Android
Dynamic UI - Fragments

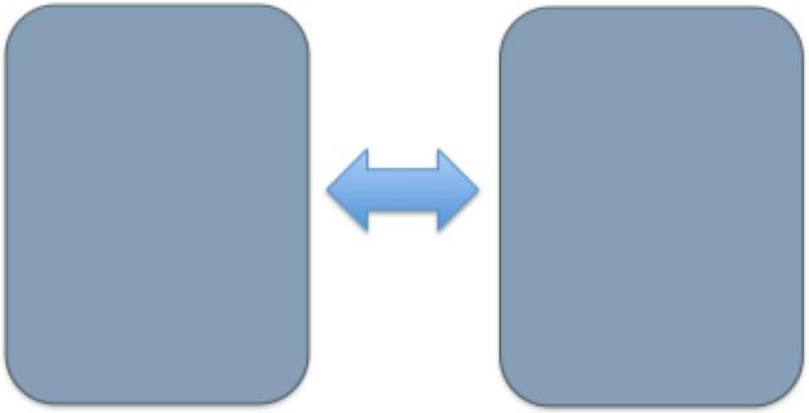
CSCI3310 Mobile Computing & Application Development

Overview

- Fragments in Activity
- Implementing Fragments
- Fragment lifecycle
- Back Stack for Fragment
- (Optional) Fragment Tag…

Days before Fragment

- An activity is a container for views, a screen to hold all UI views that the user interacts with
- In the beginning, if you wanted 2 different screens, the usual way is to have 2 activities

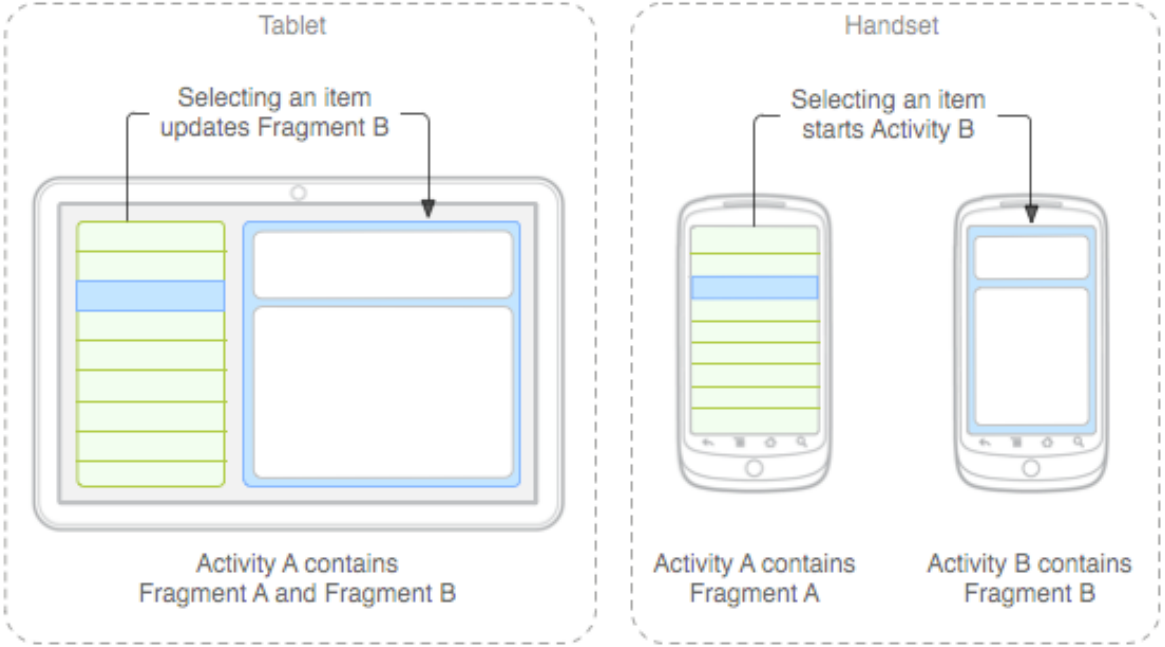


- When you have a larger screen device than a phone –like a tablet it can look too simple to use phone interface here.

Fragment Idea

Since Android 3.0, fragments are introduced

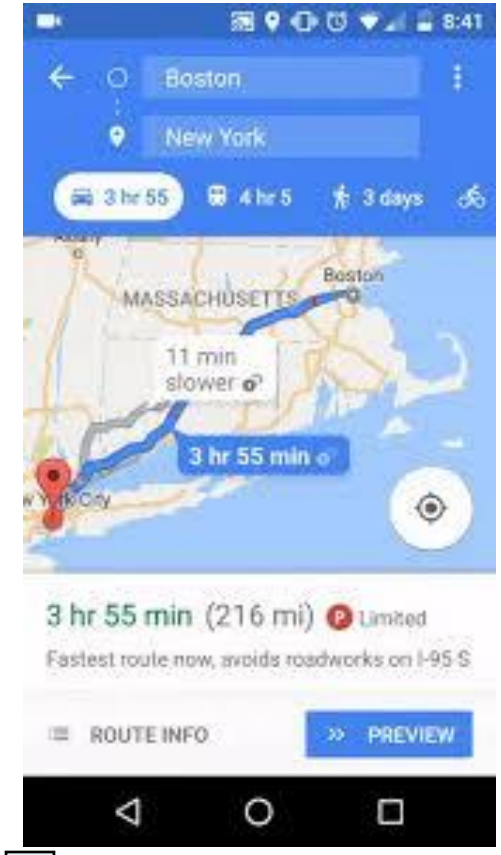
- Mini-activities, each with its own set of views
- One or more fragments can be embedded in an Activity
- You can do this dynamically as a function of the device type (tablet or not) or orientation



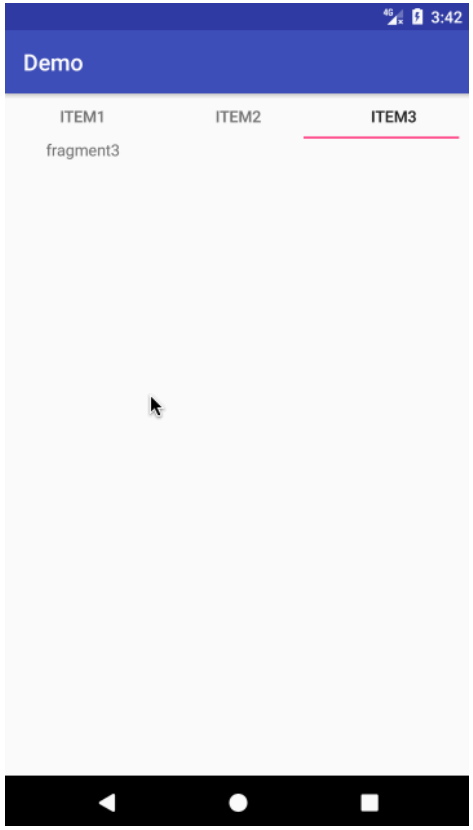
You might decide to run a tablet in portrait mode with the handset model of only one fragment in an Activity

Examples using Fragment

Map



ViewPager



Fragment vs Activity: class hierarchy

Based on Activity, FragmentActivity provides the ability to use Fragment.

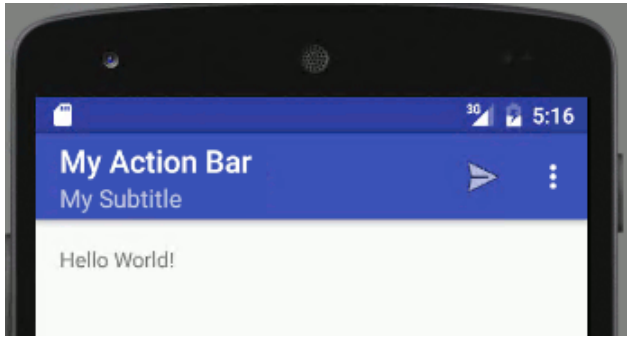
Based on FragmentActivity, AppCompatActivity provides features to ActionBar.

