Andorid Programming

Week 3

Mina Jung

EECS, Syracuse University

Spring 2017

Part I

Fragments, Master/Detail Flow, and ViewPager

Outline I

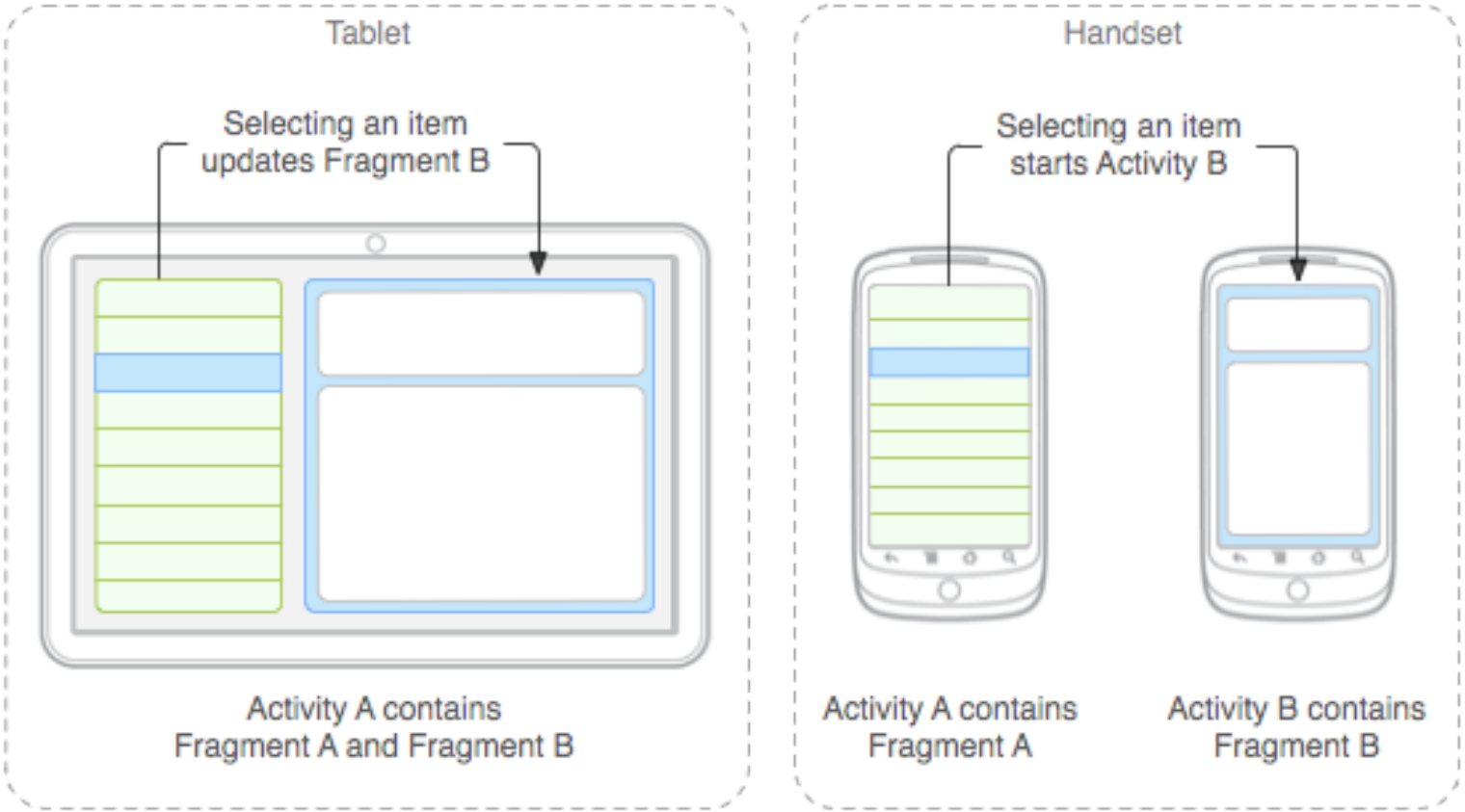
Outline II

Fragment I

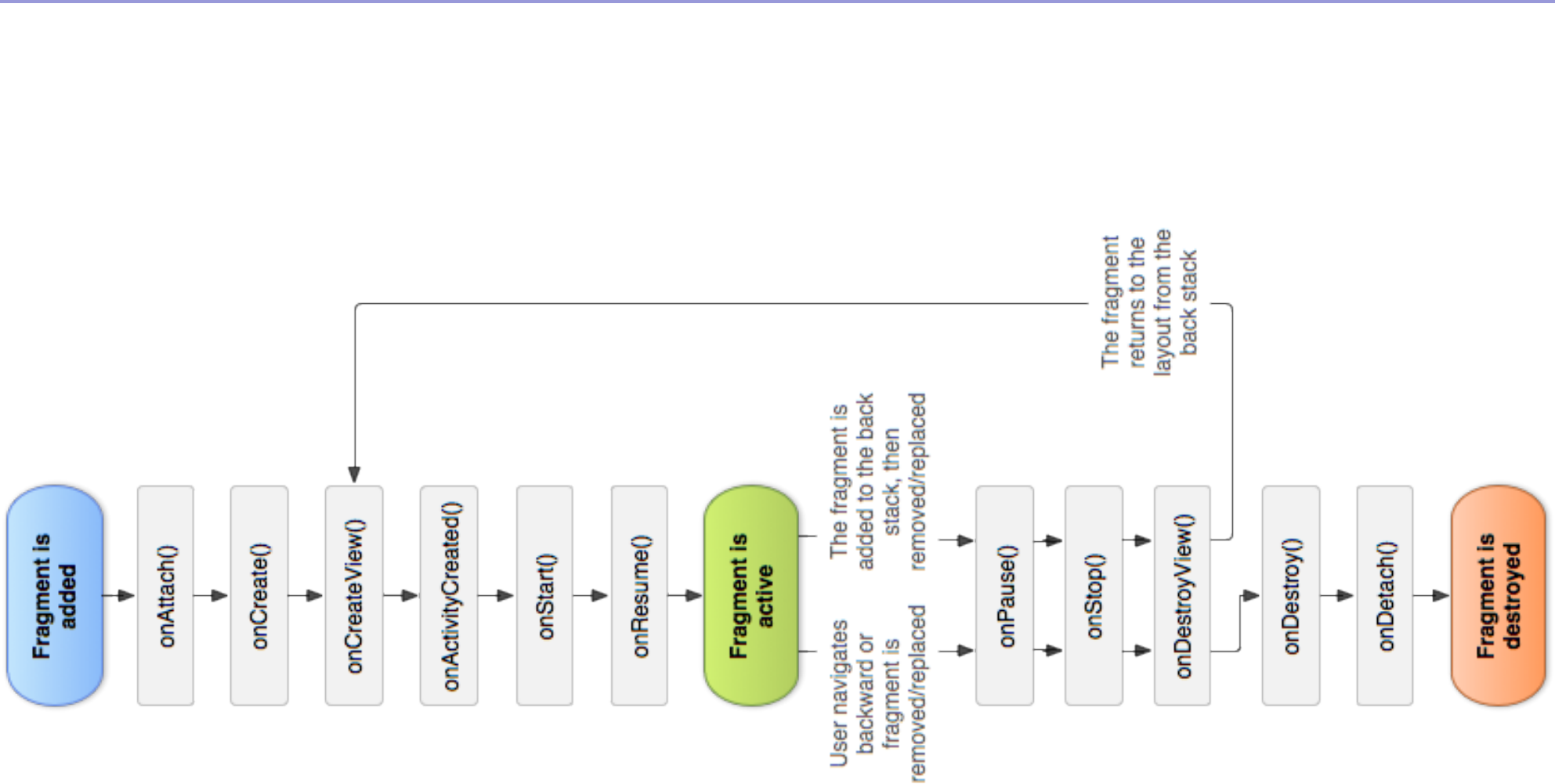
- represents a behavior or a portion of UI in an Activity
  - multiple fragments in a single activity to build multi-pane UI
  - reuse a fragment in multiple activities
- modular section of an activity (like "sub-activity")
  - has its own lifecycle
    - directly affected by the host activity's lifecycle
  - receives its own input events
  - dynamically added or removed while the activity is running
- support more dynamic and flexible UI design on large screens
  - by dividing the layout of an activity into fragments
  - can modify the activity's appearance at runtime

Fragment II

- preserve changes in back stack



- Click Fragment Class


Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
The fragment is added to the back stack, then removed/replaced
User navigates backward or fragment is removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed

should implement lifecycle methods	
onCreate()	should initialize essential components of the fragment that you want to retain when it is paused or stopped, then resumed
onCreateView()	called when it's time for the fragment to draw its user interface for the first time. must return a view that is the root of the fragment' layout or null if the fragment doesnot provide a UI
onPause()	usually where you should commit any changes

1. Decide how many fragments in an activity
2. Based on the number of fragments, create classes which will extend the Fragment class
3. Corresponding to each fragment, create layout files in XML file
4. Modify activity file to define the actual logic of replacing fragments

1. XML layout file (UI) for a new Fragment (fragment_a.xml)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
3  android:orientation="vertical" android:layout_width="match_parent"
4  android:layout_height="match_parent"
5  android:background="#FFAAAA">
6
7  <TextView
8      android:layout_width="match_parent"
9      android:layout_height="wrap_content"
10     android:textSize="30sp"
11     android:text="Fragment A"
12     android:gravity="center" />
13
14 </LinearLayout>
```

2. corresponding Java class (FragmentA.java) extends Fragment class

```
1 public class FragmentA extends Fragment {
2     public FragmentA() {
3         // Required empty public constructor
4     }
5     ...
6 }
```

- provide a layout for a fragment (fragment_a.xml)
- implement onCreateView() callback method

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
     container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_a, container, false);
}
```

- three arguments of inflate() method
  - resource ID of the layout
