IMD 4008

Mobile User Interfaces: Design & Development

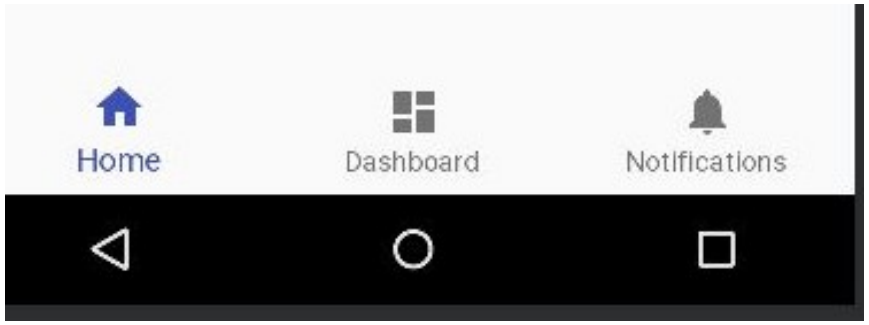
Week 5 – Navigation via Fragments

Objectives

- Look at some new Navigation controls
  - BottomNavigationView
- Investigate more options for multi-screen apps
  - Fragments

- Prep for the Tutorial 4 activity!

BottomNavigationView



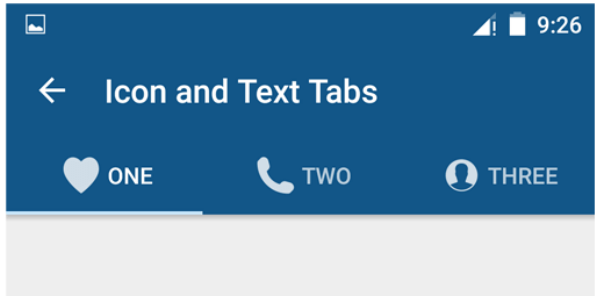

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/navigation"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="0dp"
    android:layout_marginStart="0dp"
    android:background="?android:attr/windowBackground"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:menu="@menu/navigation" />
```

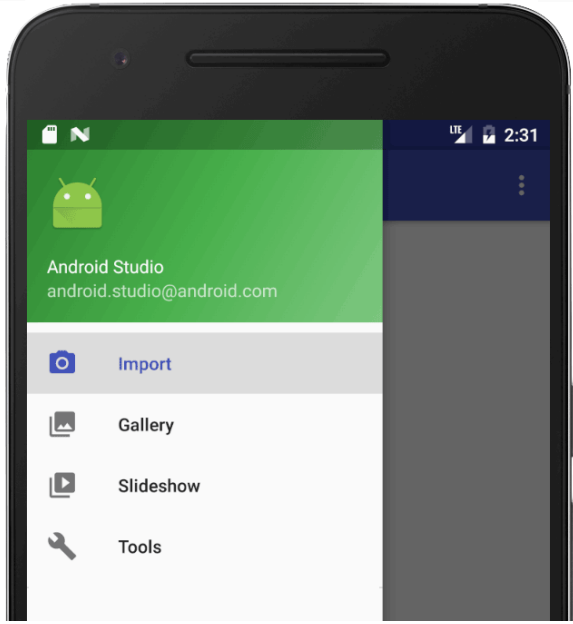
activity_main.xml | navigation.xml | app

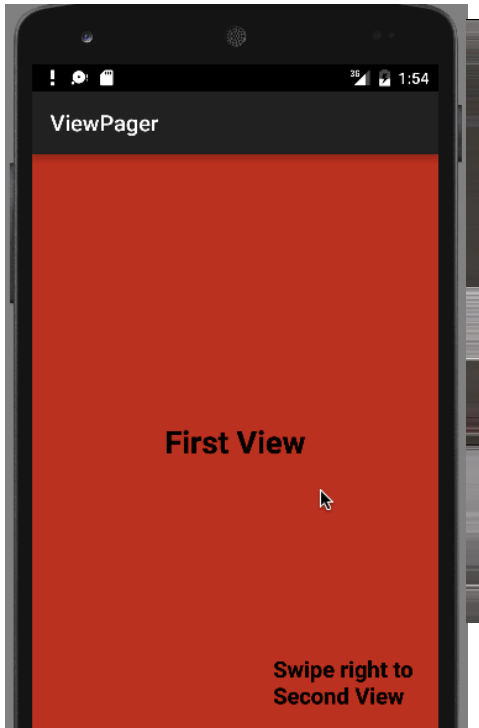
```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/navigation_home"
        android:icon="@drawable/ic_home_black_24dp"
        android:title="Home" />