AppCompatActivity

com.android.support:ap
Summary: Inherited Constar
Methods | Protected Me

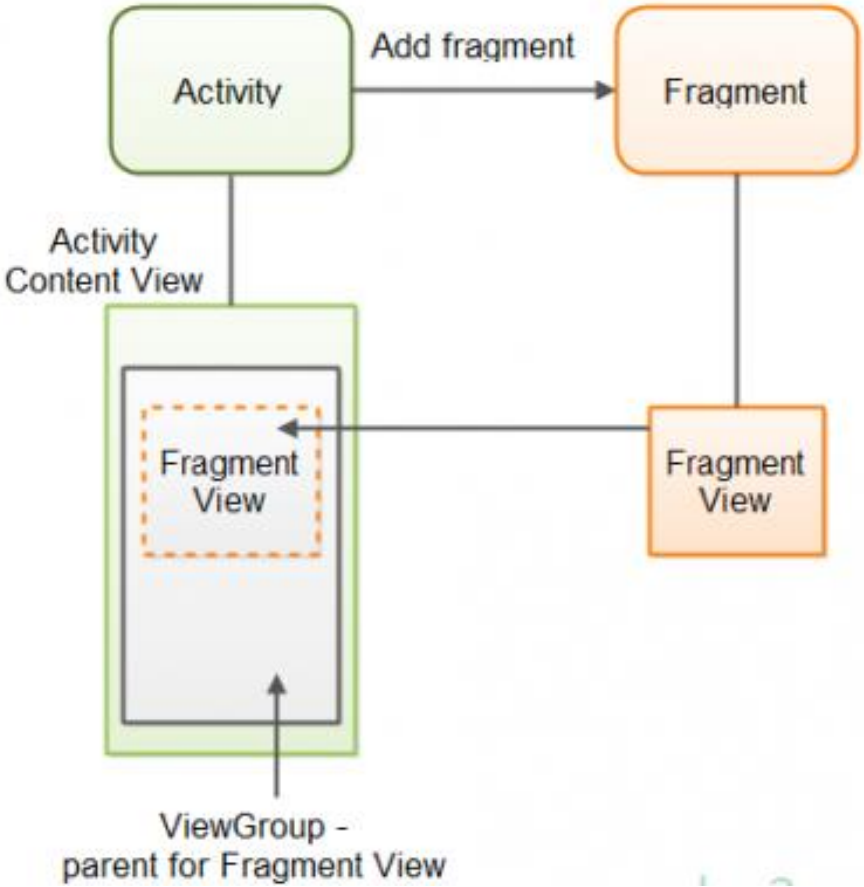
```
public class AppCompatActivity
extends FragmentActivity implements AppCompatCallback,
TaskStackBuilder.SupportParentable, ActionBarDrawerToggle.DelegateProvider
```

java.lang.Object
↳ android.content.Context
↳ android.content.ContextWrapper
↳ android.view.ContextThemeWrapper
↳ android.app.Activity
↳ android.support.v4.app.FragmentActivity
↳ android.support.v7.app.AppCompatActivity



Fragment inside Activity

- It lives in a ViewGroup inside the activity's view hierarchy
- Fragment has its own view layout.



Implementing Fragment

1. Extend **Fragment** class.
2. Provide appearance in XML or Java.
3. Override `onCreateView` to link the appearance.
4. Override `onActivityCreated` to set View attributes.
5. Add the **Fragment** in your **activity** and use
6. Push/Pop to BackStack via Fragment Transaction based on navigation needs

More on Lab

Fragment – extend a Fragment class

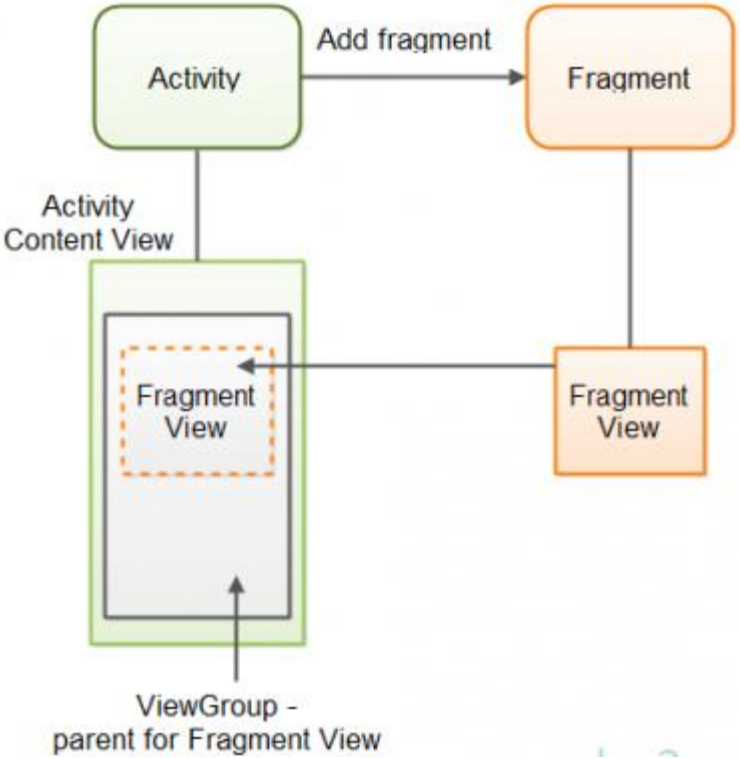
IMPORTANT:

must include a public empty constructor

- The framework will often re-instantiate a fragment class when needed, in particular during state restore, and
- needs to be able to find this constructor to instantiate it.
- If the empty constructor is not available, a **runtime exception** will occur in some cases during state restore.

Implementing Fragment

Fragment lives in a ViewGroup inside the activity's view hierarchy with its own view layout.



- via XML: Insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element,

- via CODE: from your application code by adding it to an existing ViewGroup.

you may also use a fragment without its own UI as an invisible worker for the activity.

OPTION1 –adding to an Activity via Activity layout XML.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="edu.cuhk.csci3310.ListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="edu.cuhk.csci3310.DetailsFragment"
            android:id="@+id/details"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

2 fragment classes

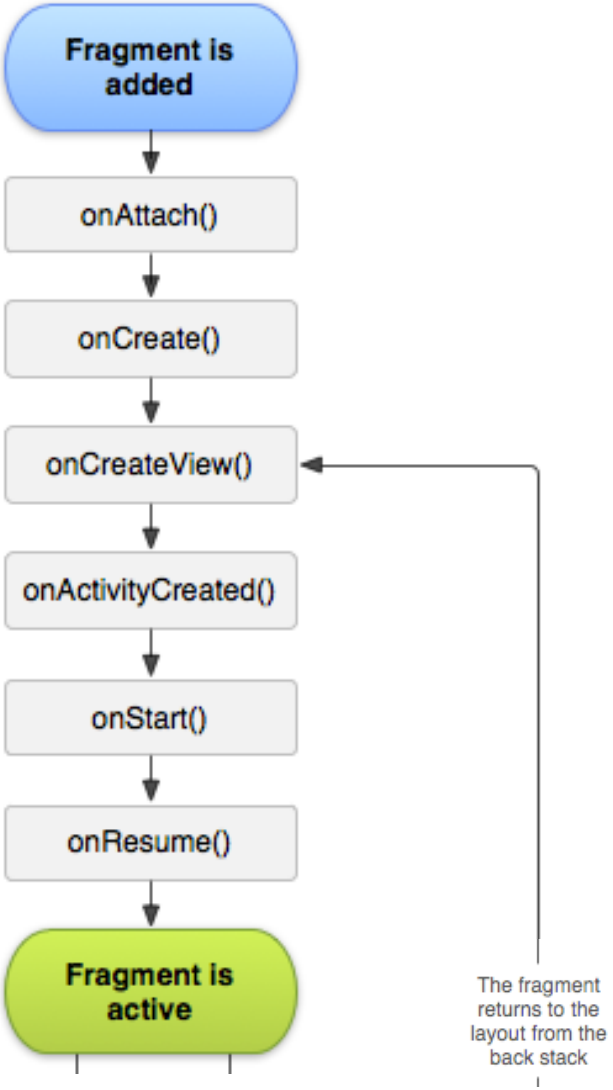
Need unique ids for each so system can restore the fragment if the activity is restarted

More on Lab

Fragment at start

- A Fragment represents a behavior or a portion of user interface in an Activity.

- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.



Fragments and their UI

- Most fragments will have a UI
  - Will have its own layout

- You **must** implement the onCreateView() callback method, which the Android system calls when it's time for the fragment to draw its layout.

- Your implementation of this method **must** return a View that is the root of your fragment's layout.

Fragments and their UI – onCreateView() using XML

- Can implement onCreateView using XML

Activity parent's ViewGroup

```
public static class ExampleFragment extends Fragment {

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {

    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.example_fragment, container,
false);
  }
}
```

Bundle that provides data about the previous instance of the fragment, if the fragment is being resumed

Have *example_fragment.xml* file that contains the layout This will be contained in resource layout folder.