  - ViewGroup to be the parent of the inflated layout
  - true to create a redundant view group / false (system is already inserting the inflated layout into the container)

- Static way: Declare fragments inside the activity's layout file

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent">
6      <fragment
7          android:layout_width="match_parent"
8          android:layout_height="wrap_content"
9          android:id="@+id/fragmentA"
10         android:name="com.examples.fragmentexample1.FragmentA"
11         android:layout_weight="1"/>
12     <FrameLayout
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:layout_weight="1"
16         android:id="@+id/fragmentBorC" />
17     <Button
18         android:id="@+id/button1"
19         android:layout_width="match_parent"
20         android:layout_height="wrap_content"
21         android:text="Switch Fragment"
22         android:textSize="20sp" />
```

```
23 </LinearLayout>
```

- **android:name** attribute in the <fragment> specifies the Fragment class to instantiate in the layout

- Dynamic way: Add the fragment to an existing ViewGroup in the Activity programmatically
  1. Get an instance of FragmentTransaction from the activity

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction =
    fragmentManager.beginTransaction();
```

  2. Add a fragment using add() method, then MUST call commit()

```
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    FragmentManager fm = getFragmentManager();
    FragmentTransaction fragmentTransaction =
        fm.beginTransaction();
    fragmentTransaction.add(R.id.fragmentBorC, new FragmentB());
    fragmentTransaction.commit();

    // ...
}
```

- Use **FragmentManager**
  - call **getFragmentManager**() from the activity
- Get fragments
  - **findFragmentById**() (for fragments with UI)
  - **findFragmentByTag**() (for fragments wihtout UI)
- Pop fragments off the back stack with **popBackStack**()
- Register a listener for changes to the back stack with **addOnBackStackChangedListener**()
- Click FragmentManager Class

- ability to add, remove, replace, and perform other actions with fragments
- First, acquire an instance of FragmentTransaction

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction =
    fragmentManager.beginTransaction();
```

- transaction using methods such as add(), remove(), and replace(), then must call commit()
  - addToBackStack(): add the transaction to a back stack of fragment transactions, which allows the user to return to the previous fragment state

```
// 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();
```

- Example: when a button is clicked, change Fragments in FrameLayout

```
public class MainActivity extends AppCompatActivity {

    private boolean isFragmentB = true ;

    // ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // ...

