeding to extract the text of the button and according to this value we store the opposite value:

```
        button5.setOnClickListener {
            if (button5.text == "Do not play in a circular way")
                button5.setText ("Play in circular form")
            else
                button5.setText ("Do not play in circular form")
        }
```

The onDestroy method is executed when the application is closed and in our case, we call the release method of the MediaPlayer class to free the resources:

```
    override fun onDestroy () {
        super.onDestroy ()
        mp.release ()
    }
```