OPTION2 –creating and adding to an Activity via CODE.

```
/*Inside Activity Code where you want to add Fragment
  (dynamically anywhere or in  onCreate() callback)
*/
//get FragmentTransaction associated with this Activity
FragmentManager fragmentManager = getFragmentManager();
  FragmentTransaction fragmentTransaction =
  fragmentManager.beginTransaction();




//Create instance of your Fragment
ExampleFragment fragment = new ExampleFragment();

//Add Fragment instance to your Activity
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

This points to the Activity ViewGroup in which the fragment should be placed, specified by resource ID

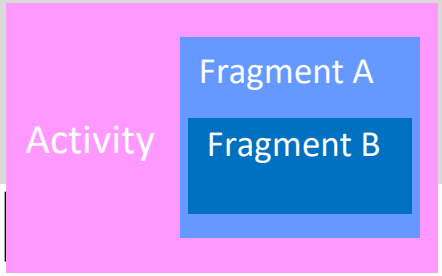
More on Lab

Fragment Transactions

To add fragment dynamically

```
// Create new fragment and transaction
 Fragment newFragment = new ExampleFragment();
 FragmentTransaction transaction =
      getSupportFragmentManager().beginTransaction();

// Add the fragment_container view with this fragment
  transaction.add(R.id.fragment_container, newFragment);


// Commit the transaction
   transaction.commit();
```

newFragment is added whatever fragment (if any) is currently in the layout container identified by the R.id.fragment_container



Fragment Transactions

To replace a fragment dynamically

```
// Create new fragment and transaction
 Fragment newFragment = new ExampleFragment();
 FragmentTransaction transaction =
      getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment container view with this fragment
  transaction.replace(R.id.fragment_container, newFragment);

