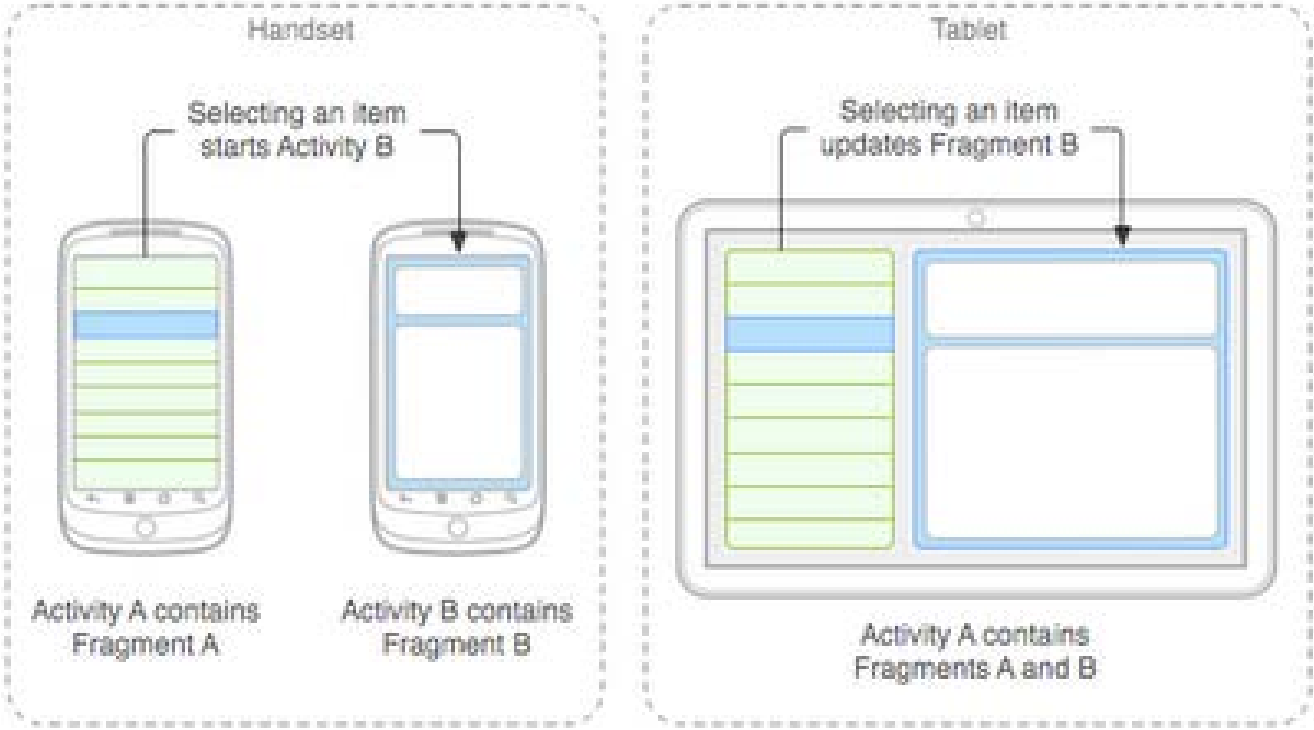
Fragments

Programming the Android Platform

CS 282
Principles of Operating Systems II
Systems Programming for Android

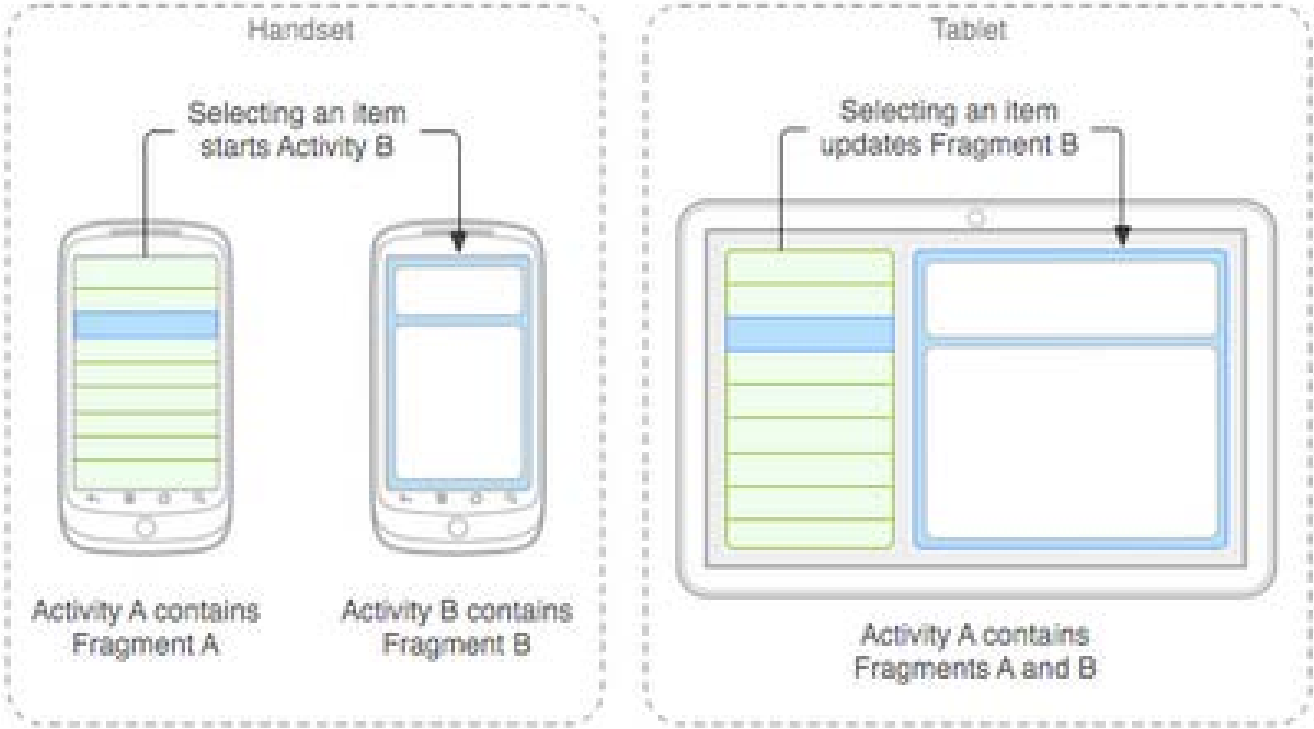
Fragment Overview

- Tablets have larger displays than phones do, so they can support multiple UI panes / user behaviors at same time
