# Fragments and ViewPager2

Shingai Zindi

## Fragments

### Basics

Fragments are basically mini-Activities that are contained in View Containers. They are useful because they can be added, swapped and removed from the user interface using Fragment Managers and Fragment Transactions. They are also useful because they can aid in building a responsive user interface.

### Lifecycles

The lifecycles of Activities and Fragments are similar but slightly different.

|  |  |
| --- | --- |
| **Activity Lifecycle** | **Fragment Lifecycle** |
| Android Activity Lifecycle - javatpoint | Android Fragments - javatpoint |

#### Notes

* onRestart() is missing from the Fragment lifecycle.
* The Fragment has additional methods to handle attaching and detaching it from its views. Handy methods are:
  + onCreateView() to inflate the fragment layout into its container.
  + onCreate() is similar to the same method in the activity lifecycle however this should not be used in the same way because onCreateView() comes after it in the life cycle.
  + onViewCreated() does not appear in the Fragment lifecycle but this is useful when finding views in the fragment view.
  + An empty constructor to create the Fragment
  + newInstance() is provided by Android Studio when a Blank Fragment is added to a project. This is useful when making multiple instances of the same fragment. I implemented this method but did not use it as my apps only use each fragment once.

### Migrating from Activities to Fragments

#### Simplified

If you plan on using static fragments that do not require complex swapping of fragments it can be very simple to migrate an activity to a fragment. In my example most of the work is done in the onStart() method. This method and any non-lifecycle methods can be copied over as is. ***Note that the onStart() method of fragments if public and not protected.***

#### Migrating onCreate()

The activity onCreate() method requires a little more work to migrate to a fragment. Two method must be implemented for the fragment to be functional. I opted to include a third method in order to simplify this process.

1. onCreateView()
   1. Because fragments are contained within a view their layouts must be inflated. This is where this process happens.
   2. If a blank view is created Android Studio will fully implement a simple version of this method for you.
   3. Similar methods we have used in class:
      1. getView() from Adapters
      2. setContent() from activates
2. onCreate()
   1. This method can be used to instantiate any variables needed for the app. Like how this would be used in activities.
   2. ***However, setting variables that require anything from the layout cannot be set here because this method is called before onCreateView() and thus the layout has not been inflated yet.***
3. onViewCreated()
   1. This method is called after onCreateView(). So, I opted to use this function to set any variable that require elements of the layout.
   2. ***Note findViewById() requires the getView() method in front of it.***
4. newInstance()
   1. AndroidStudio will include a newInstance() method when creating a blank fragment. This will aid in the creation of multiple instances of a fragment. I only needed to create each fragment once, so I didn’t use this method. I did implement it in PokedexFragment if interested.

I please compare the activities used in PokedexStarter to the corresponding fragments used in PokedexFragments. Note that the layouts used for each are identical.

#### Activities to Manage Static Fragments.

Fragments are contained in Container Views. If the layout is simple and fragments are only going to be set once, then the fragment view can be used. When this view is selected Android Studio will prompt for the fragment that will be set in this view.

I opted to use the Fragment Container view which is included in the navigation dependency. Doing so required that I set the fragment within the container.

#### Setting Fragments in containers

A Fragment Manger may be used to facilitate Fragment Transactions that can add, replace and set fragments into a container. Bundles may be added to the transaction in order to pass information when setting a fragment into a container. Bundles are just collections of key-value pairs that can be passed to fragments and activities. You may have used these when putting extras in intents.

Fragment Managers are used to control fragments from an activity. They may also be used to coordinate the transfer of information to and from fragments. I only required the setting of a fragment because I had already added my bundle of data when I created the fragment.

Fragment Managers control fragments through Fragment Transactions. I used the replace transaction though I could have used add because I only set each fragment once. **These are similar to transactions used in SQL in that they need to be begun (beginTransaction()) and ended (commit()).**

### Responsive UI/UX

Fragments are not quite needed for this part of the project, but it is helpful.

Layouts may be duplicated in a project if it is put in a layout folder with a slight modification of the folder that it resides in.

PokedexFragment uses this twice.

It has two folders in the resource folder to pay attention to if looking from the project view and not Android view. From Android view the layouts have a greyed-out portion at the end of the file name.

* Layout
* Layout-sw600dp

Two of the layouts in the app are in both folders

* activity\_abilities\_moves\_fragment
* activity\_pokemon\_fragment

The suffix on the layout folder tells the app the it should use the layouts in the layout-sw600dp if the device the app is running on has a screen is at least 600dp on its shortest side. This number was recommended by Android Developers as a suitable split between Phones and 7” Tablets.

A layout-w600dp should be used if you want the layout to change depended on the width of the device in its current orientation.

If PokedexFragment is opened on a tablet the fragments are displayed side by side. On phones they are displayed vertically.

This way of providing alternative layouts may also be done for menus.

***\*Make sure not to change the names of any views that are being altered programmatically as the app will act the same when dealing with both layouts, only the UI will change. More complex responsiveness will require programmatically checking for the screen and then coding differently for the intended situations.***

***\*If you do this make sure that the app can recover from changes in screen width if needed. For example, the app should be able to handle what happens when the screen is rotated.***

## ViewPager2

PokedexViewPager

### Basics

In simple terms ViewPager2 is a Recycler View for fragments.

Implementation requires a ViewPager2 view in a layout, and an adapter such a s Fragment State Adapter. Very similar to how Recycler Views and List Views require adapters.

A Fragment Stat Adapter should implement three methods. A constructor, createFragment() and getItemCount().

* getItemCount() will determine how many fragments will be set in the View Pager.
* createFragment() will handle the creation of fragments to be put in the View Pager.
* The constructor is to create the adapter.

### Creating Fragments in FragmentStateAdapter

Because I only had 4 fragments, they could easily be created based on what order they should appear. The View Pager will basically store them in a list of fragments. The getItemCount() could also be set to 4 because I only had 4 fragments.

\*Other adapters can be used to control the View Pager

### Putting it Together

The last step requires an activity that will create the View Pager and Create the Adapter and set the adapter to the View Pager.

### Tab Layout

Tab Layouts can be linked to a View Pager in order to provide quick navigation. When a Tab Layout is added to a layout it will include some tab items. These can be discarded because they can be set based on the View Pager it is attached to.

A Tab Mediator is used to link the Tab Layout and View Pager while simultaneously allowing the tab titles to be set.

Again, because I only had 4 static fragments, I could simply store the title in a list.