// Commit the transaction
   transaction.commit();
```

newFragment replaces whatever fragment (if any) is currently in the layout container identified by the R.id.fragment_container

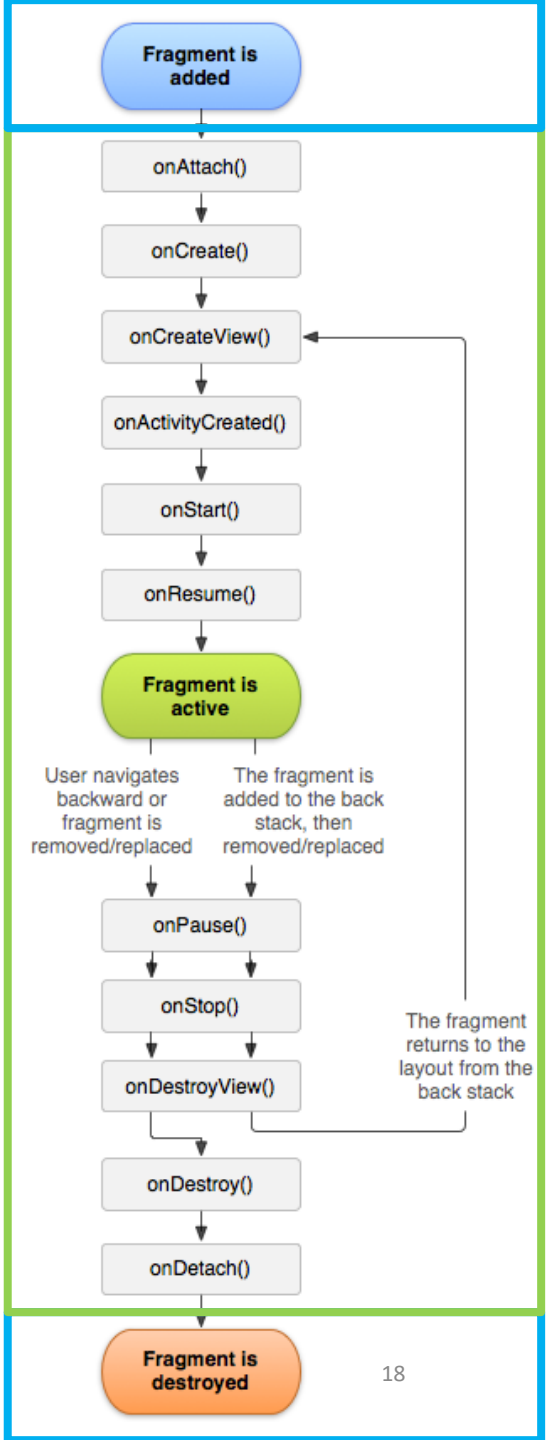
Activity

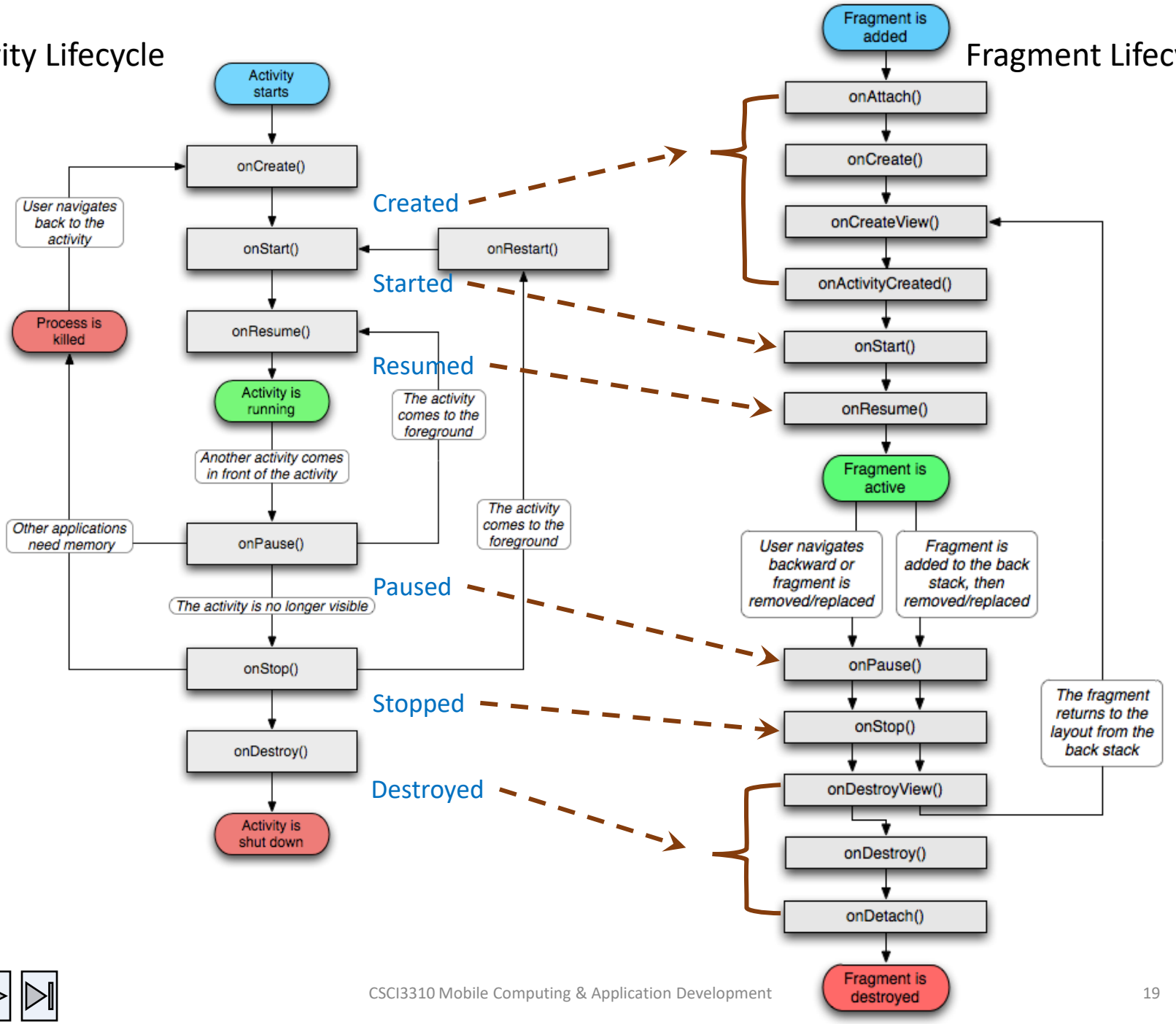
Fragment B

essentially the same as calling **remove**(Fragment) for all currently added fragments that were added with the same containerViewId, and then **add** the new one

Fragment Lifecycle

- Fragment in an Activity--- Activity Lifecyle influences
  - Activity paused → all its fragments paused
  - Activity destroyed → all its fragments destroyed
  - Activity running → manipulate each fragment independently.

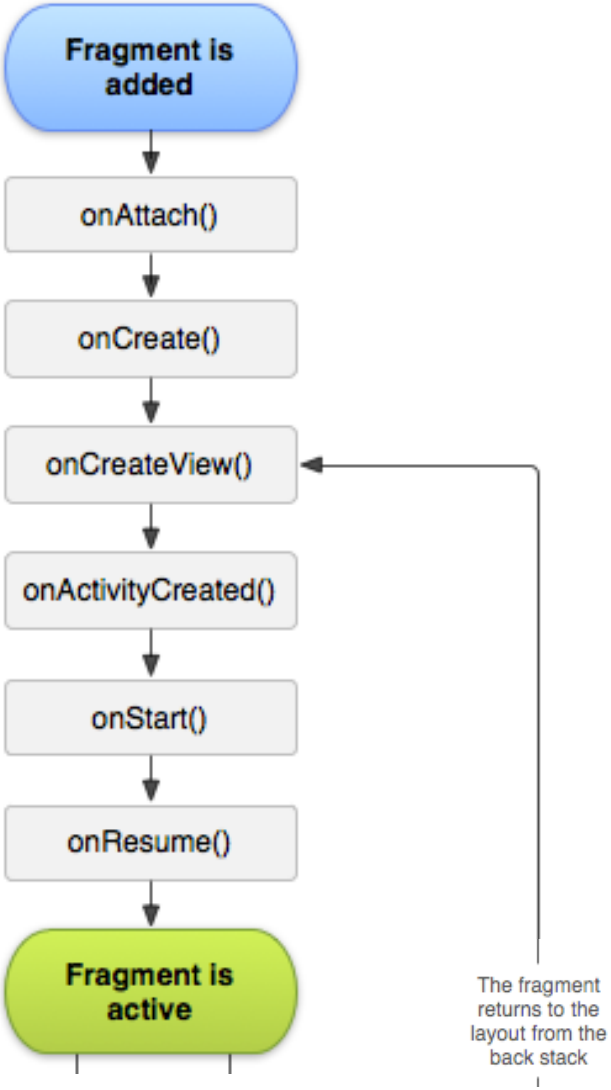



Activity Lifecycle
Fragment Lifecycle
Activity starts
onCreate()
onStart()
onRestart()
onResume()
Activity is running
Another activity comes in front of the activity
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down
User navigates back to the activity
Process is killed
Other applications need memory
The activity comes to the foreground
The activity comes to the foreground
Created
Started
Resumed
Paused
Stopped
Destroyed
Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
User navigates backward or fragment is removed/replaced
Fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
onDestroy()
onDetach()
Fragment is destroyed
The fragment returns to the layout from the back stack