  - May not need "one activity per screenful of data" rule of thumb
- Android Fragments are an optional layer you can put between your activities & your widgets to help you reconfigure your activities to support screens both large & small



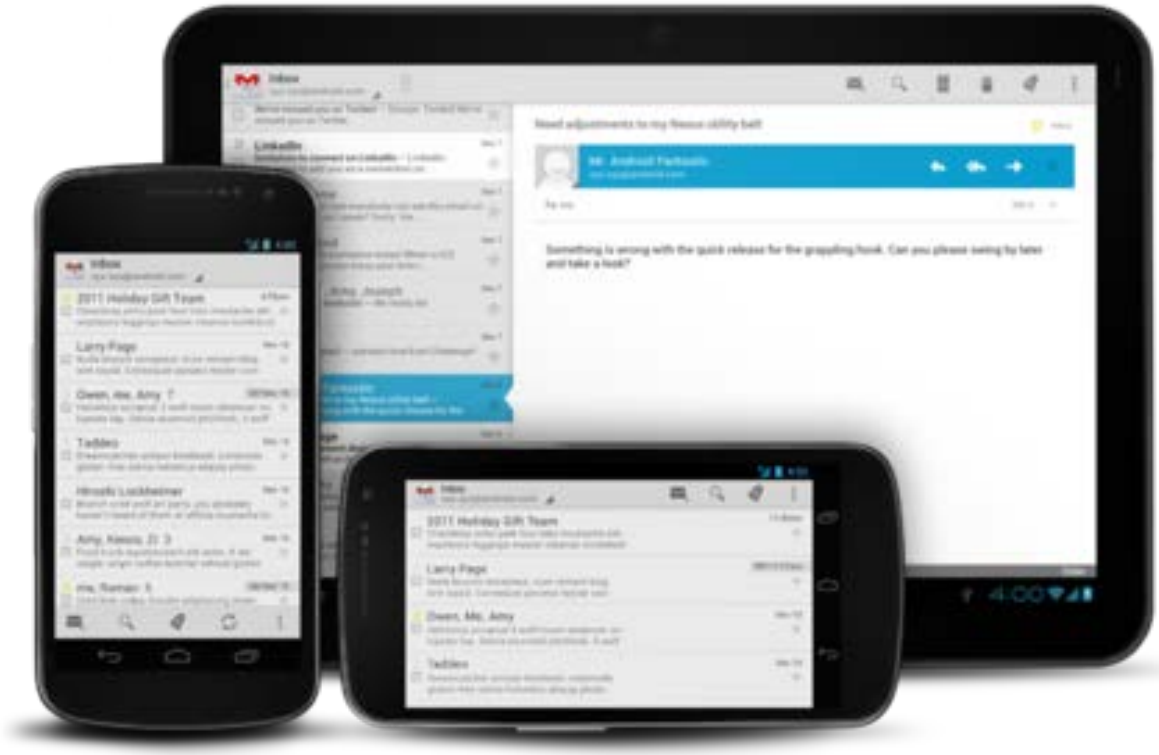
Fragment Overview (cont'd)

