```
package com.internshala.echo.adapters
import android.content.Context
import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.RelativeLayout
import android.widget.TextView
import com.internshala.echo.R
import com.internshala.echo.activities.MainActivity
import com.internshala.echo.fragments.AboutUsFragment
import com.internshala.echo.fragments.FavoriteFragment
import com.internshala.echo.fragments.MainScreenFragment
import com.internshala.echo.fragments.SettingsFragment
/**
 * Created by Harsh Deep Singh on 2/12/2018.
 */

/*This is the adapter class, which is used to set the views indise the recycler views. This
class acts as bridge between the view and its data.
* The parameters used in the class are the list for the names of the items, images for it
and the context for the Adapter respectively*/
class NavigationDrawerAdapter(_contentList: ArrayList<String>, _getImages: IntArray,
_context: Context)
    : RecyclerView.Adapter<NavigationDrawerAdapter.NavViewHolder>() {

    /*Declaring the variables used*/
    var contentList: ArrayList<String>? = null
    var getImages: IntArray? = null
    var mContext: Context? = null

    /*This is the constructor initialisation of the parameters. This converts the data
passed from the parameters as the local params, which are used in this class*/
    init {
        this.contentList = _contentList
        this.getImages = _getImages
        this.mContext = _context
    }

    /*The onBindViewHolder() method is used to display the data at the specified position.
    * The params i.e. holder and position are used to set the data and the position of that
data within the recycler view*/
    override fun onBindViewHolder(holder: NavViewHolder?, position: Int) {

        /*Here we set the icon and the name of that icon with the setBackgroundResource()
and the setText() method respectively*/
        holder?.icon_GET?.setBackgroundResource(getImages?.get(position) as Int)
        holder?.text_GET?.setText(contentList?.get(position))

        /*Now since we want to open a new fragment at the click for every item we place the
click listener according to the position of the items*/
        holder?.contentHolder?.setOnClickListener({

            /*Loading the Main Screen Fragment as the first(remember that the index starts
at 0) item is All songs and the fragment corresponding to it is the Main Screen fragment*/
            if (position == 0) {
                val mainScreenFragment = MainScreenFragment()
                (mContext as MainActivity).supportFragmentManager
                        .beginTransaction()
                        .replace(R.id.details_fragment, mainScreenFragment)
                        .commit()
            }
```

```kotlin
            /*The next item is the Favorites option and the fragment corresponding to it
is the favorite fragment at position 1*/
            else if (position == 1) {
                val favoriteFragment = FavoriteFragment()
                (mContext as MainActivity).supportFragmentManager
                        .beginTransaction()
                        .replace(R.id.details_fragment, favoriteFragment)
                        .commit()
            }

            /*Similarly to the above we load the Settings and the About Us fragment
respectively*/
            else if (position == 2) {
                val settingsFragment = SettingsFragment()
                (mContext as MainActivity).supportFragmentManager
                        .beginTransaction()
                        .replace(R.id.details_fragment, settingsFragment)
                        .commit()
            } else if (position == 3) {
                val aboutUsFragment = AboutUsFragment()
                (mContext as MainActivity).supportFragmentManager
                        .beginTransaction()
                        .replace(R.id.details_fragment, aboutUsFragment)
                        .commit()
            }

            /*As we tap on any item we want our drawer to close automatically as the
fragment loads. The function closeDrawers() is used for doing the same
            * Note here that we have used the drawer layout in the exact similar way we
did in our MainActivity as MainActivity.Statified.drawerLayout.
            * This is because we created an object of it and hence it can be used in a
similar way anywhere in our project*/
            MainActivity.Statified.drawerLayout?.closeDrawers()
        })
    }

    /*This function is used to create the view for the single row of the recycler view. We
inflate the view used for single row inside this method.
    * Let's discuss the params of this method:
    * i) parent: ViewGroup? ->  The view group is the base class for layouts and views
containers. Here the parent is the view group into which the new view will be added
    * ii) viewType: Int -> The type of the view to be inflated*/
    override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): NavViewHolder {

        /*Here we inflate our view taking the context from the parent. The inflate()
function takes the resource(R.layout.row_custom_navigationdrawer)
        * sets it to the parent and does not attach this to the root. You can skip the
details of this as of now*/
        val itemView = LayoutInflater.from(parent?.context)
                .inflate(R.layout.row_custom_navigationdrawer, parent, false)

        /*Here we pass this view into the holder and return that and our view is created.
The below tow lines can be reduced as
        * return NavViewHolder(itemView)*/
        val returnThis = NavViewHolder(itemView)
        return returnThis
    }


    /*This function returns the number elements present in our recycler view. The number of
these items can be calculated by the number of items in our arraylist(contentList)*/
    override fun getItemCount(): Int {

        /*Here we return the size of the list we used.*/
        return (contentList as ArrayList).size
    }

    /*Class for creating a view holder for our recycler view. This class sets up the single
object for our recycler view*/
    class NavViewHolder(itemView: View?) : RecyclerView.ViewHolder(itemView) {

        /*Declaring the widgets and the layout used*/
        var icon_GET: ImageView? = null
        var text_GET: TextView? = null
        var contentHolder: RelativeLayout? = null

        /*Constructor initialisation for the variables*/
        init {
            icon_GET = itemView?.findViewById(R.id.icon_navdrawer)
            text_GET = itemView?.findViewById(R.id.text_navdrawer)
            contentHolder = itemView?.findViewById(R.id.navdrawer_item_content_holder)
        }
    }
}
```