    <item
        android:id="@+id/navigation_dashboard"
        android:icon="@drawable/ic_dashboard_black_24dp"
        android:title="Dashboard" />

    <item
        android:id="@+id/navigation_notifications"
        android:icon="@drawable/ic_notifications_black_24dp"
        android:title="Notifications" />

</menu>
```

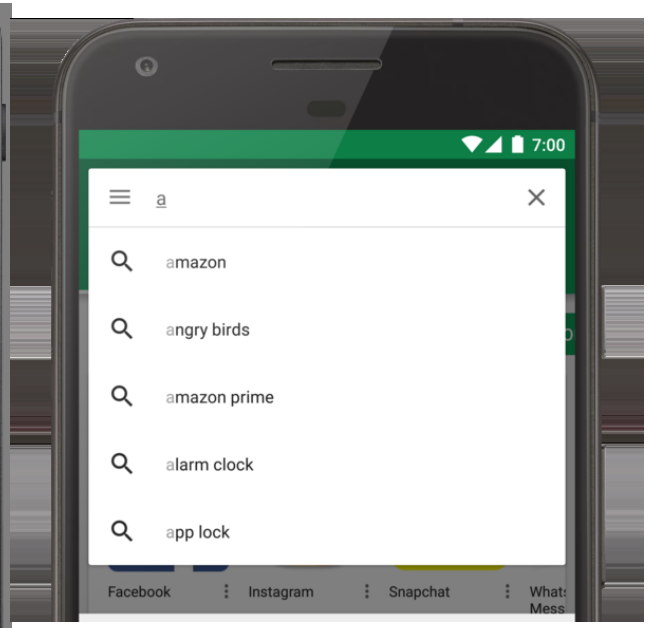
Other Common Navigation Views

- These are left as an exercise to you
- All can employ fragments, like the BottomNavigationView
