```
package com.internshala.echo
import android.Manifest
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.support.v4.app.ActivityCompat
import android.widget.Toast
class SplashActivity : AppCompatActivity() {

    /*The SplashActivity file inherits the AppCompatActivity. The AppCompatActivity is the
base activity
    * on which every android application's activity is made upon. When developing Android
applications
    * that support multiple android devices running on different versions (Oreo, Nougat,
Marshmallow, Lollipop, Kitkat, etc.),
    * we want a standard way to provide newer features on earlier versions of Android
    * or gracefully fall back to equivalent functionality. This AppCompatActivity is a
version of such support library
    * which provides us with that.*/
    /*The Manifest class contains the different permissions used in Android. Let's
understand what each permission is used for.*/
    var permissionsString = arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE, /*This
allows the application to read from external storage.
                                                                                    We
need this to read the songs on your device*/
            Manifest.permission.MODIFY_AUDIO_SETTINGS, /*Allows an application to modify
the global audio settings. This helps  us in turning the
                                                        volume up or down when music
is playing*/
            Manifest.permission.READ_PHONE_STATE, /*This gives the application access to
get the cellular network information, phone state
                                                    including the phone number, the
status of any ongoing calls, and a list of any
                                                    phone accounts registered on the
device. This checks whether the phone is active or in flight mode.*/
            Manifest.permission.PROCESS_OUTGOING_CALLS, /*Allows the application to see
the number dialed during an outgoing call and gives it
                                                        an option to redirect or
abort the call. Helps us in checking when a call is made
                                                        so that we can pause the
media player*/
            Manifest.permission.RECORD_AUDIO) /*This only allows the application to record
the audio. This helps the visualizer to sync with the music*/
    override fun onCreate(savedInstanceState: Bundle?) {
        /*Mandatory method of Android activity life-cycle.
        * The saved instance state parameter is used to save the state of the activity when
the
        * activity is launched for the second time and onwards.*/
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)
        /*The setContentView() method is used to set the view to the activity.
        * Now, you will be wondering what is this R file and why are we doing R.layout
before mentioning the name of the activity.
        * This is done so that the compiler understands as to which file is to be displayed
as the view to this activity.
        * The R file is the auto-generated file which contains the id's for each file
present in our Android project.
        * The layout directory is present inside the R file and the activity_splash is
present inside layout directory.
        * Hence we use R.layout.activity_splash*/
        /*Here we check if the user has granted the permissions to the activity or not. If
yes, then we want to execute the else block
        * otherwise we execute the if block*/
```

```
        if (!hasPermissions(this@SplashActivity, *permissionsString)) {
            /*Now wkhen the permissions were not granted, we want our application to ask
for permissions. This is done with the help of
            * the lines of code written below*/
            ActivityCompat.requestPermissions(this@SplashActivity, permissionsString,
131)/*We request the user for permissions by

passing the array of permissions.

The request code is the unique number,

with which the Android OS will identify

which request was fired. We used 131,

you can use any other distinct number*/
        } else {
            /*Let's understand this in an easy way. The handler is used to handle the
tasks i.e. it is used to delay the tasks which need to be
            * performed. Here we are delaying the opening of the next activity by 1000ms
(1 sec) in order to make the welcome/splash screen visible
            * to the user and give our app a nice user experience(known as the UX)*/
            Handler().postDelayed({
                val startAct = Intent(this@SplashActivity, MainActivity::class.java)/*As
we already know this is used to define the

path of navigation from one activity to another*/
                startActivity(startAct) /*This statement is used to launch the new
activity*/
                this.finish() /*Now, this statement is used to finish the current activity
when the user moves on to the next activity.
                                    This prevents the user to view the splash again by
pressing back button, which won't be a good UX*/
            }, 1000)
        }
    }
    /*This is the method called when the user has completed the actions to be taken when
the permissions were asked for. Now, its time to check the
    * result of the permissions request. This was eatlier not present till the android
version 6.0.0(Marshmallow) or API 23*/
    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out
String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        when (requestCode) {/*Here we check the request code of the permission request made
by the user*/
            131 -> {/*Now remember that number specified here should match the request
code sent when the permission was requested. In our case
                        it was 131*/
                if (grantResults.isNotEmpty() /*Check whether the results which came are
not empty*/
                        /*The remaining lines check whether all the 5 permissions are
granted or not. The && is the logical AND operator which
                        * returns true only if all the statements inside the if block
return true. The complete condition becomes false if
                        * any of the condition return false*/
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED
                        && grantResults[1] == PackageManager.PERMISSION_GRANTED
                        && grantResults[2] == PackageManager.PERMISSION_GRANTED
                        && grantResults[3] == PackageManager.PERMISSION_GRANTED
                        && grantResults[4] == PackageManager.PERMISSION_GRANTED
                ) {
                    /*This is the same as above. As all permissions were granted we
launch the next activity*/
                    Handler().postDelayed({
```

```
                        val startAct = Intent(this@SplashActivity,
MainActivity::class.java)
                        startActivity(startAct)
                        this.finish()
                    }, 1000)
                    /*If in case any permission was denied by the user the application
exits giving a message to the user that, you haven't
                    *granted all the permission and this application needs them in order
to continue*/
                } else {
                    /*The toast message is a small prompt which appears at the bottom of
the screen*/
                    Toast.makeText(this@SplashActivity, "Please grant all the permissions
to continue.", Toast.LENGTH_SHORT).show()
                    this.finish()
                }
                /*Here we add the return statement to prevent the further execution of the
code. The control returns outside of this function
                * and now application has nowhere to go. Hence after the execution of if-
else nothing happens i.e either new activity
                * launches or the applicaiton exits but does not remain running in the
background thus consuming the resources*/
                return
            }
            else -> {
                /*In cases where something with the Android OS goes wrong, we want our
application to exit thus telling the user that
                * somethings has gone wrong. In such cases try re-opening the
application*/
                Toast.makeText(this@SplashActivity, "Something went wrong",
Toast.LENGTH_SHORT).show()
                this.finish()
                return
            }
        }
    }
    /*As the name suggests, this method checks whether the user has granted the permissions
asked by the application*/
    fun hasPermissions(context: Context, vararg permissions: String): Boolean { /*This
functions takes two params namely the context of the

application and the array of permissions */
        var hasAllPermissions = true /*This variable is used as a flag for checking the
status of permissions*/
        for (permission in permissions) { /*for loop to check for every single permission*/
            val res = context.checkCallingOrSelfPermission(permission) /*The method is
used to determine whether the user is granted
                                                                                    a
permission or not*/
            if (res != PackageManager.PERMISSION_GRANTED) {
                hasAllPermissions = false /*If the permission has not been granted we
change the value of hasAllPermission to false
                                                    else it remains true*/
            }
        }
        return hasAllPermissions /*The return statement returns the status of all the
permissions*/
    }
}
```