- An Android Fragment represents a behavior / portion of UI within an Activity
- Multiple Fragments can be embedded in an Activity to create a multi-pane UI
- A single Fragment can be reused across multiple Activities



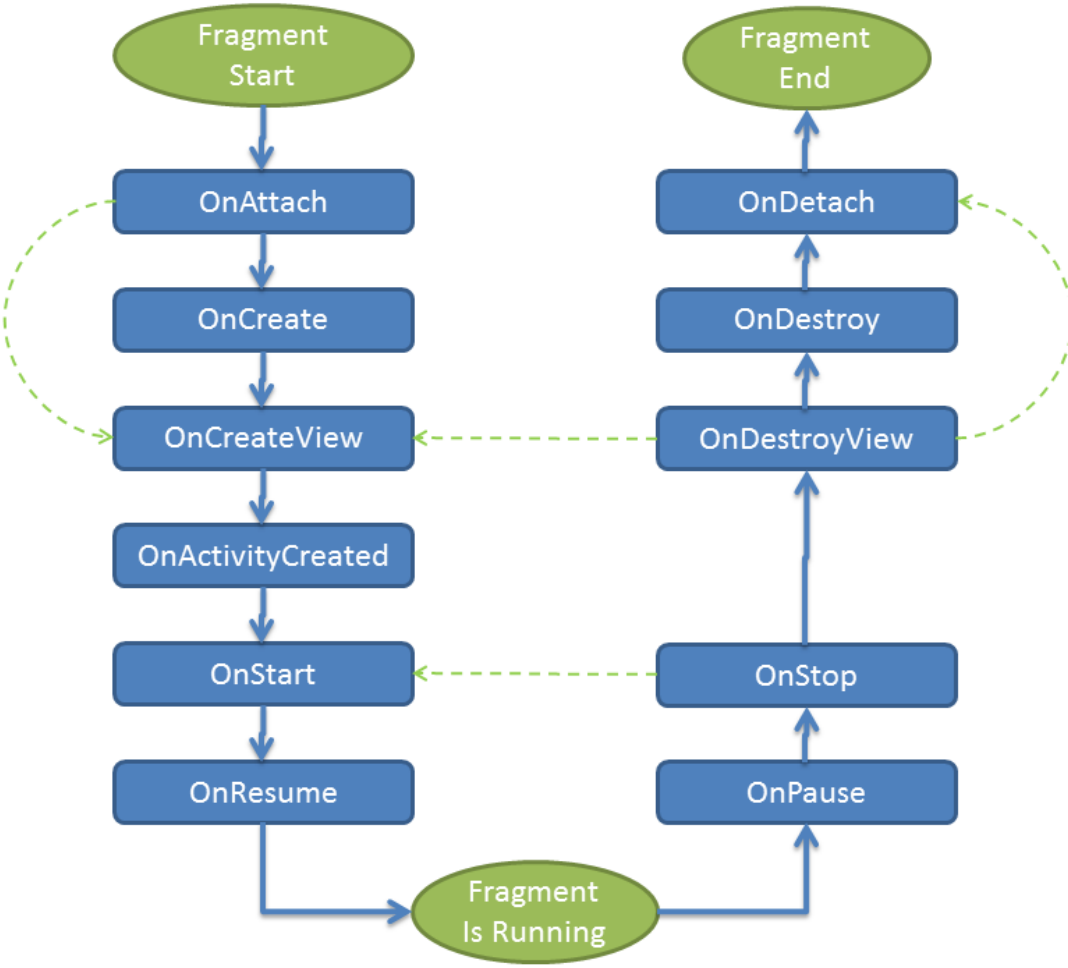
Email Viewer Example

- On phone – 2 Activities
  - List email messages
  - View selected email message
- On tablet – 2 Fragments embedded in 1 Activity