- TabLayout (Reference)
  - A Tutorial
- SearchDialog (Google Tutorial)
- NavigationDrawer (Reference)
  - A tutorial
- ViewPager – slide between fragments
  - A tutorial



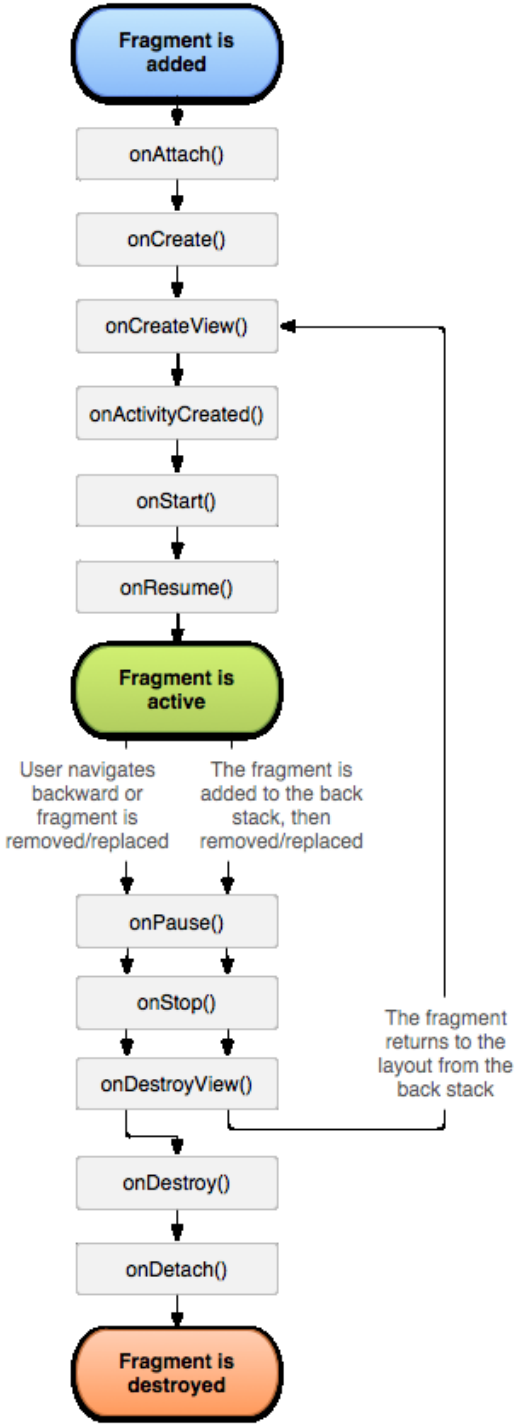






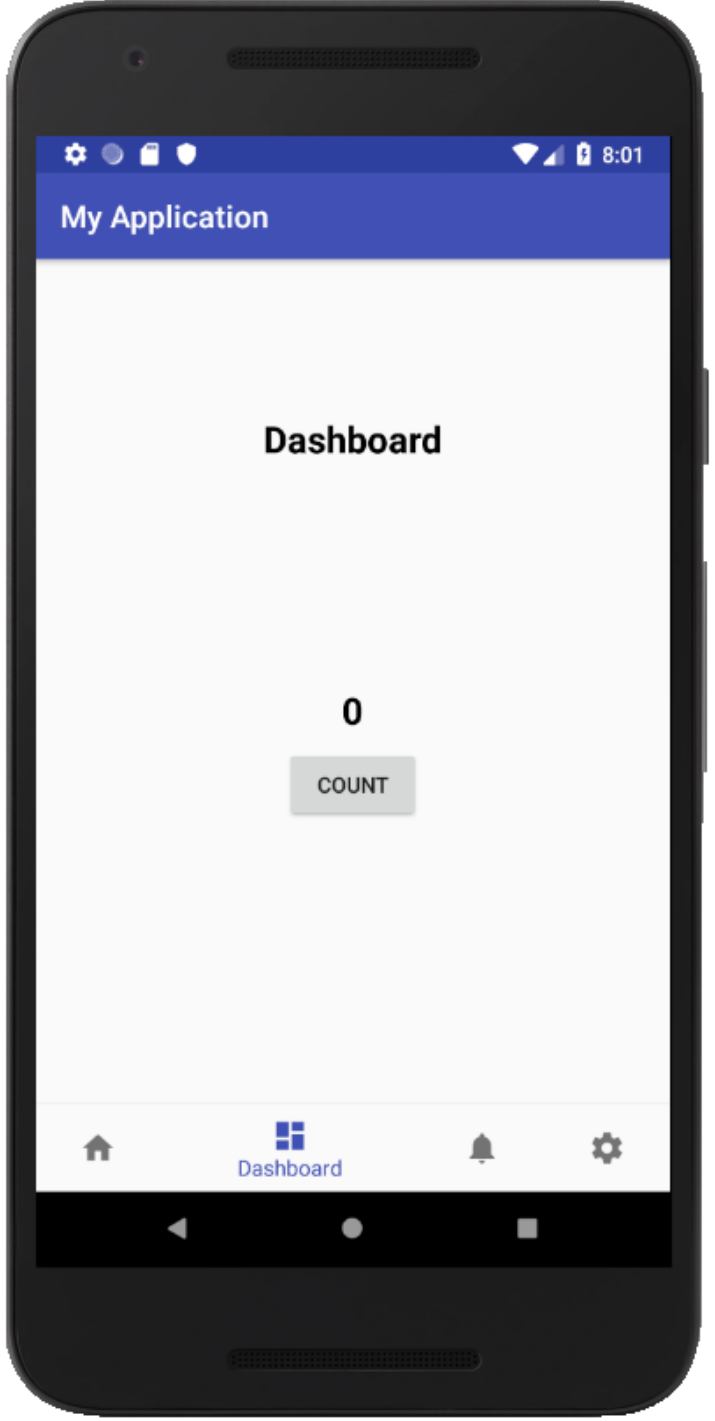
Fragments

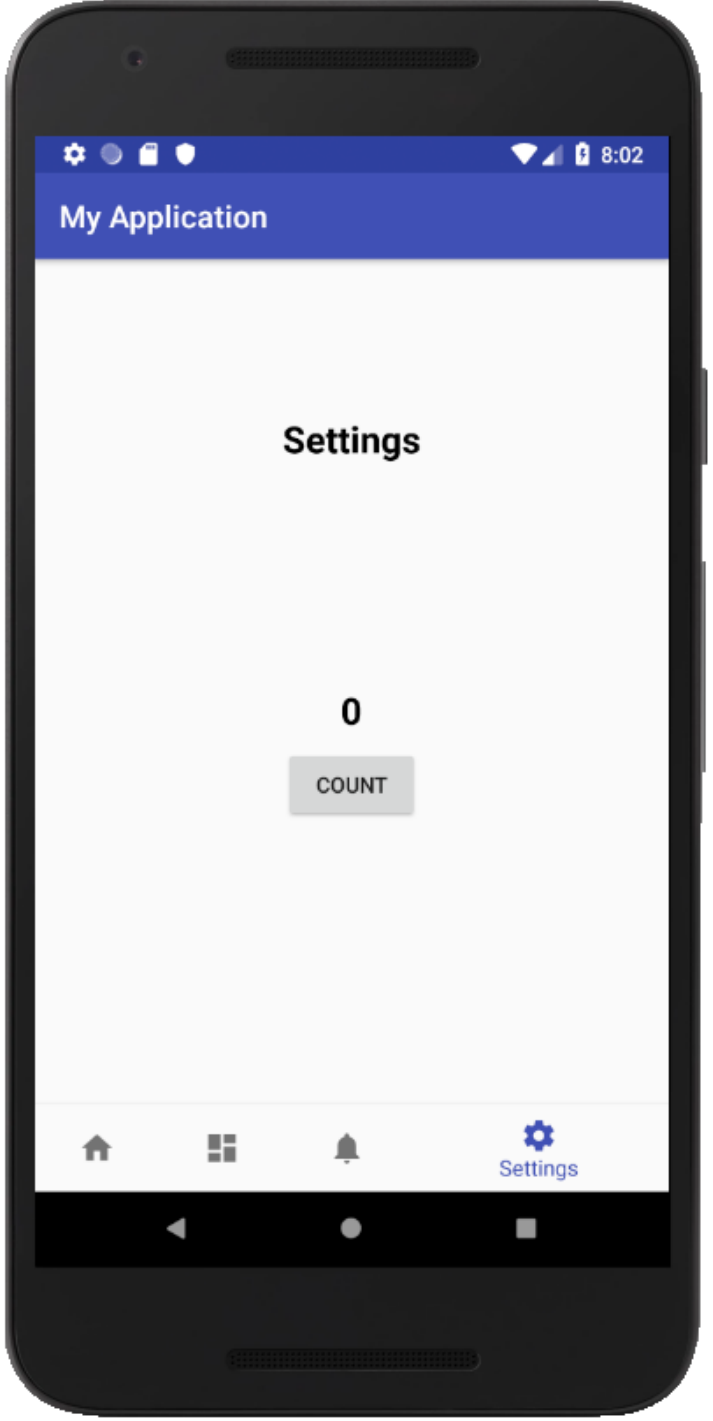
- Like a “sub-activity”
  - Support multiple screens in your app, while keeping other parts the same
  - Important for navigation!
- Modular section of an Activity that has its own lifecycle