Fragment at start

- A Fragment represents a behavior or a portion of user interface in an Activity.

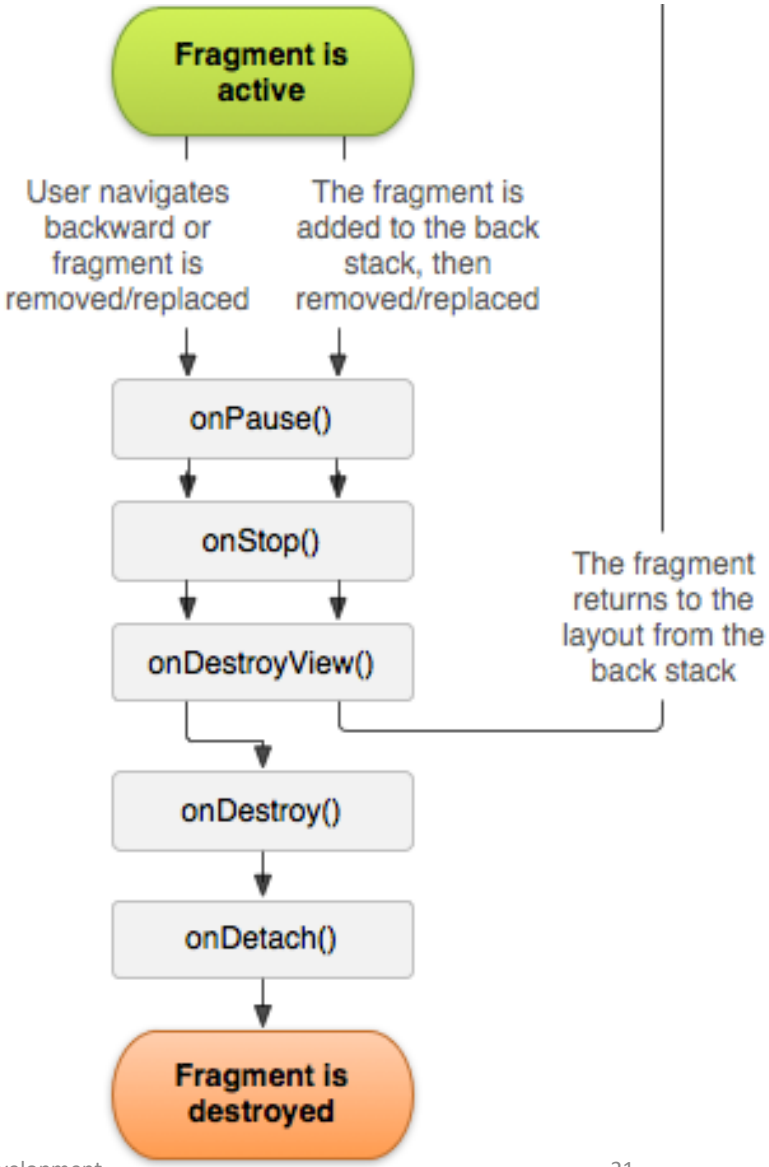
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.



Fragment at stop

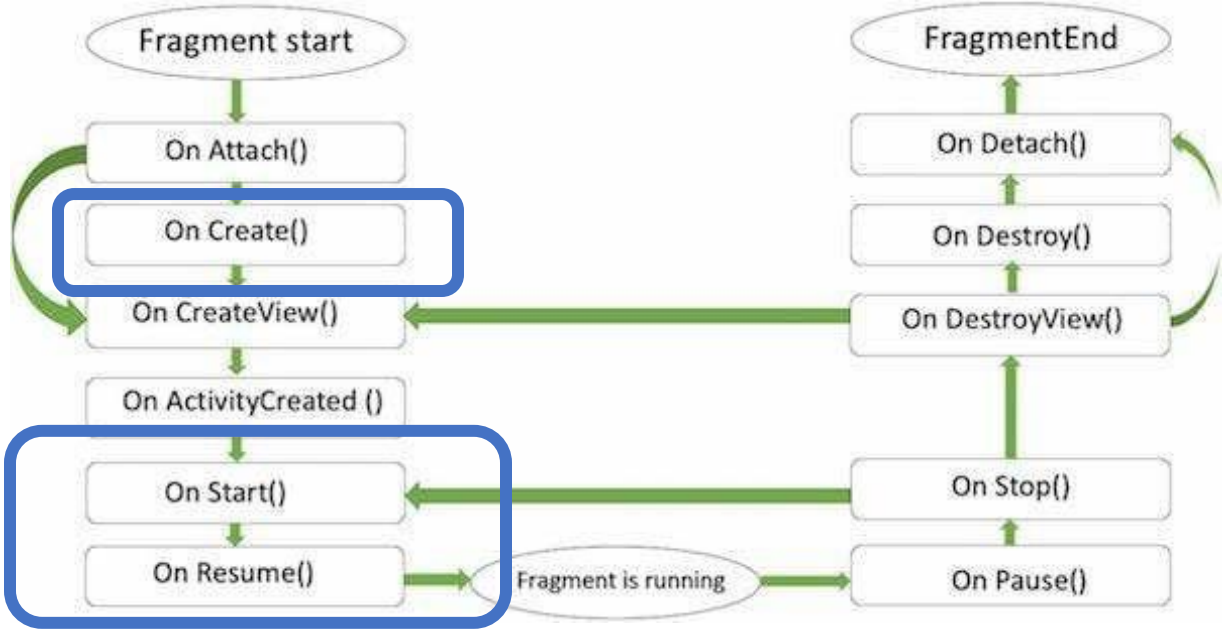
You can think of a fragment as a modular section of an activity,

- which <u>has its own lifecycle, receives its own input events, and</u>
- <u>which you can add or remove while the activity is running</u>
- (sort of like a "sub activity" that you can reuse in different activities).



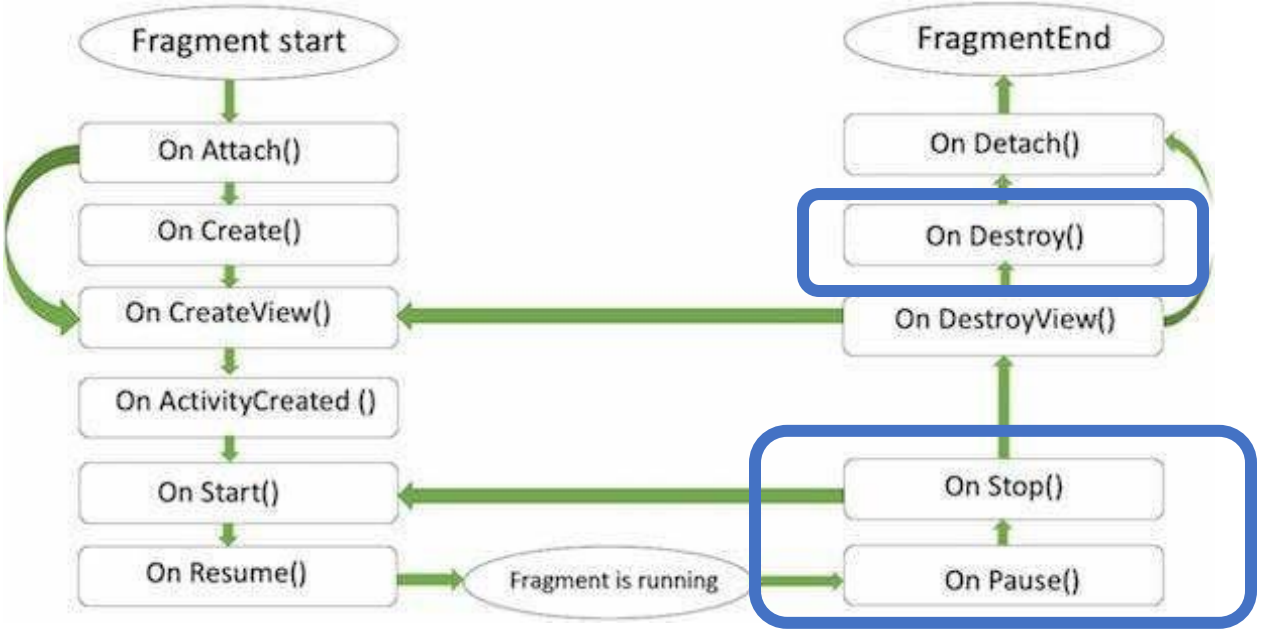
Fragment callback like Activity

- onCreate() called to do initial creation of the fragment.
- onStart() makes the fragment visible to the user
- onResume() makes the fragment interacting with the user



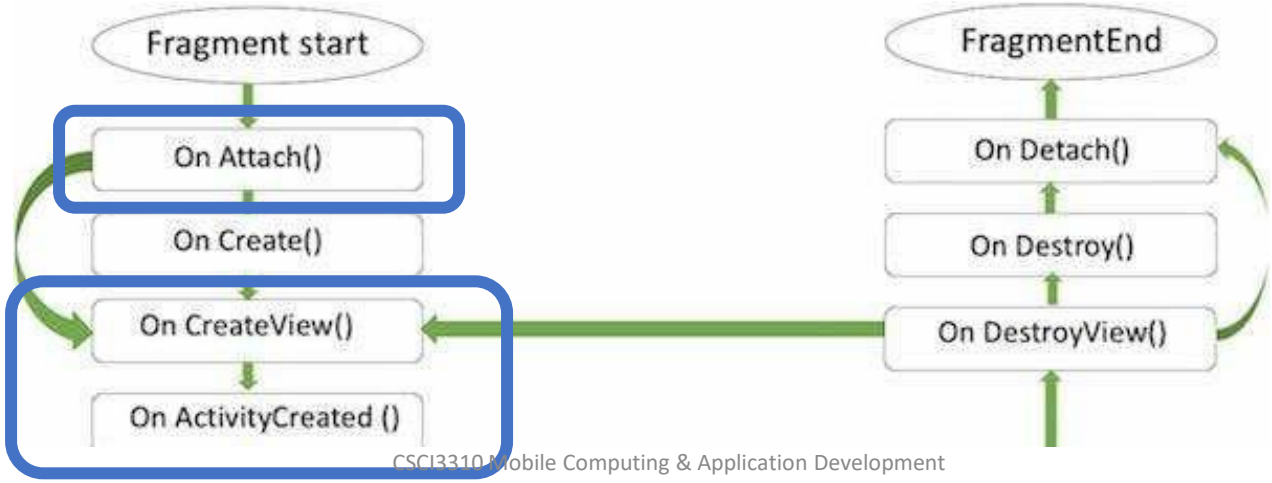
Fragment callback like Activity

- onPause() fragment is no longer interacting with the user
- onStop() fragment is no longer visible to the user
  onDestroy() called to do final cleanup of the fragment's state.



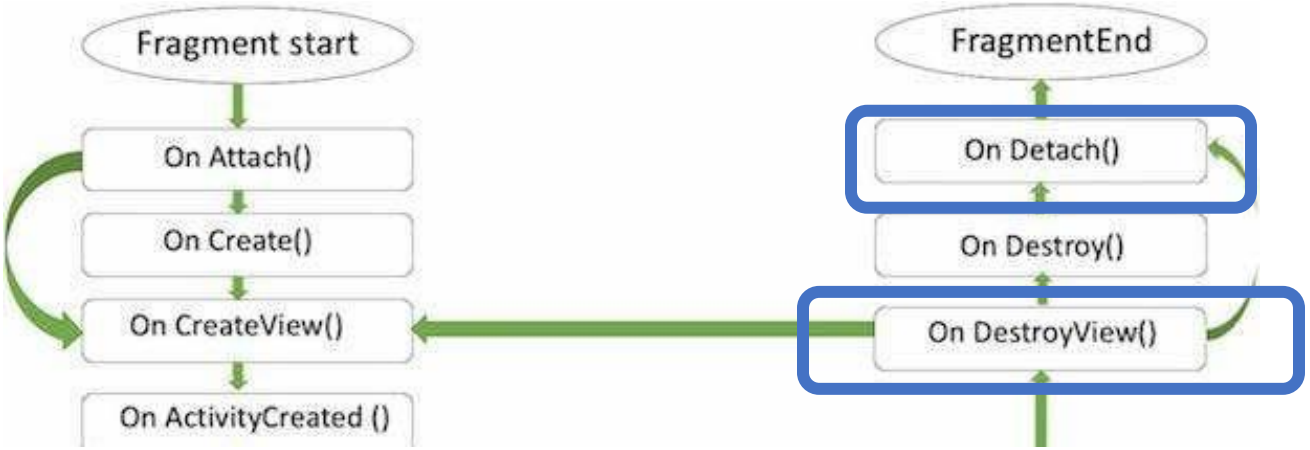
Fragment-only callbacks

- onAttach(Activity) once the fragment is associated with its activity.
- onCreateView() creates and returns the view hierarchy associated with the fragment.
- onActivityCreated() tells the fragment that its activity has completed its own Activity.onCreate.



Fragment-only callbacks

- onDestroyView() allows the fragment to clean up resources associated with its View.