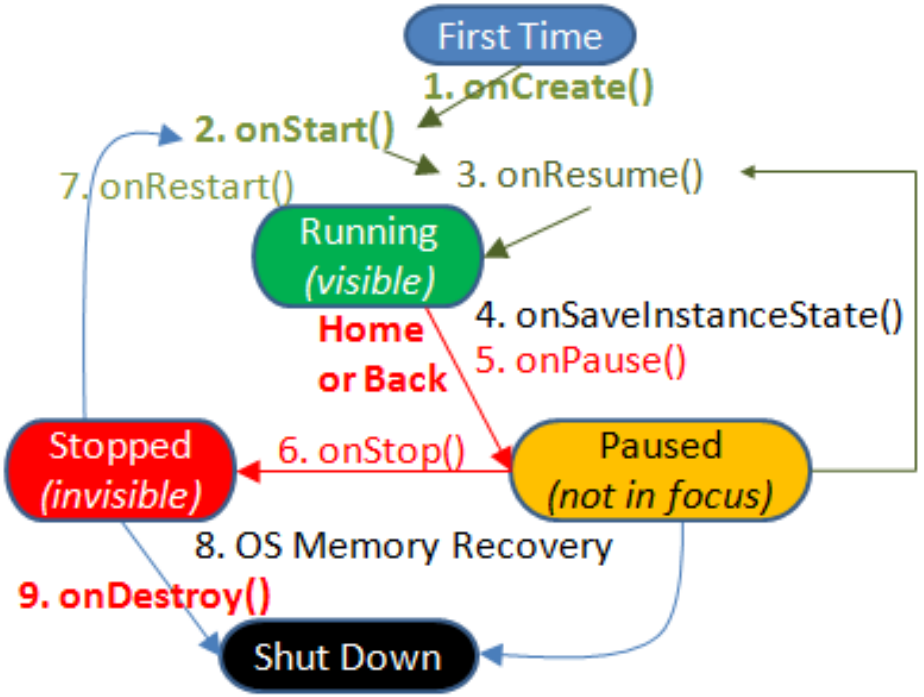
Fragment Lifecycle

- Fragments have their own lifecycles & receive their own events
- But Fragment lifecycle interacts with containing Activity's lifecycle, e.g.,
  - When Activity pauses, its Fragments are paused
  - When Activity is destroyed, its Fragments are destroyed



Fragment Lifecycle States

- Similar to Activity Lifecycle States
- Resumed
  - Fragment is visible in the running activity
- Paused
  - Another activity is in the foreground & has focus
  - The containing activity is still visible
- Stopped
  - The fragment is not visible



Fragment Lifecycle Callbacks

- **onCreate()**
  - Initial creation of the fragment
- onStart()
  - Fragment is visible to the user
- onResume()
  - Fragment is visible to the user & actively running
- onPause()
  - Fragment is visible, but does not have focus
- onStop()
  - Fragment is no longer visible
- **onDestroy()**
  - Fragment is no longer in use
- onAttach()
  - Fragment is first attached to its activity
- **onCreateView()**
  - Fragment instantiates its user interface view
- onActivityCreated()
  - Fragment's activity created & Fragment's view hierarchy instantiated
- onDestroyView()
  - View previously created by onCreateView() detached from the Fragment
- onDetach()
  - Fragment no longer attached to its activity

Fragment & Activity Lifecycles

Activity Callbacks:

onCreate() onStart() onResume() onPause() onStop() onDestroy()

