```kotlin
package com.internshala.echo.activities
import android.os.Bundle
import android.support.v4.widget.DrawerLayout
import android.support.v7.app.ActionBarDrawerToggle
import android.support.v7.app.AppCompatActivity
import android.support.v7.widget.Toolbar
import com.internshala.echo.fragments.MainScreenFragment
import com.internshala.echo.R

class MainActivity : AppCompatActivity() {

    /*The line below is used to define the variable for drawer layout*/
    var drawerLayout: DrawerLayout? = null

    /*While choosing onCreate() method, you might see two options i.e.
    * override fun onCreate(savedInstanceState: Bundle?, persistentState:
PersistableBundle?) {
        super.onCreate(savedInstanceState, persistentState)
    }
    * and the other one which we chose. You also choose the one we chose as the other one
is not the one which is included in the lifecycle method*/
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        /*Here we define the variable for toolbar and by using findViewById, we told our
kotlin file that the toolbar which we made
        * is present with the id as toolbar. This id is also present in the R file.*/
        val toolbar = findViewById<Toolbar>(R.id.toolbar)

        /*Now the code below is used to tell the compiler that we would be doing some
actions on our toolbar so therefore put our toolbar
        * in the form of Action bar which can support multiple actions thus making it
interactive*/
        setSupportActionBar(toolbar)

        /*Here we did the similar thing which we did for toolbar earlier, using the
findViewById() and telling the compiler that we will use the drawer layout which
        * is present in our XML with the id "drawer_layout"*/
        drawerLayout = findViewById(R.id.drawer_layout)

        /*The icon having 3 parallel horizontal lines is known as the action bar drawer
toggle. The 4 parameters taken by the this can be understood as the:
        * 1. Activity: In which activity the toggle is placed
        * 2. Drawer Layout: For which drawer layout we are placing this toggle
        * 3. Toolbar: In which toolbar are we placing it
        * 4, 5: Action Commands: The function of drawer when toggle is click, which is open
and close the drawer*/
        val toggle = ActionBarDrawerToggle(this@MainActivity, drawerLayout, toolbar,
R.string.navigation_drawer_open, R.string.navigation_drawer_close)

        /*Earlier we only defined the toggle, now we place the listener using the
addDrawerListener() method
        * Note: In the videos, we told to use setDrawerListener() but now this method cuts
itself when we use it which means that this method is now deprecated which
        * can be removed from the future versions, hence here we have told the latest
version of the method i.e. addDrawerListener(). Although as of now the usage
        * setDrawerListener() is also correct*/
        drawerLayout?.addDrawerListener(toggle)


        /*When we click the toggle icon, we can see the shape of the icon changing its
shape. This may only be visible for a small amount of time as our drawer covers it up.
        * But it does and this change happens because we sync the state of the toggle
accordingly using the function syncState()*/
```

```
        toggle.syncState()

        /*Creating an object of the MainScreenFragment*/
        val mainScreenFragment = MainScreenFragment()

        /*Since our app will contain different fragments and we want to display main screen
fragment whn the app launches, hence we add main screen fragment to the details fragment*/
        this.supportFragmentManager /*the support fragment manager helps us interact with
the fragment in the activity*/
                .beginTransaction() /*This is used for starting a series of functions
which are handled by fragment manager*/
                .add(R.id.details_fragment, mainScreenFragment, "MainScreenFragment")
/*Adding a new fragment in place of the details fragment*/
                .commit() /*This used to save the changes made to the app and the main
screen fragment is made visible to the user*/
    }

    override fun onStart() {
        super.onStart()
    }
}
```