        Button button1 = (Button) findViewById(R.id.button1) ;
        button1.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                switchFragment() ;
            }
        });
    }

    public void switchFragment() {
        Fragment fr;

        if (isFragmentB) {
            fr = new FragmentB() ;
        } else {
            fr = new FragmentC() ;
        }

        isFragmentB = (isFragmentB) ? false : true ;

        FragmentManager fm = getFragmentManager();
        FragmentTransaction fragmentTransaction =
            fm.beginTransaction();
        fragmentTransaction.replace(R.id.fragmentBorC, fr);
        fragmentTransaction.commit();
    }
}
```

- Fragment constructor cannot take any argument
- can use setArgument() only before the fragment is attached to Activity
- Android prefers **static newInstance()** method
  1. create a Fragment object
  2. set an argument
  3. return argument

```
public class MyFragment extends Fragment {
    private String name;
    private int age;

    private TextView mNameTextView;
    private TextView mAgeTextView;

    public static MyFragment newInstance(String name, int age) {
        Bundle bundle = new Bundle();
        bundle.putString("name", name);
        bundle.putInt("age", age);

        MyFragment fragment = new MyFragment();
        fragment.setArguments(bundle);

        return fragment;
    }

    private void readBundle(Bundle bundle) {
        if (bundle != null) {
            name = bundle.getString("name");
            age = bundle.getInt("age");
        }
    }

    @Override
```

```
    public View onCreateView(LayoutInflater inflater, ViewGroup
         container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_sample,
             container, false);
        mNameTextView = (TextView)
             view.findViewById(R.id.nameTextView);
        mAgeTextView = (TextView)
             view.findViewById(R.id.ageTextView);

        readBundle(getArguments());

        mNameTextView.setText(String.format("Name: %s", name));
        mAgeTextView.setText(String.format("Age: %d", age));

        return view;
    }
}
```

- use Bundle to pass Data (Click)
- use Bundle to pass an Object using marshaling/unmarshaling
  - Parcelable
    - custom code for marshaling and unmarshaling
    - less garbage objects
    - better performance (2xfaster than serialization)
  - Serialization
    - marker interface
    - marshaling performed on JVM (slow)

- A Fragment is implemented as an object that is independent from an Activity
  - can be used inside multiple activities
  - a given instance of the fragment is directly tied to the activity that contains it

- access from Fragment to Activity

```
TextView txtView = getActivity().findViewById(R.id.txtView);
```

- access from Activity to Fragment after acquiring a reference to the Fragment

```
FragmentA fragment =
(FragmentA)getFragmentManager().findFragmentById(R.id.fragment_a);
```

- a fragment share events with its host activity
  1. Define a callback interface inside the fragment
  2. Host activity implements the interface

- no communication between fragments
  - must be through the host activity

- communication between Activities
  - use Intent!! (Click Common Intent)

- Example: two fragments share events through its host activity
  - FirstFragment with two buttons
  - SecondFragment with a TextView – when a button is clicked, the TextView display messages

1. FirstFragment.java

```
public class FirstFragment extends Fragment {
    Button mButton1;
    Button mButton2;

    public interface CustomOnClickListener {
        public void onClicked( View v );
    }

    private CustomOnClickListener customOnClickListener;

    @Override
    public View onCreateView( LayoutInflater inflater, ViewGroup
         container, Bundle savedInstanceState ) {
        View view = inflater.inflate( R.layout.first_fragment,
            container, false );
        mButton1 = (Button)view.findViewById( R.id.button1 );
        mButton2 = (Button)view.findViewById( R.id.button2 );
        mButton1.setOnClickListener( new View.OnClickListener() {
            public void onClick( View v ) { buttonClicked( v ); } }
            );
        mButton2.setOnClickListener( new View.OnClickListener() {
            public void onClick( View v ) { buttonClicked( v ); } }
            );
        return view;
    }
```

```
    public void buttonClicked( View v ) {
        customOnClickListener.onClicked(v);
    }

    @Override
    @Deprecated
    public void onAttach( Activity activity ) {
        super.onAttach( activity );
        customOnClickListener = (CustomOnClickListener)activity;
    }
}
```

2. SecondFragment.java

```
public class SecondFragment extends Fragment {
    TextView mTextView1;

    @Override
    public View onCreateView( LayoutInflater inflater, ViewGroup
         container, Bundle savedInstanceState ) {
        View view = inflater.inflate( R.layout.second_fragment,
            container, false );
        mTextView1 = (TextView)view.findViewById( R.id.textView1 );
        return view;
    }

    public void setText( String text ) {
        mTextView1.setText( text );
    }
}
```

3. HostActivity.java

```
public class HostActivity extends AppCompatActivity implements
     FirstFragment.CustomOnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_host);

        ....
    }

    ....

    @Override
    public void onClicked( View v )  {
        FragmentManager fragmentManager = getFragmentManager();
        SecondFragment secondFragment =
             (SecondFragment)fragmentManager.findFragmentById(
             R.id.fragment2 );
        switch( v.getId() ) {
            case R.id.button1: { secondFragment.setText( "Button1
                 was clicked." ); break; }
            case R.id.button2: { secondFragment.setText( "Button2
                 was clicked." ); break; }
        }
```

```
        }
    }
```

- Retaining an Object During a Configuration Change
  1. Extend the Fragment class and declare references to your stateful objects.
  2. Call setRetainInstance(boolean) when the fragment is created.
  3. Add the fragment to your activity.
  4. Use FragmentManager to retrieve the fragment when the activity is restarted.