  - Also affected by host’s lifecycle (e.g., when Activity paused or destroyed, so are all its fragments
- Receives its own input events
- Can be added/removed while the main Activity is running
  - Resides in a ViewGroup within your Activity layout



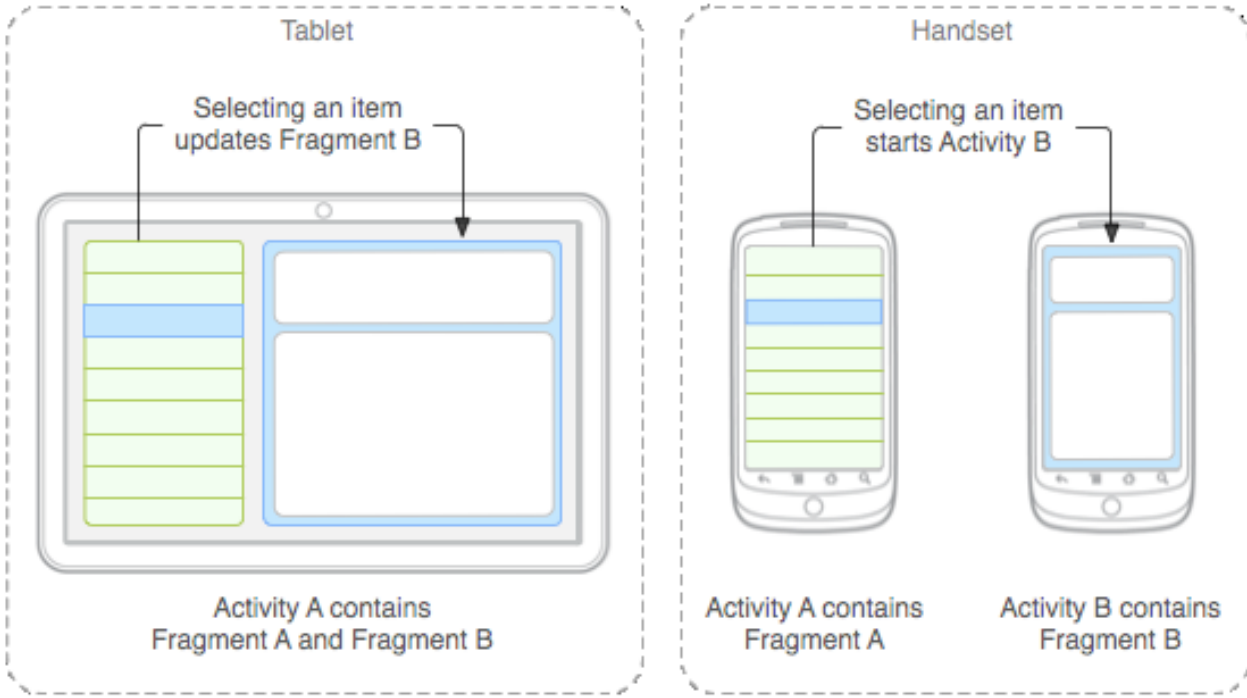
https://developer.android.com/guide/components/fragments


8:01
My Application
Home
0
COUNT
Home


8:01
My Application
Dashboard
0
COUNT
Dashboard


8:02
My Application
Settings
0
COUNT
Settings

Fragments (2)



- Have their <u>own layout and code</u>
  - Can be used in multiple activities
  - A single activity can have N fragments
- Use with FrameLayout
  - FrameLayout provides a container for fragments to go in
- LayoutInflater
  - Instantiates a layout XML file into its corresponding View objects
    - i.e., “inflates” XML layout code to be presented, allowing changing the currently presented layout

Fragment Example

In HomeFragment.java