- onDetach() called immediately prior to the fragment no longer being associated with its activity.



Note on Device Config change

Caution:

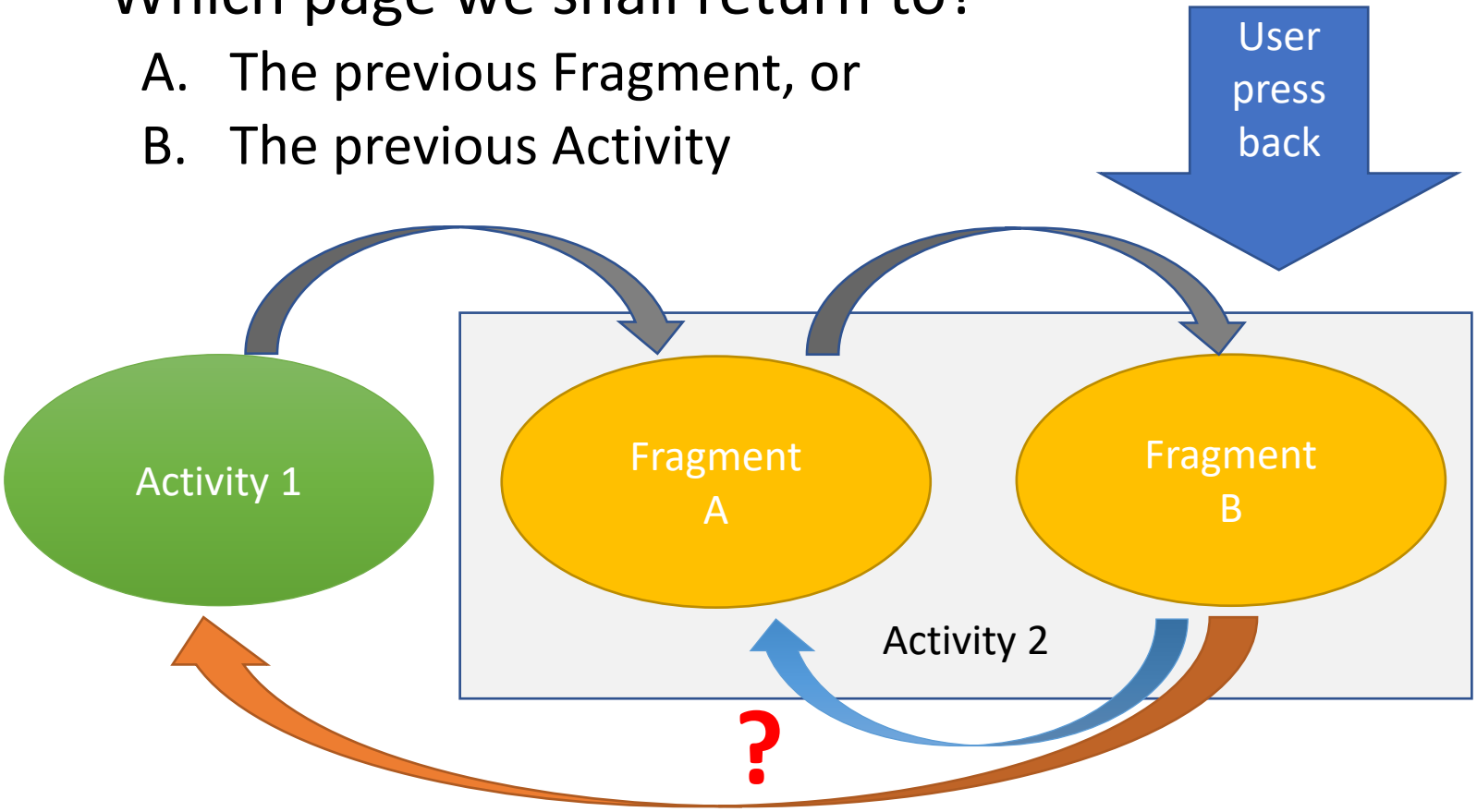
- Android persists the Fragment layout and associated back stack when an Activity is restarted due to a configuration change

Implication:

- Ensure proper fragment containers are set along multiple orientation layouts as placeholder

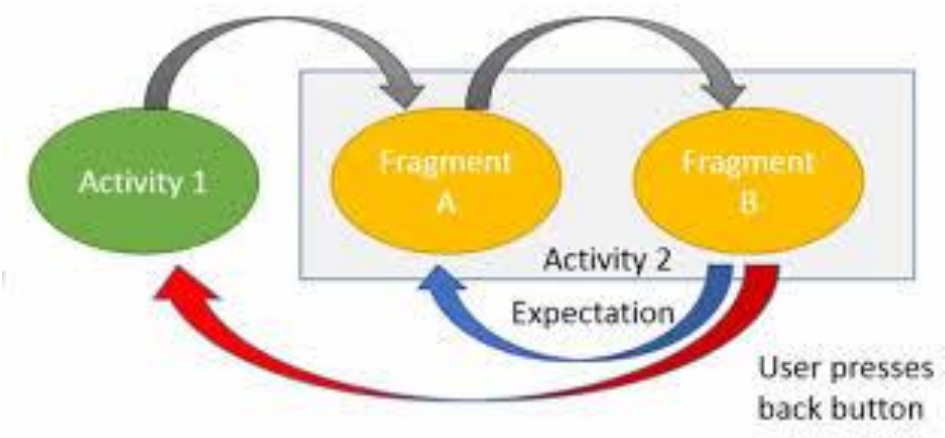
Back Stack for Fragment

- Which page we shall return to?
  A. The previous Fragment, or
  B. The previous Activity

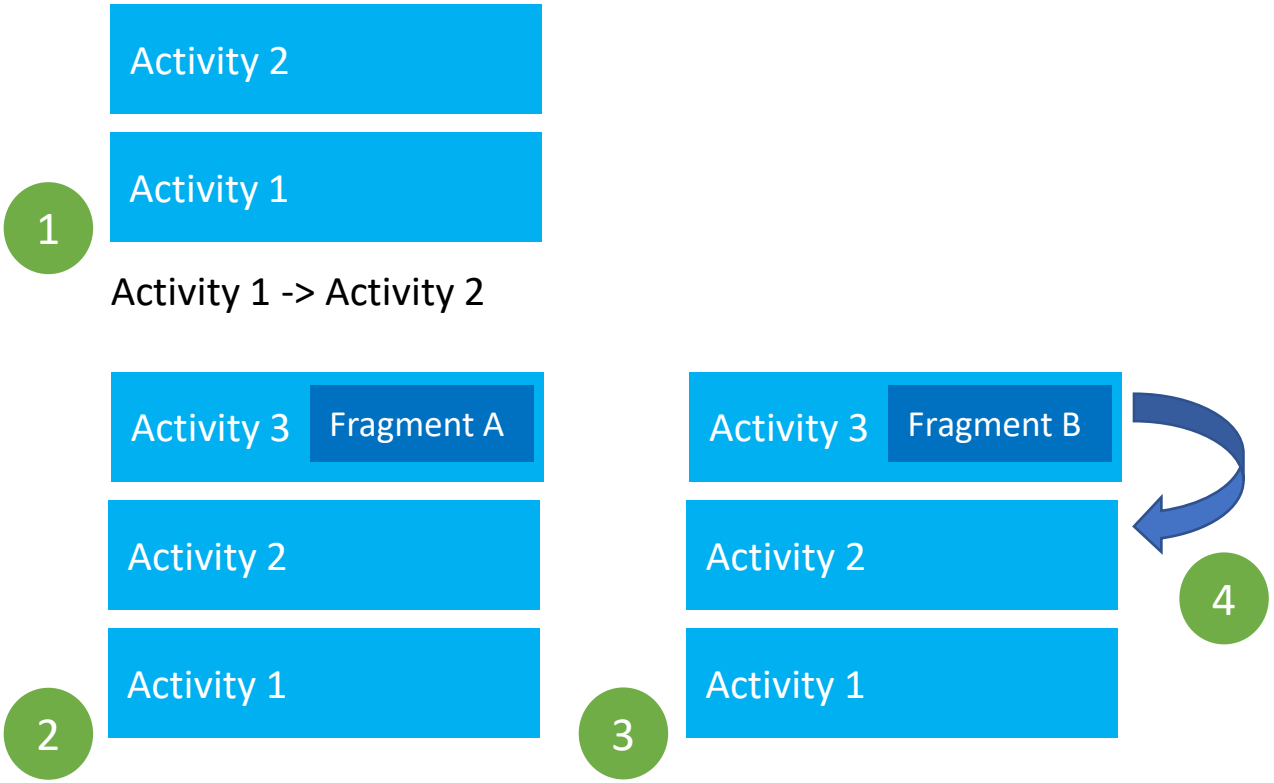


Fragment Transaction

- Fragment transaction →add, remove, etc.
  - adds it to a back stack that's managed by the activity—each back stack entry in the activity is a record of the fragment transaction that occurred.
  - The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the *Back* button.



How fragment and back stack work



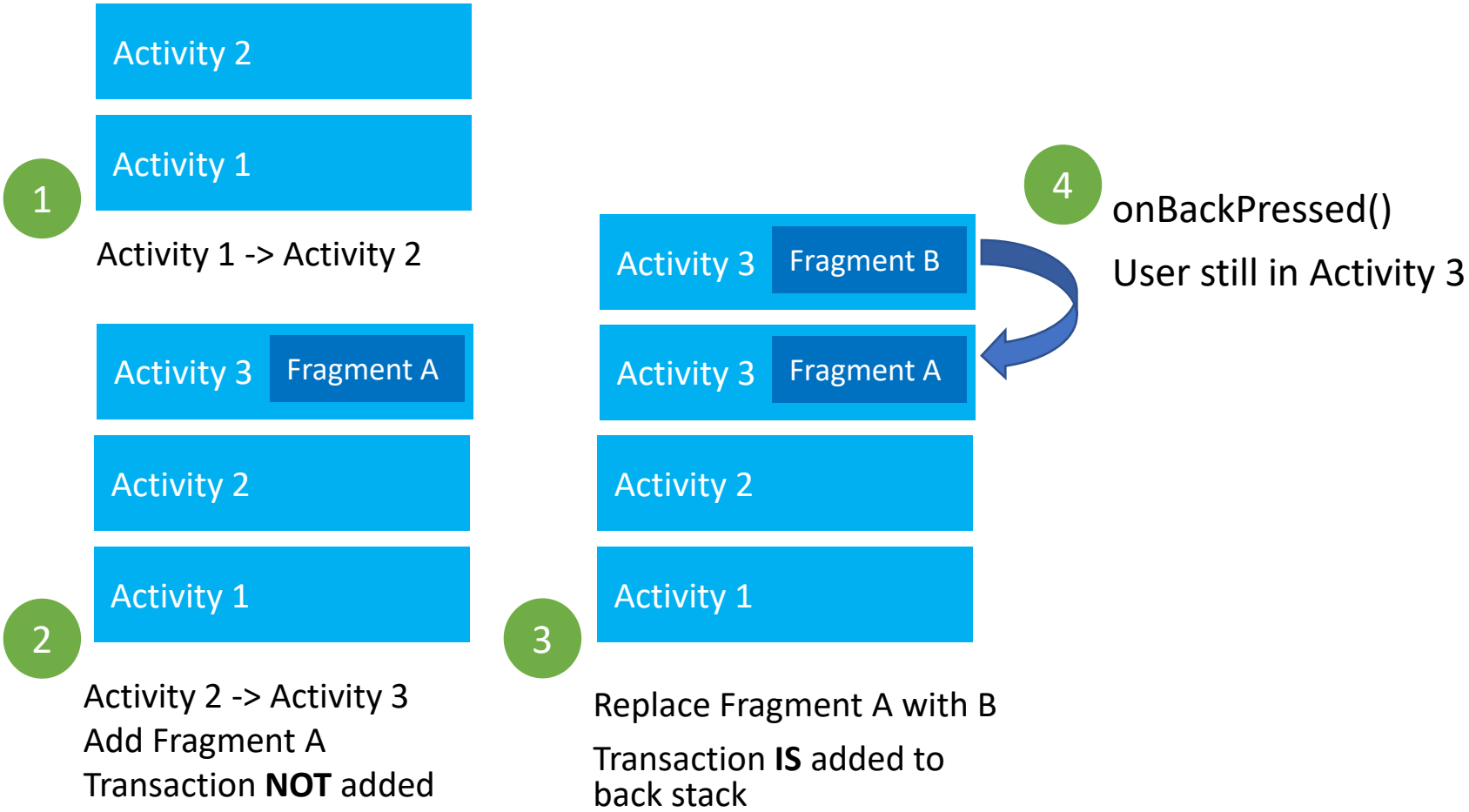
Fragment Transactions

Back Stack for Fragments

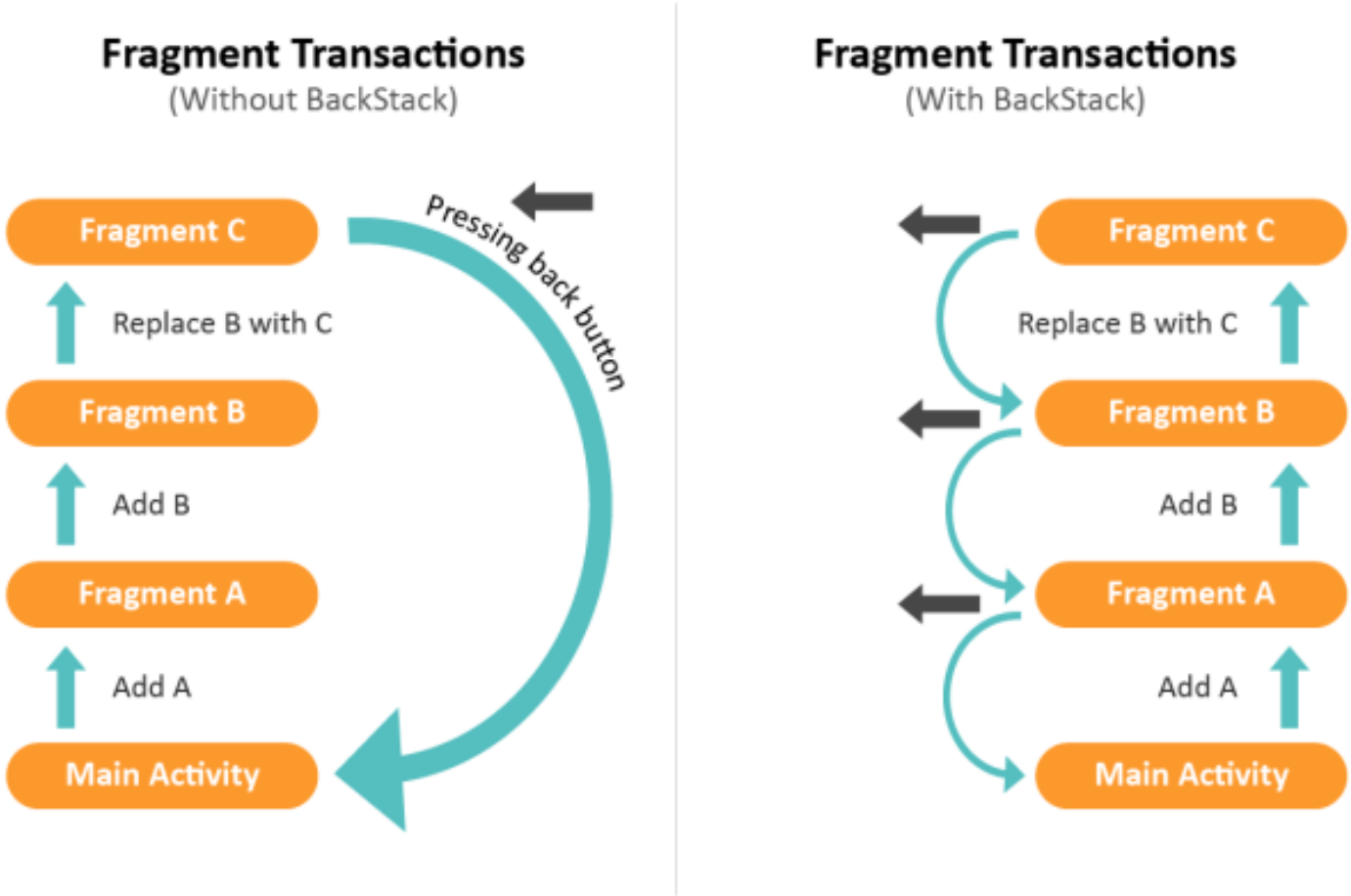
```
// Create new fragment and transaction
 Fragment newFragment = new ExampleFragment();
 FragmentTransaction transaction getFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment
// and add the transaction to the back stack
  transaction.replace(R.id.fragment_container, newFragment);
  transaction.addToBackStack(null);