```
//+++++++++++++++++++++++++++
public class RetainedFragment extends Fragment {

    // data object we want to retain
    private MyDataObject data;

    // this method is only called once for this fragment
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // retain this fragment
        setRetainInstance(true);
    }
```

```
    public void setData(MyDataObject data) {
        this.data = data;
    }

    public MyDataObject getData() {
        return data;
    }
}
//++++++++++++++++++++++++++++
public class MyActivity extends Activity {

    private static final String TAG_RETAINED_FRAGMENT =
        "RetainedFragment";

    private RetainedFragment mDataFragment;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find the retained fragment on activity restarts
        FragmentManager fm = getFragmentManager();
        mDataFragment = (DataFragment)
            fm.findFragmentByTag(TAG_RETAINED_FRAGMENT);
```

```
        // create the fragment and data the first time
        if (mDataFragment == null) {
            // add the fragment
            mDataFragment = new DataFragment();
            fm.beginTransaction().add(mDataFragment,
                    TAG_RETAINED_FRAGMENT).commit();
            // load data from a data source or perform any
                calculation
            mDataFragment.setData(loadMyData());
        }

        // the data is available in mDataFragment.getData() even
            after subsequent configuration change restarts.
        ...
    }

    @Override
    public void onPause() {
        // perform other onPause related actions
        ...
        // this means that this activity will not be recreated now,
            user is leaving it
        // or the activity is otherwise finishing
        if(isFinishing()) {
            FragmentManager fm = getFragmentManager();
```

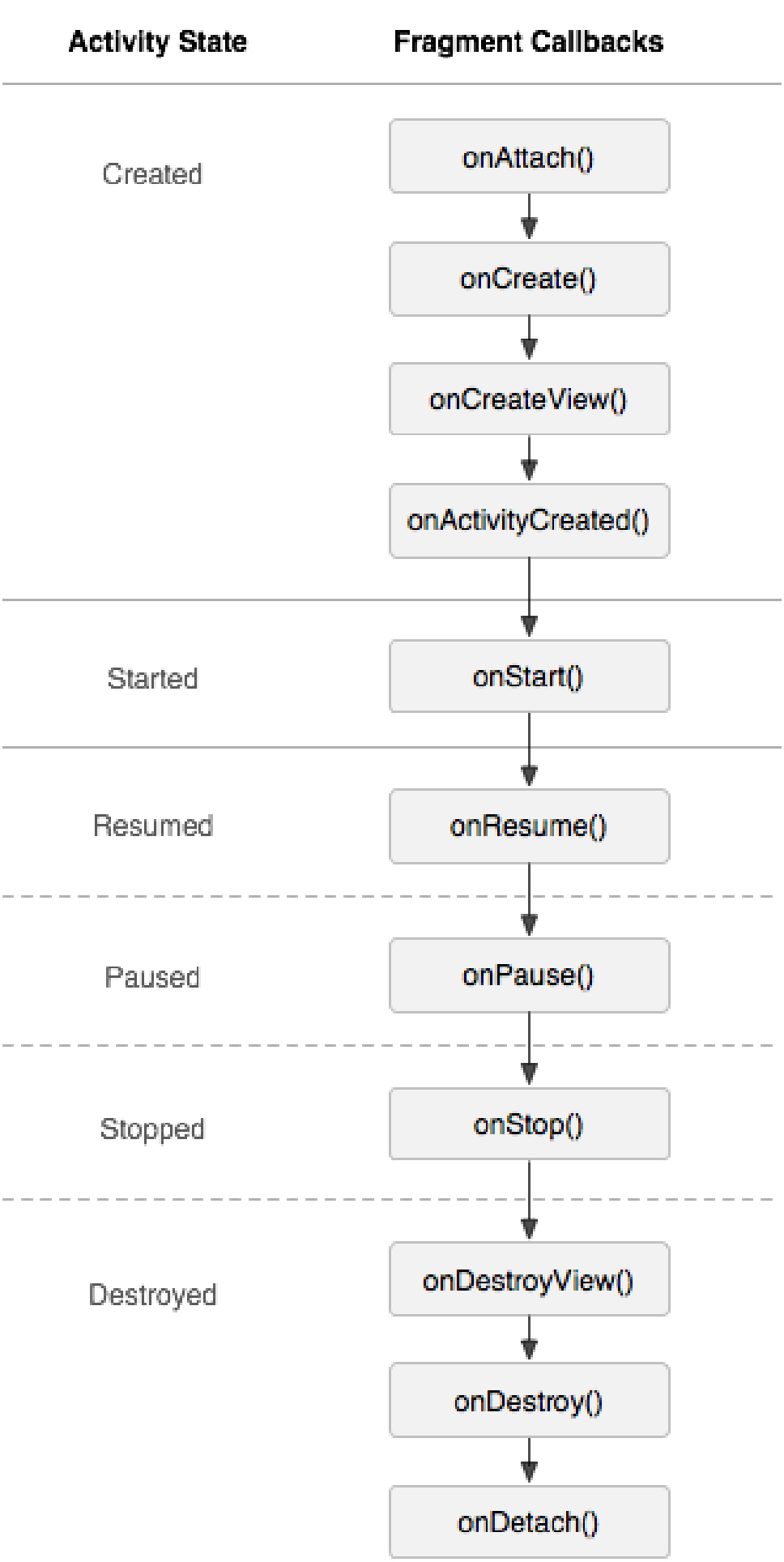
```
                // we will not need this fragment anymore, this may
                    also be a good place to signal
                // to the retained fragment object to perform its own
                    cleanup.
                fm.beginTransaction().remove(mDataFragment).commit();
            }
        }
    }
```

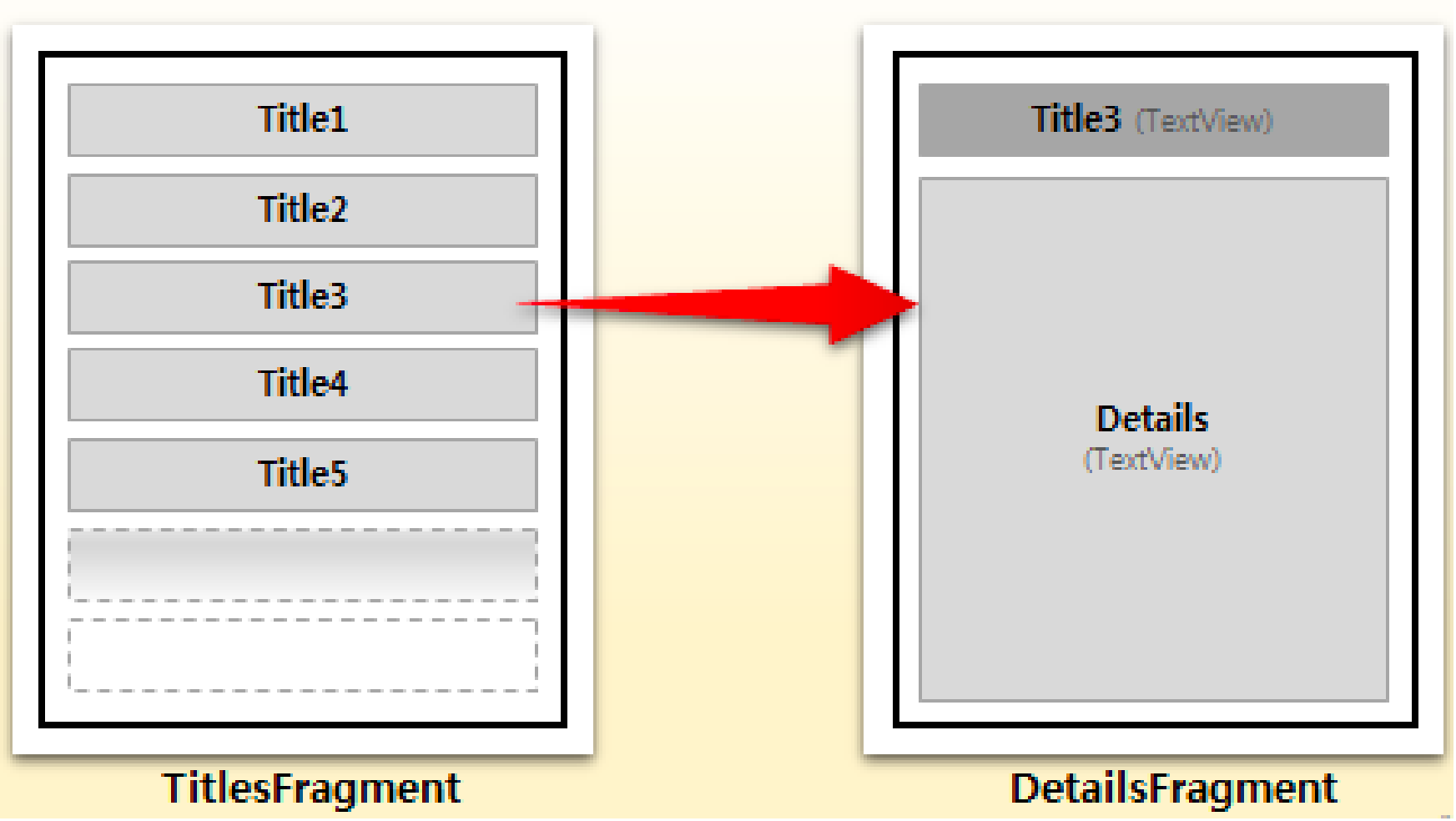
- Handling the Configuration Change Yourself
  - Edit the appropriate <activity> element in the manifest file to include the **android:configChanges** attribute with a value that represents the configuration you want to handle

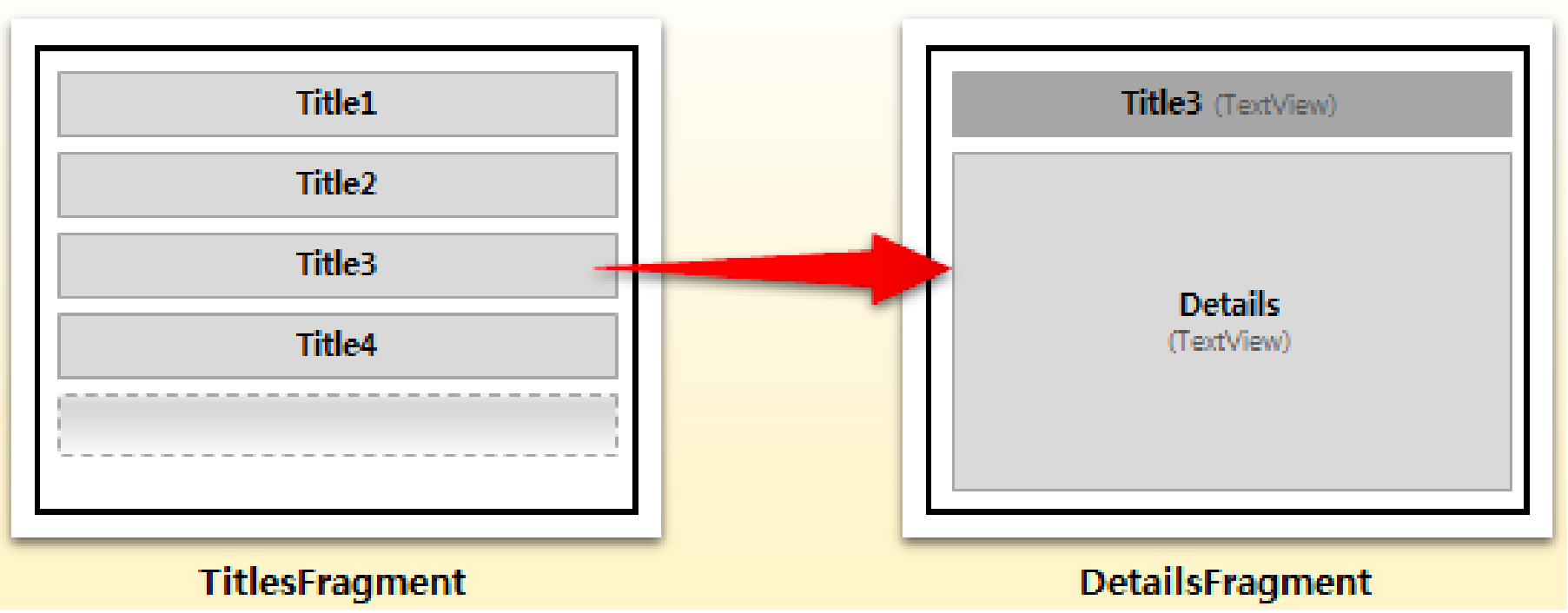
```
<activity android:name=".MyActivity"
          android:configChanges="orientation|keyboardHidden"
          android:label="@string/app_name">
```

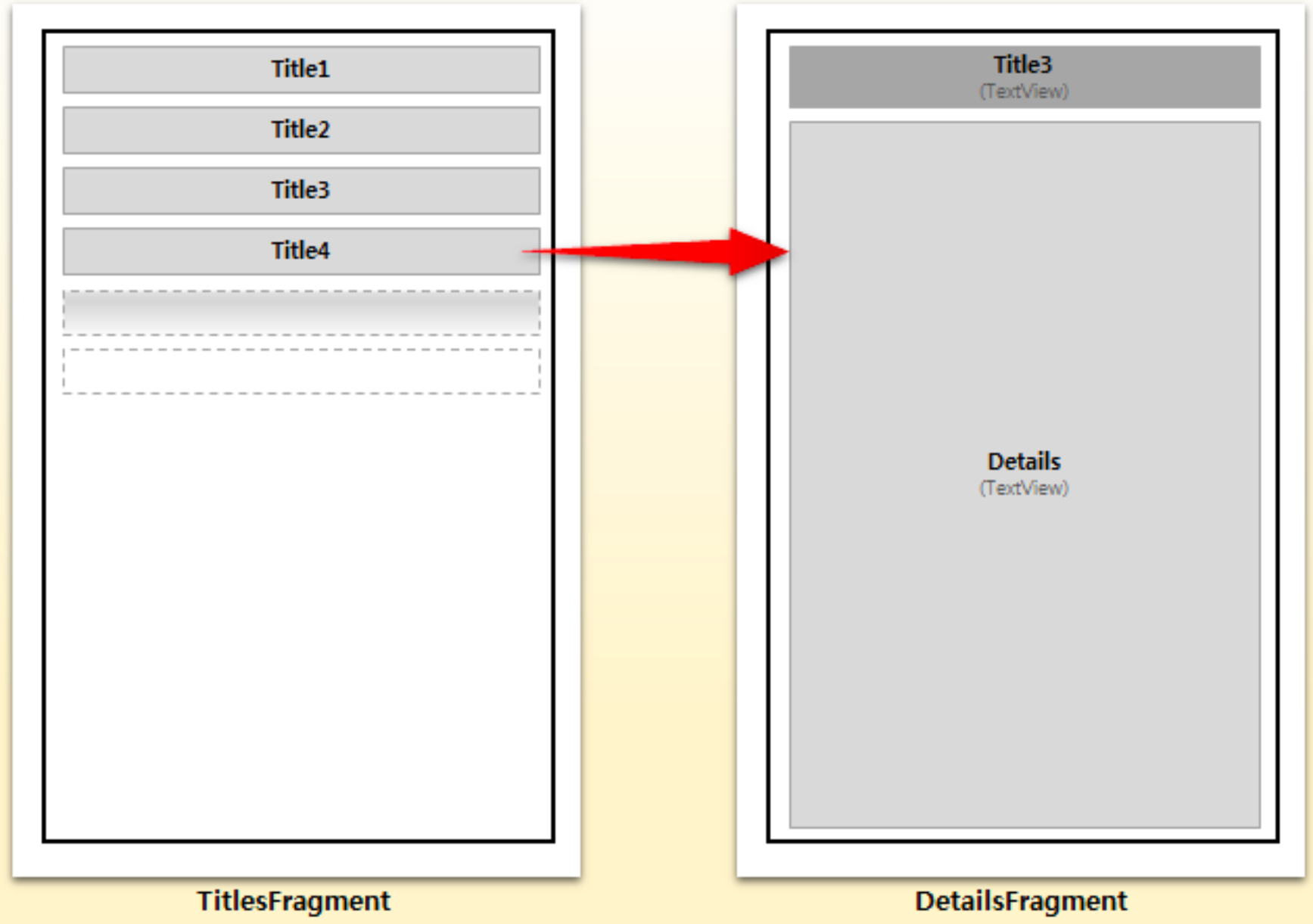
```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
    if (newConfig.orientation ==
         Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape",
             Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation ==
         Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