```
public class HomeFragment extends Fragment {

    TextView counter;
    Button inc;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState)
    {
        view = inflater.inflate(R.layout.home_fragment, null);
               … etc …

        counter = view.findViewById(R.id.home_counter);
        inc = view.findViewById(R.id.home_button);

               …etc…
        return view;
    }
```

Fragment Entry Point (like onCreate seen with Activities)

Fragment Manager

- Handles your app's fragments
  - Acquire with getSupportFragmentManager
  - Example:

```
getSupportFragmentManager().beginTransaction()
                            .replace(R.id.fragment_container, frag)
                            .commit();
```

  - .beginTransaction starts a FragmentTransaction (such as add, replace, etc.)
  - .replace to change the current fragment for a different one in specified container (*note*: fragment_container is a FrameLayout, frag is the new Fragment)
  - .commit to schedule the FragmenTransaction to take place

Managing Fragments Example

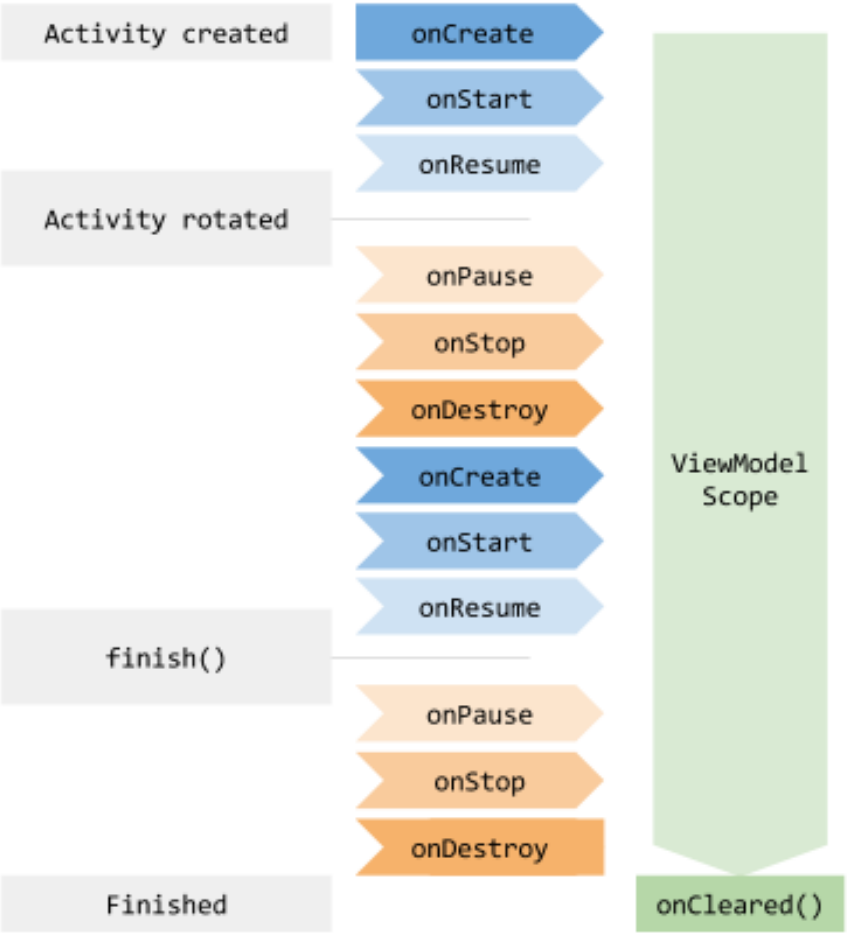
```
public class MainActivity extends Activity {

    private BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelectedListener
            = new BottomNavigationView.OnNavigationItemSelectedListener() {

        Fragment frag = null;
                .
                .
                .
        switch (item.getItemId()) {
        case R.id.navigation_home:
            frag = new HomeFragment();
            currentFragment = 0;
            break;
                .
                .
                .
getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, frag).commit();
```

In this switch statement, you would have a case for each possible navigation option, setting **frag** to the corresponding fragment.

This line of code (after the switch statement) actually causes the fragment_container to be replaced with the fragment set in the switch statement

Sharing Data Between Fragments

- Fragments, like Activities, have their own data scope, and lose their data when changed
- Solve with the ViewModel
  - Class responsible for preparing and managing data for an Activity or Fragment
    - Handles communication
    - Scope is same as creating object (e.g., Activity)
  - Classes extending the ViewModel can have data that persists through the usual things that cause data to be lost
    - E.g., onDestroy
  - Like a global variable
    - Can share data between Fragments
  - Unfortunately, quite arcane syntax…



ViewModel Example

- Declaring a ViewModel class

```
public class DataContainer extends ViewModel {

    int settingCounter = 0;
    … other variables you want to share go here …
}
```

ViewModel Example (2)

- Instantiating a ViewModel (requires ViewModelProviders)

```
final DataContainer dc = ViewModelProviders.of(this).get(DataContainer.class);
```

- .of creates ViewModelProvider, which retains ViewModels in scope of *this* Activity (instantiated in MainActivity)
- .get returns a ViewModel based on class provided
  - Provides an instance of that class that acts as a ViewModel, i.e., has scope within the entire Activity

ViewModel Example (3)

- To get/set data (e.g., an int called “counter”) in the ViewModel class:

```
count = ViewModelProviders.of((MainActivity) getActivity()).get(DataContainer.class).counter;

ViewModelProviders.of(MainActivity.this).get(DataContainer.class).counter = currentCount;
```

Summary

- Tutorial 4 involves:
  - Setting up a Navigation Bar
  - Setting up four Fragments with four layouts
    - (basically identical for the purpose of the tutorial)
  - Changing between Fragments
  - Storing data with a ViewModel
  - Accessing data from different Fragments
- Homework:
  - The usual ☺
    - (Go back through all the reference links in this presentation, and review them while working through Tutorial 3)