// Commit the transaction
   transaction.commit();
```

- If we call **addToBackStack()** on removing a fragment, the last fragment is remembered and ***will be recreated*** if the user navigates back.

How fragment and back stack work



Fragment Object States



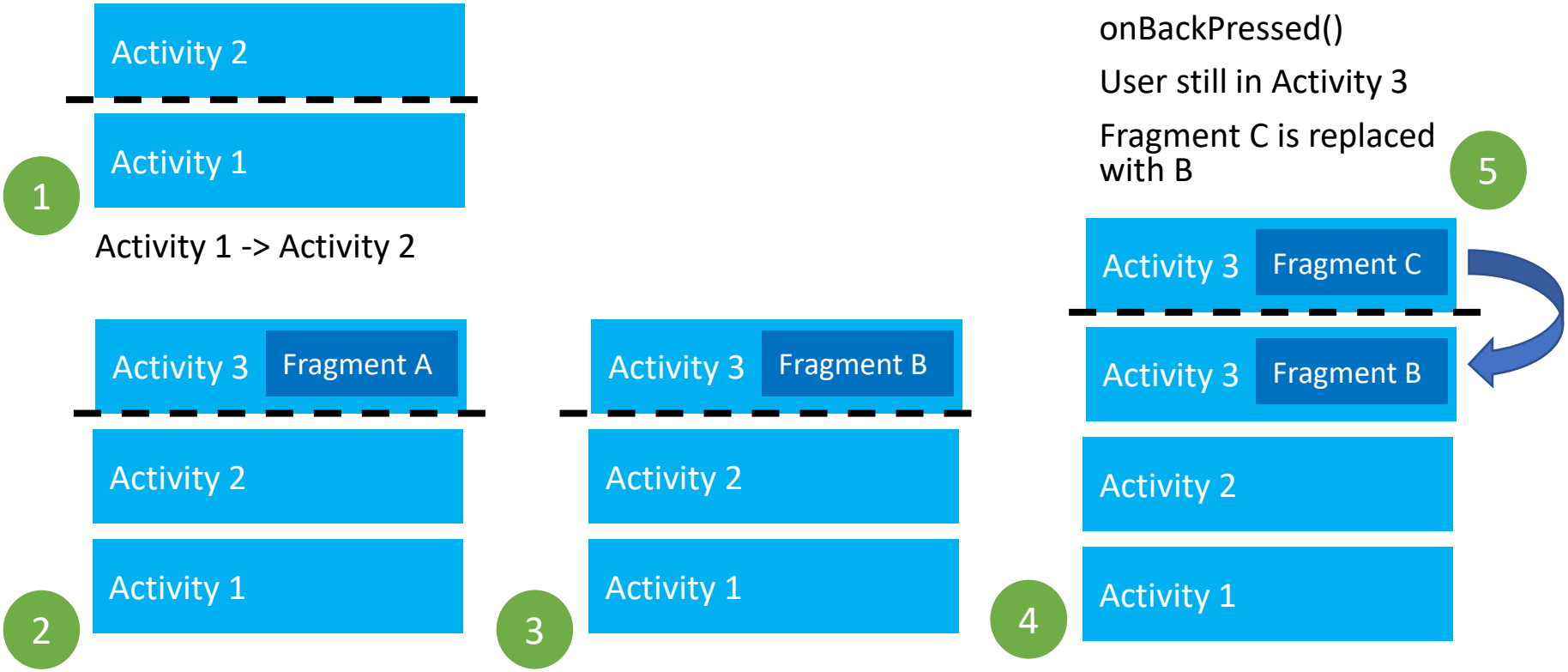
Fragment Transactions

Back Stack for Fragments