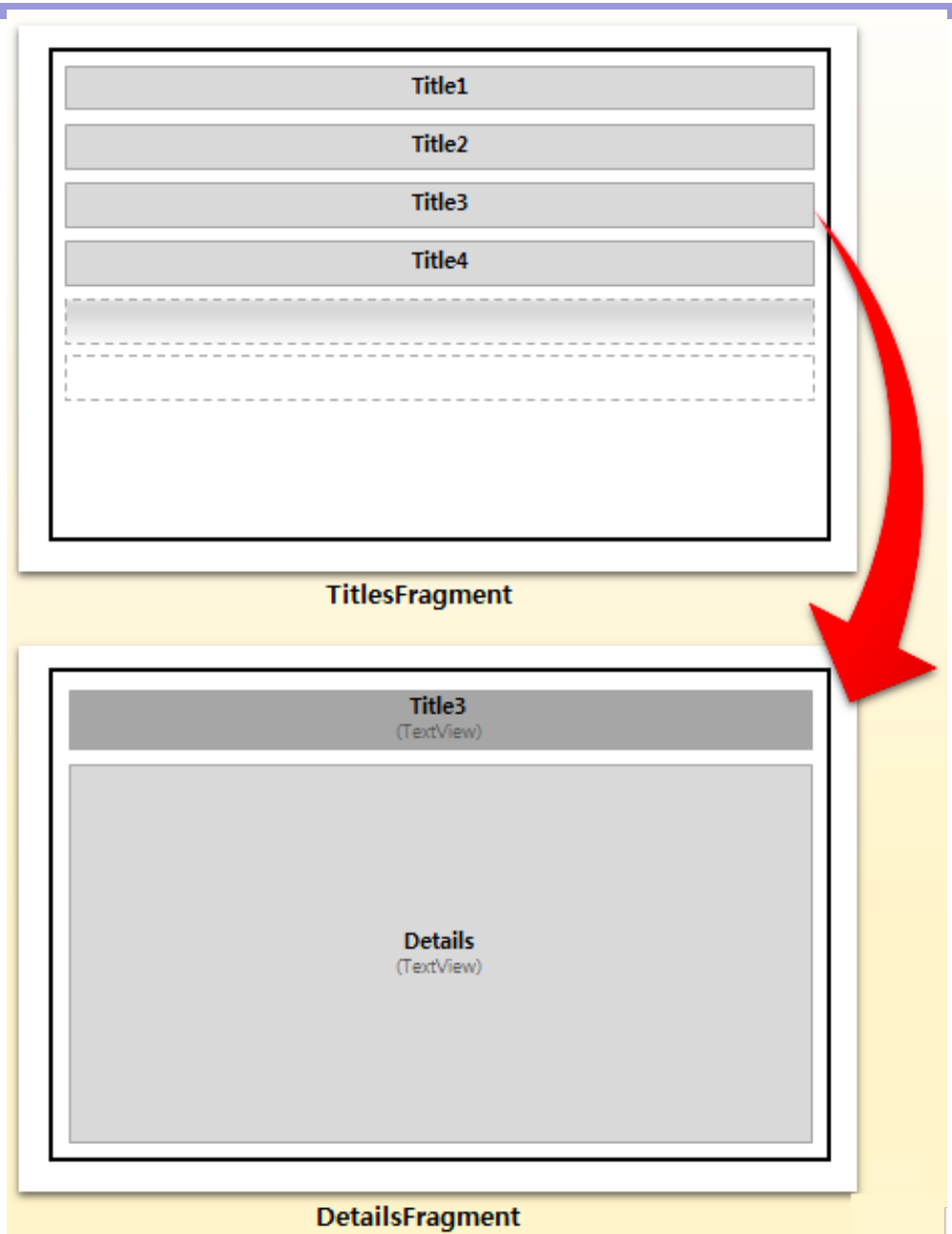
- Save the current Fragment using savedInstanceState and onSaveInstanceState() callback method