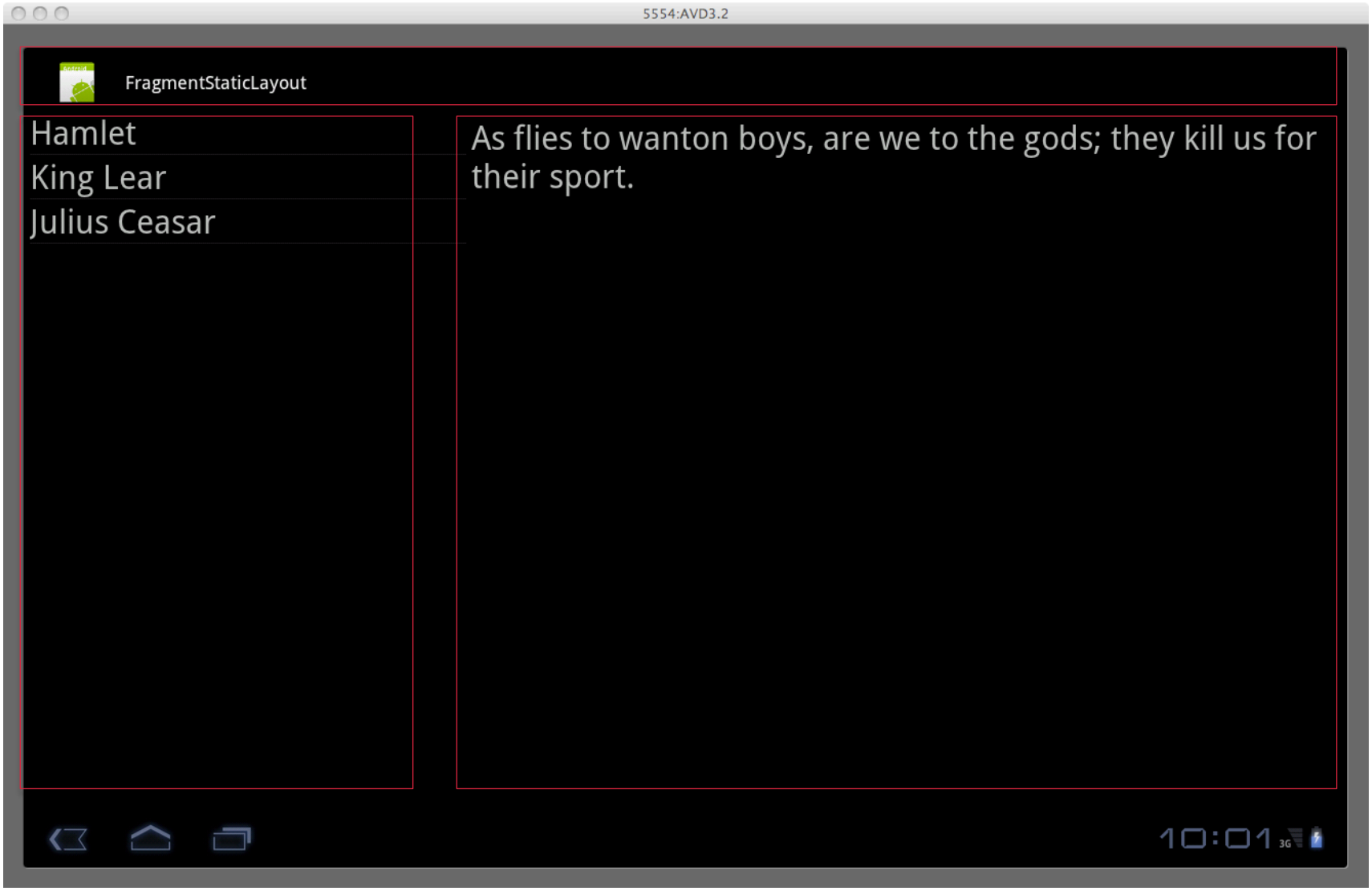
Fragment Callbacks:

onAttach()	onStart()	OnResume()	onPause()	onStop()	onDestroyView()
onCreate()					onDestroy()
onCreateView()					onDetach()
onActivityCreated()					

Fragment Layout

- Fragments usually, but not always, have a UI
- Layout can be inflated/implemented in onCreateView()
  - onCreateView() must return the View at the root of the Fragment's layout
  - The returned View will be added to the containing Activity
    - Container represented as a ViewGroup within the containing Activity's view hierarchy
- Two ways to add Fragments to an Activity's layout
  - Declare it statically in the Activity's layout file
  - Add it programmatically to a ViewGroup in the Activity's layout

Fragment Layout (cont.)



Fragment Layout via XML (cont.)

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  …
}
```

main.xml

```
<LinearLayout …>
   <fragment
       class="Fragments.TitlesFragment"
       android:id="@+id/titles”  android:layout_weight=”1” … />
   <fragment
       class="Fragments.DetailsFragment"
       android:id="@+id/details” android:layout_weight=“2” … />
</LinearLayout>
```

Fragment Constructors

- Static factory methods are the preferred way to instantiate fragments
- Fragment arguments should be set with the setArguments(Bundle args) method
  - Allows Android to preserve the arguments across the Fragment lifecycle
- Static factory methods construct a new fragment, set the arguments, and then return that fragment

Fragment Constructors (cont.)