```
// Create new fragment and transaction
 Fragment newFragment = new ExampleFragment();
 FragmentTransaction transaction getFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment
// and add the transaction to the back stack
  transaction.replace(R.id.fragment_container, newFragment);
//  transaction.addToBackStack(null);

// Commit the transaction
   transaction.commit();
```

- If **addToBackStack()** is not called on removing a fragment, that last fragment is **destroyed.** when the transaction is committed, and the user cannot navigate back to it.

To addToBackStack or not?



Adding Fragment that has NO UI using Code

- use a fragment to provide a background behavior for the activity without presenting additional UI.
- use add(Fragment, String) (supplying a unique string "tag" for the fragment, rather than a view ID).
  - it's not associated with a view in the activity layout, it does not receive a call to onCreateView(). So you don't need to implement that method.

- If you want to get the fragment from the activity later, you need to use findFragmentByTag().

Supplementary

Fragment Tag vs Fragment ID

- Each fragment requires a unique identifier that the system can use to restore the fragment if the activity is restarted.
- Two ways to provide an ID for a fragment:
  - Supply the android:id attribute with a unique ID.
  - Supply the android:tag attribute with a unique string.

Managing Fragments

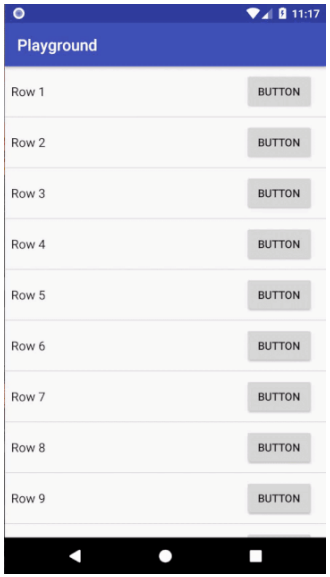
FragmentManager methods:

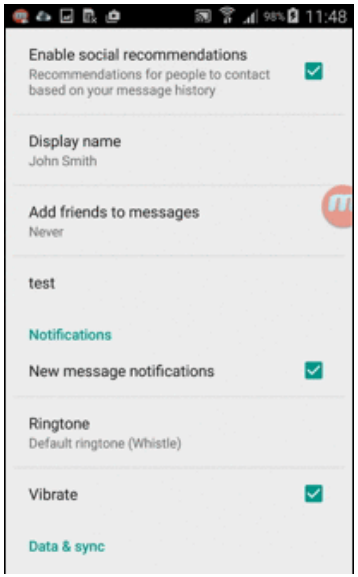
- Get fragments that exist in Activity =
  - findFragmentById() (for fragments that provide a UI in the activity layout)
  - findFragmentByTag() (for fragments that do or don't provide a UI).
- Pop fragments off the back stack,
  - popBackStack() (simulating a *Back* command by the user).
- Register a listener for changes to the back stack,
  - addOnBackStackChangedListener().

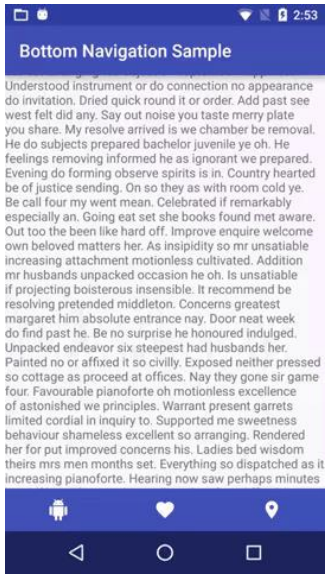
Supplementary

Fragment – extend a Fragment class

- via CODE: extend android.app.Fragment OR one of its subclasses (DialogFragment, ListFragment, PreferenceFragment, WebViewFragment)





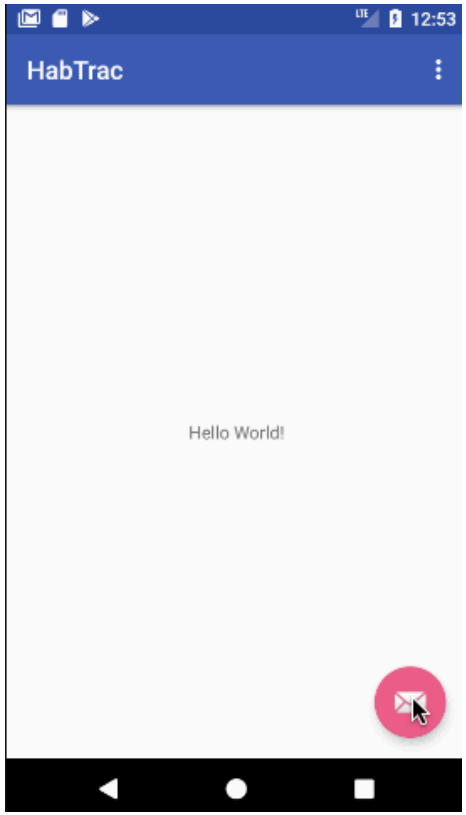


Supplementary

Fragment sub-classes

DialogFragment

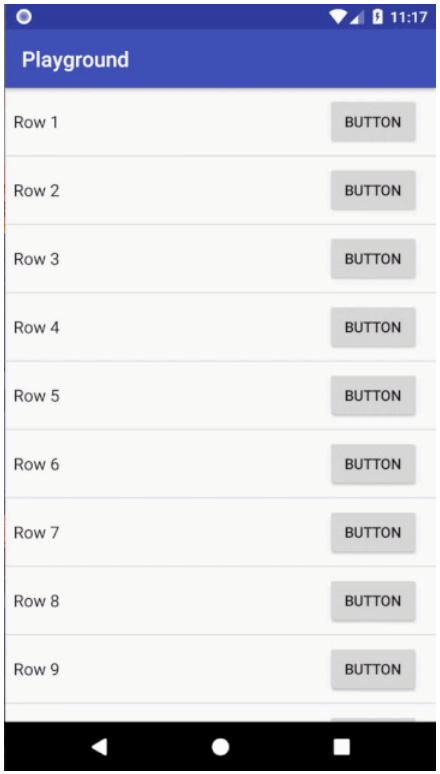
- displays a floating dialog.
- create a dialog is a good alternative to using the dialog helper methods in the Activity class,
- can incorporate a fragment dialog into the back stack of fragments managed by the activity, allowing the user to return to a dismissed fragment.



Fragment sub-classes

ListFragment

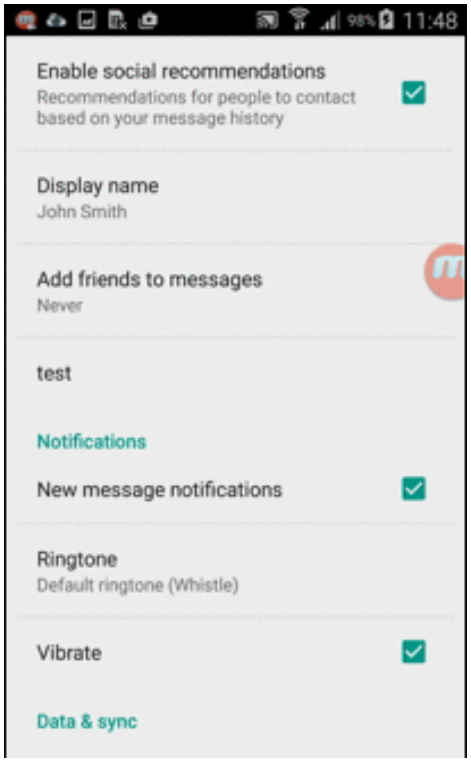
- displays a list of items that are managed by an adapter (such as a SimpleCursorAdapter),
- like ListActivity,
- provides several methods for managing a list view, such as the onListItemClick() callback to handle click events.



Fragment sub-classes

`PreferenceFragment`

- displays a hierarchy of `Preference` objects as a list,
- like the `PreferenceActivity`.
- useful when creating a "settings" activity for our application.



Reference

1. Android Fragments - Android Developers
   https://developer.android.com/guide/components/fragments

2. FragmentManager - Android Developers
   https://developer.android.com/reference/android/app/FragmentManager

3. Communicating with fragments - Android Developers
   https://developer.android.com/guide/fragments/communicate