Exercise w1d2 JetPack Compose list or RecyclerView Fragments

Create an application which shows a list of presidents. By tapping on a president name (on the list), a detailed view with details (e.g. name, start duty, end duty and description) on that particular president is shown on the new screen.

This exercise can be implemented **either** using JetPack Compose **or** RecyclerView and Fragments. You can use the following Singleton for the president data:

```
/* Singleton consept obtained from here
https://medium.com/@adinugroho/singleton-in-kotlin-better-approach-8c5e28a140a5 */
object DataProvider {
   val presidents: kotlin.collections.MutableList<President> = java.util.ArrayList()

   init {
       Log.d("USR", "This ($this) is a singleton")

       // construct the data source
       presidents.add(President("Kaarlo Stahlberg", 1919, 1925, "Eka presidentti"))
       presidents.add(President("Lauri Relander", 1925, 1931, "Toka presidentti"))
       presidents.add(President("P. E. Svinhufvud", 1931, 1937, "Kolmas presidentti"))
       presidents.add(President("Kyösti Kallio", 1937, 1940, "Neljas presidentti"))
       presidents.add(President("Risto Ryti", 1940, 1944, "Viides presidentti"))
       presidents.add(President("Carl Gustaf Emil Mannerheim", 1944, 1946, "Kuudes
presidentti"))
       presidents.add(President("Juho Kusti Paasikivi", 1946, 1956, "Äkäinen ukko"))
       presidents.add(President("Urho Kekkonen", 1956, 1982, "Pelimies"))
       presidents.add(President("Mauno Koivisto", 1982, 1994, "Manu"))
       presidents.add(President("Martti Ahtisaari", 1994, 2000, "Mahtisaari"))
       presidents.add(President("Tarja Halonen", 2000, 2012, "Eka naispresidentti"))
       presidents.add(President("Sauli Niinistö", 2012, 2024, "Ensimmäisen koiran, Oskun,
omistaja"))
       presidents.sort()
   }
}
```

JetPack Compose implementation

Use navController for an easy switching between those two separate composables.

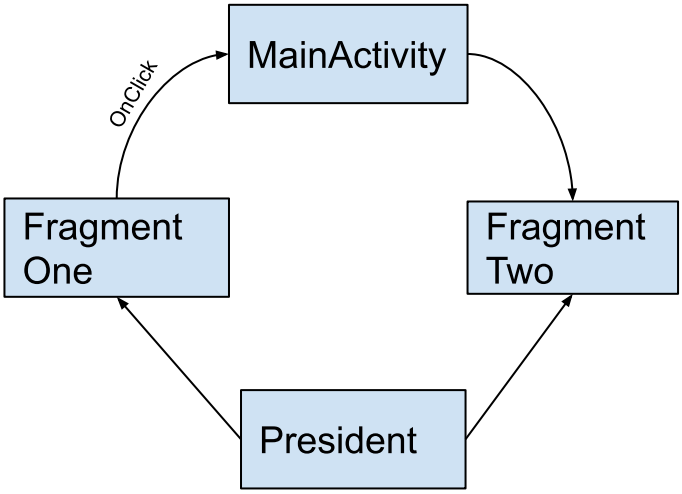
Hint 1: remember to add androidx.navigation:navigation-compose dependency to the modules build gradle.

Hint 2: It is easiest to pass strings as parameters between composables. To find a matching object from the ArrayList, use Kotlin find method, e.g. `DataProvider.presidents.find {it.name == name}`

RecyclerView and Fragment implementation

This list is implemented using a simple RecyclerView. President list should be in one fragment. Detailed presidential information is shown on the second fragment.

Use back stack for fragments where appropriate.



Hint 1: Direct fragmentManager is deprecated since Android API 27 (and thus a lot of Internet example code is obsolete). Today we use AndroidX Fragment library, so remember to add the following lines to your **module** build.gradle:

```
dependencies {
    def fragment_version = "1.3.6"
    implementation "androidx.fragment:fragment-ktx:$fragment_version"
                .
                .
                .
```

Hint 2: You main layout can contain only the place for the fragment, like

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

and in order to put a fragment on the screen, you can use fragment manager via Android Kotlin Extensions (Android KTX) as

```
supportFragmentManager.commit {
    setReorderingAllowed(true)
    add<FragmentOne>(R.id.fragmentContainerView) // this is same than
add(R.id.fragmentContainerView, FragmentOne())
}
```

Hint 3: Communication between fragments can be done using Bundles (here the index of the selected president is passed using a key named pos) and fragment manager can help when switching to the other fragment is needed. Using Android KTX, this becomes very easy, like

```
val bundle = bundleOf("pos" to position)
supportFragmentManager.commit {
    setReorderingAllowed(true)
    replace<FragmentTwo>(R.id.fragmentContainerView, args = bundle)
    addToBackStack(null)
}
```

Hint 4: Controlling user selections is a little tricky. FragmentOne does not know anything of the FragmentTwo, because of component isolation. All the communication should be done through the MainActivity. Therefore MainActivity should receive button click information from the FragmentOne. This can be done by implementing an interface from the FragmentOne in MainActivity, like

```
class MainActivity : AppCompatActivity(R.layout.activity_main),
FragmentOne.FragmentOneListener {
        .
        .
        .
   override fun onButtonClick(position: Int) {
       Log.d("USR", "MainActivity received $position")
        .
        .
        .
}
```

and the interface should be defined in FragmentOne like

```
class FragmentOne : Fragment(R.layout.fragment_one) {
   internal var activityCallBack: FragmentOneListener? = null

   interface FragmentOneListener {
       fun onButtonClick(position: Int)
   }

   override fun onAttach(context: Context) {
       super.onAttach(context)

       Log.d("USR", "onAttach received!!!")
       activityCallBack = context as FragmentOneListener
   }
        .
        .
        .
```

now when the MainActivity creates the FragmentOne, his object reference (context) is passed to the FragmentOne. And because MainActivity implements FragmentOneListener, FragmentOne knows that there is `onButtonClick()` method that can be called when the user taps the president. When you need to inform MainActivity from the FragmentOne that the button has been pressed, just call

```
activityCallBack!!.onButtonClick(position)
```

Hint 5: With a new AndroidX Fragment library, there is no need to “inflate” the fragment you anymore. Just give the parameter to the constructor, like

```
class FragmentTwo : Fragment(R.layout.fragment_two) {
   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
       val position = requireArguments().getInt("pos")
       val president = GlobalModel.presidents[position];

       Log.d("USR", "Fragment 2, president $position")

       view.findViewById<TextView>(R.id.PresidentName).text = president.name
       view.findViewById<TextView>(R.id.PresidentDescription).text = president.description
   }
}
```

the new callback function, `onViewCreated()`, is called when the fragment has been “inflated”.

Hint 6: For the recyclerview, use just the code given at the lecture slides and add the callback call when item is clicked on the list.