```
…
public static DetailsFragment newInstance(int index) {
        DetailsFragment f = new DetailsFragment();

        // Supply index as argument
        Bundle args = new Bundle();
        args.putInt(index);
        f.setArguments(args);

        return f;
}
…
```

DetailsFragment onCreateView

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

    return inflater.inflate(R.layout.detail_fragment,
                            container, false);
}
```

detail_fragment.xml

```
<ScrollView ...>
  <TextView android:id="@+id/quoteView” ...>
  </TextView>
</ScrollView>
```

Programmatic Layout

- While Activity's running you can add a Fragment to the Activity's layout
  - Specify a containing ViewGroup
  - Get reference to FragmentManager
  - Execute a FragmentTransaction

main.xml

```
<LinearLayout xmlns:android=android:id="@+id/activityFrame"
    android:orientation="horizontal"  ...>
  <FrameLayout android:id="@+id/titleFrame"  ...
      android:layout_width="0dp" android:layout_weight="1">
  </FrameLayout>
  <FrameLayout android:id="@+id/detailFrame" ...
      android:layout_width="0dp" android:layout_weight="2">
  </FrameLayout>
</LinearLayout>
```

Programmatic Layout (cont.)

```
private final TitlesFragment mTitlesFragment
                                        = new TitlesFragment();
private final DetailsFragment mDetailsFragment
                                        = new DetailsFragment();
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 …
 setContentView(R.layout.main);
 …
```

Programmatic Layout (cont.)

```
  ...
  FragmentManager fragmentManager = getFragmentManager();
  FragmentTransaction fragmentTransaction =
                                        fragmentManager.beginTransaction();
  fragmentTransaction.add(R.id.titleFrame, mTitlesFragment);
  fragmentTransaction.add(R.id.detailFrame, mDetailsFragment);
  fragmentTransaction.commit();
}
```

Dynamic Layout

```
TitlesFragment mTitlesFragment = new TitlesFragment();
DetailsFragment mDetailsFragment = new DetailsFragment();
...
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 ...
 setContentView(R.layout.main);
 ...
```

Dynamic Layout (cont.)

```
  ...
  mFragmentManager = getFragmentManager();
  FragmentTransaction fragmentTransaction =
                                  mFragmentManager.beginTransaction();
  // adding TitlesFragment
  fragmentTransaction.add(R.id.activityFrame, mTitlesFragment);
  fragmentTransaction.commit();
}
```

main.xml

```
<LinearLayout…
    android:id="@+id/activityFrame”
    android:orientation="horizontal"
    …
</LinearLayout>
```

TitleFragment.onCreateView()

```
public View onCreateView(LayoutInflater inflater,
                    ViewGroup container,Bundle savedInstanceState) {
  LinearLayout ll = new LinearLayout(getActivity());
  ll.setLayoutParams(
     new LinearLayout.LayoutParams(
           0,ViewGroup.LayoutParams.MATCH_PARENT, 1.0f));
  ll.addView(
       super.onCreateView(inflater, container, savedInstanceState));
  return ll;
}
```

Dynamic Layout (cont.)

```
...
public void onListSelection(int index) {
   if (!mDetailsFragment.isAdded()) {
     FragmentTransaction fragmentTransaction =
                                        mFragmentManager.beginTransaction();
     // adding DetailsFragment
     fragmentTransaction.add(R.id.activityFrame,mDetailsFragment);
     // reverse this transaction when Back button is hit
     // argument is name of back state
     fragmentTransaction.addToBackStack(null);
     fragmentTransaction.commit();
     mFragmentManager.executePendingTransactions();
   }
   ...
```