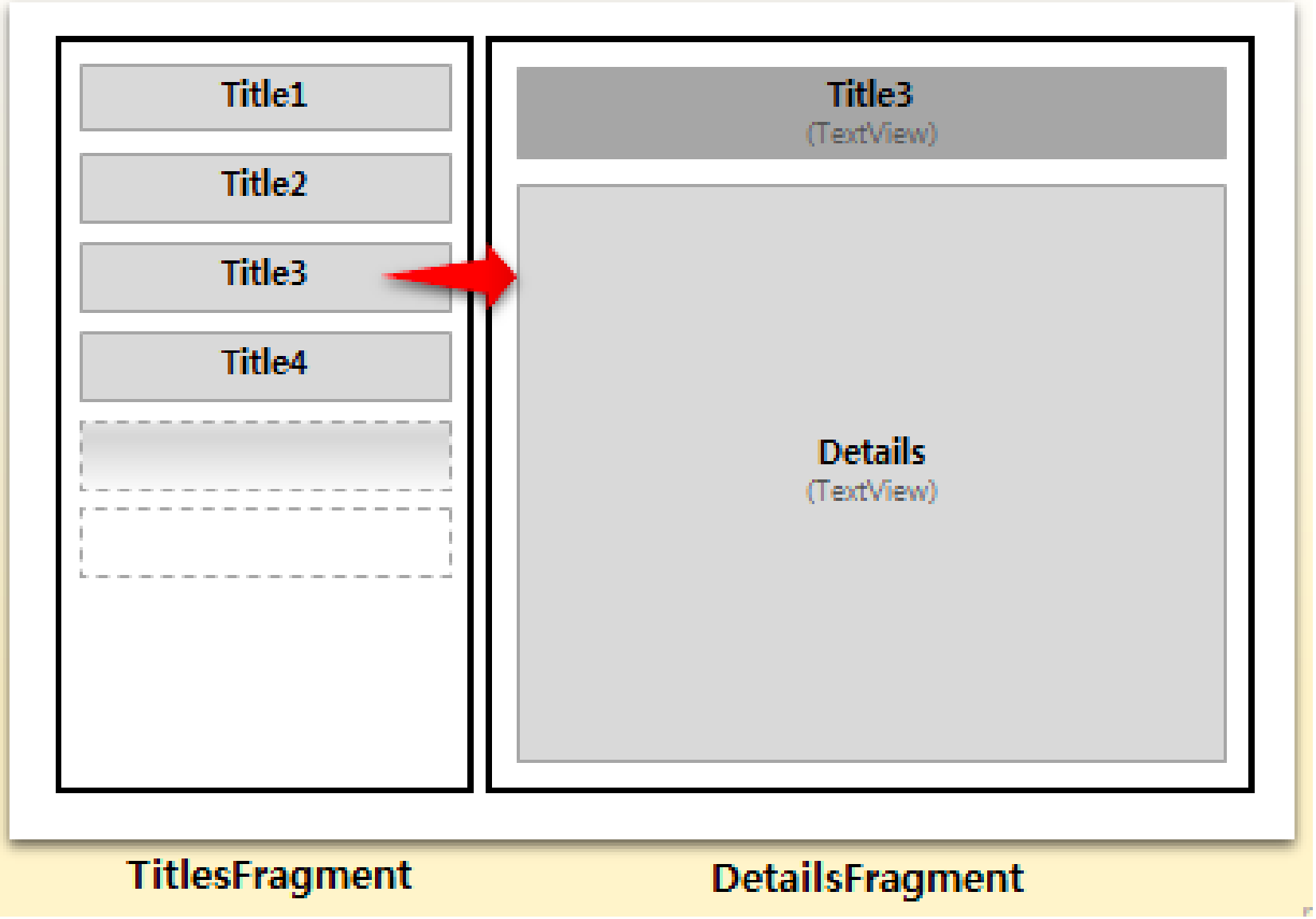

Activity State
Fragment Callbacks
Created
onAttach()
onCreate()
onCreateView()
onActivityCreated()
Started
onStart()
Resumed
onResume()
Paused
onPause()
Stopped
onStop()
Destroyed
onDestroyView()
onDestroy()
onDetach()