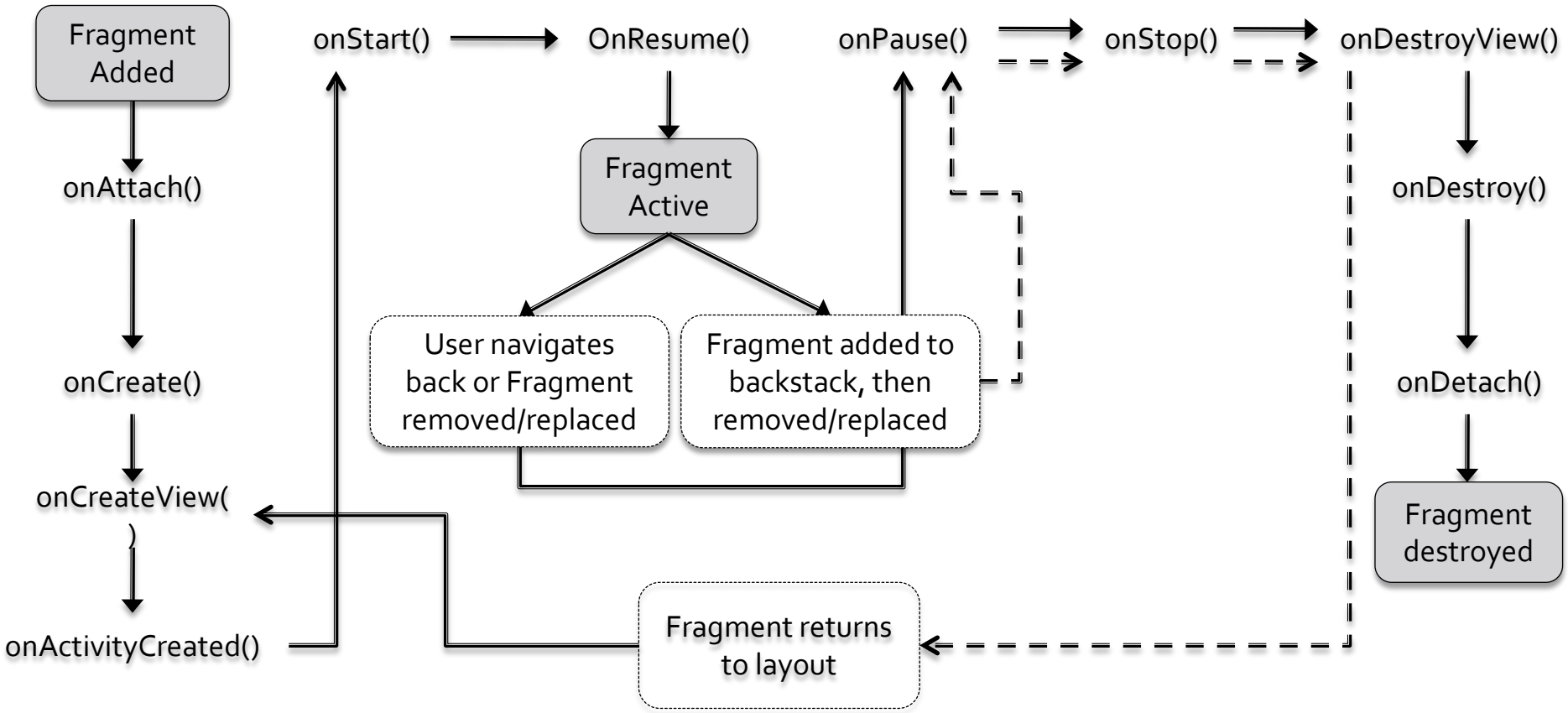
DetailFragment.onCreateView()

```
public View onCreateView( LayoutInflater inflater,
                          ViewGroup container, Bundle savedInstanceState) {
return inflater.inflate(R.layout.detail_fragment, container, false);
}
```

detail_fragment.xml

```
<LinearLayout ...
    android:id="@+id/detail_linear_layout” ...
     android:layout_width="0dp" android:layout_weight="2">
  <ScrollView  android:id="@+id/ScrollView1"
    android:orientation="vertical”  ... >
      <TextView android:id="@+id/quoteView” ...
      </TextView>
  </ScrollView>
 </LinearLayout>
```

Fragment Lifecycle Summary



Screen Rotation

- Changing the screen orientation causes the current Activity to be destroyed and recreated
- Problematic for retaining data
- Can use onSaveInstanceState(Bundle out) to save important fields
  - Bundles can only hold primitives, Parcelable objects, and Serializable objects

Non-UI Fragments

- Fragment instances can be retained across orientation changes
- Can use a Fragment with no UI to retain objects that cannot be bundled (such as Threads)
- Call setRetainInstance(true) from within a Fragment to cause it to be retained
- See FragmentRetainInstance.java in the Android API Demos for an example

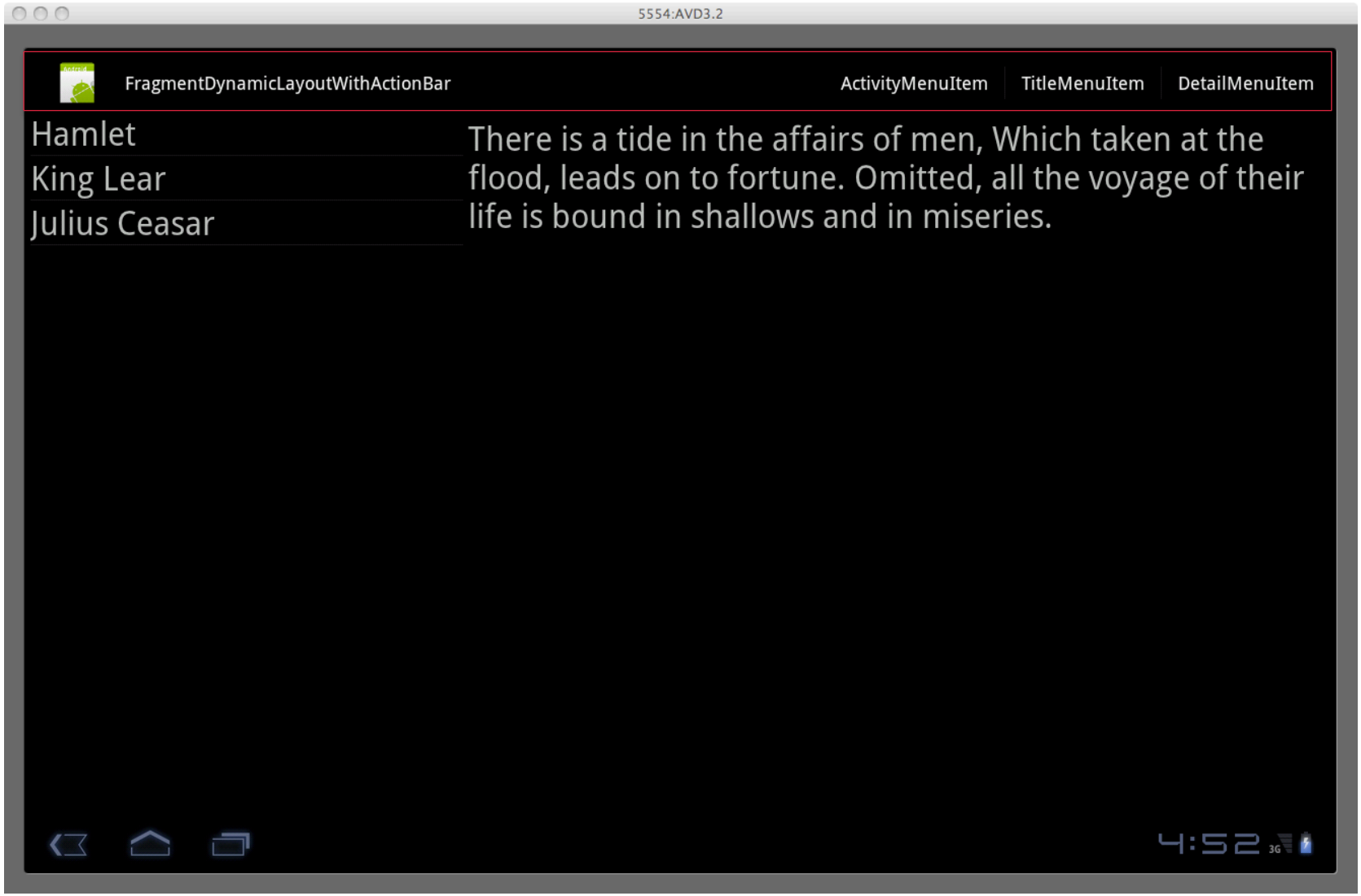
Android Support Library

- Android Support Library allows use of Fragments on pre-Android 3.0 devices
  - API is slightly different
    - Must extend FragmentActivity
    - Must call getSupportFragmentManager()
- Also adds support for other new APIs such as Loaders
- http://developer.android.com/tools/support-library/index.html

Action Bar

- Menu widget for Activities
  - Places the traditional title bar at the top of the screen.
- By default, Action Bar includes
  - Application icon
  - Activity title
  - Items from any Options Menus
- Must use Support Library (version 7) and extend ActionBarActivity for use on older devices

Activity with ActionBar



ActionBar Items

- ActionBar items work like menus
- Will discuss menus in more detail in a later class

activity_menu.xml

```
<menu ...>
  <item android:id="@+id/activity_menu_item"
     android:title="ActivityMenuItem"
      android:showAsAction="always">
  </item>
</menu>
```

DetailsActivity.java

```
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.activity_menu, menu);
  return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.activity_menu_item:
       ...
      return true;
    default:
      return super.onOptionsItemSelected(item);
}
```

DetailsFragment.java

```
...
setHasOptionsMenu(true);
...
public void onCreateOptionsMenu(Menu menu,
                                MenuInflater inflater) {
  inflater.inflate(R.menu.detail_menu, menu);
}

public boolean onOptionsItemSelected(MenuItem item) {
  ...
}
```

Source Code Examples

- FragmentStaticLayout
- FragmentProgrammaticLayout
- FragmentDynamicLayout
- FragmentDynamicLayoutWithActionBar