Title1
Title2
Title3
Title4
Title5
TitlesFragment
Title3 (TextView)
Details
(TextView)
DetailsFragment


Title1
Title2
Title3
Title4
Title3 (TextView)
Details
(TextView)
TitlesFragment
DetailsFragment


Title1
Title2
Title3
Title4
Title3
(TextView)
Details
(TextView)
TitlesFragment
DetailsFragment


Title1
Title2
Title3
Title4
TitlesFragment
Title3
(TextView)
Details
(TextView)
DetailsFragment


Title1
Title2
Title3
Title4
Title3
(TextView)
Details
(TextView)
TitlesFragment
DetailsFragment

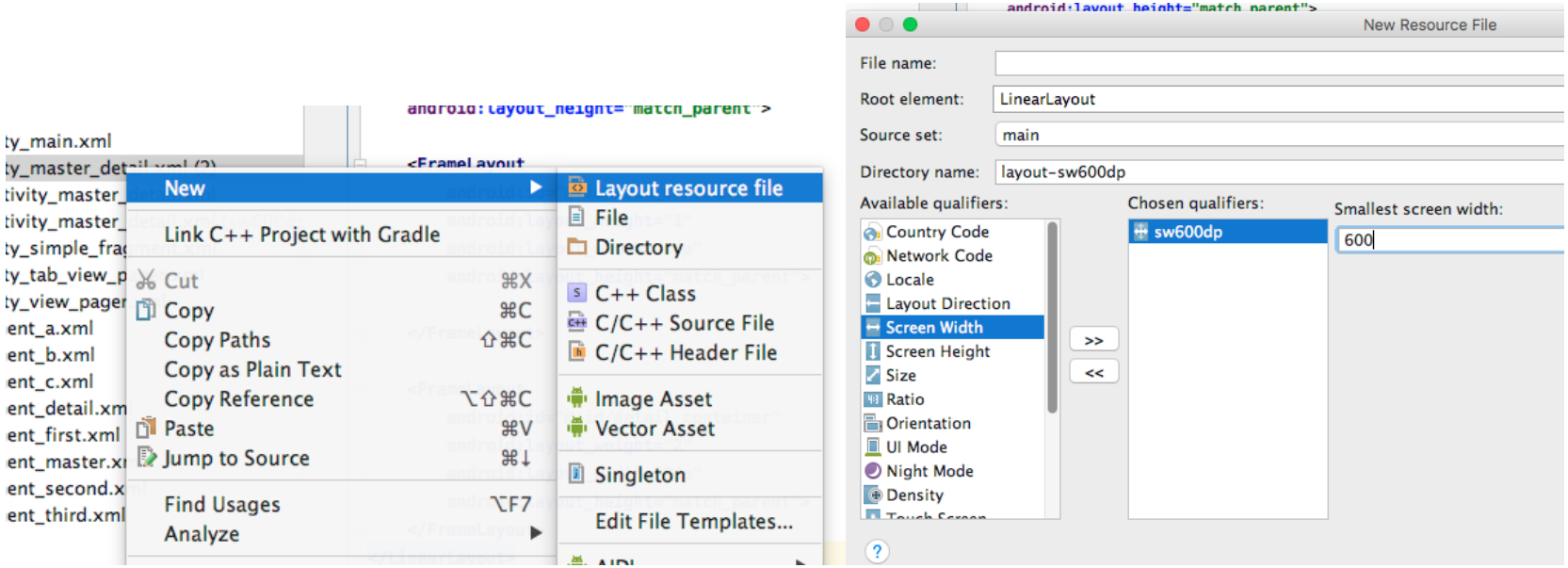
Questions?

- How do we use the same code to inflate two different layouts?
- How do we know whether we are in the Tablet style or Handset style?

- Same Layout name with different qualifiers
- Handset Layout
  - vertical orientation
  - single container for fragments

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <FrameLayout
        android:id="@+id/main_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </FrameLayout>
</LinearLayout>
```

- Tablet Layout (sw600dp)



  - horizontal orientation
  - two containers for fragments in Activity

```
<LinearLayout
    android:orientation="horizontal"
        android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/main_container"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent">
    </FrameLayout>

    <FrameLayout
        android:id="@+id/detail_container"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent">
    </FrameLayout>
</LinearLayout>
```

- Check the second container is present in the current activity

```
boolean mTwoPane;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_master_detail);

    ...

    FragmentMaster fr = new FragmentMaster();
    getSupportFragmentManager().beginTransaction().replace(R.id.ma
        fr).commit();

    if(findViewById(R.id.detail_container) != null){
        mTwoPane = true;
    }
}
```

- Use getResource().getConfiguration() method
  - get information from the returned configuration object

```
if(conf.isLayoutSizeAtLeast(Configuration.SCREENLAYOUT_SIZE_LARGE)){
 // Check Tablet
}

if(conf.orientation == Configuration.ORIENTATION_LANDSCAPE){
 // check mode
}
```

```
@Override
public void onClicked(View v){
    Button b = (Button)findViewById(v.getId());
    String t = b.getText().toString();

    if(mTwoPane){
        getSupportFragmentManager().beginTransaction().replace(R.id.detail_co
            FragmentDetail.newInstance(t)).addToBackStack(null).commit();
    }
    else{
        getSupportFragmentManager().beginTransaction().replace(R.id.main_conta
            FragmentDetail.newInstance(t)).addToBackStack(null).commit();
    }
}
```

- layout widget
- can create swipe views
- available in Support Library
- add <ViewPager> element to layout XML

```
<android.support.v4.view.ViewPager
    android:id="@+id/vp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</android.support.v4.view.ViewPager>
```

- To insert child views, need to hook the layout to a PagerAdapter

- populate pages inside of a ViewPager
- Click PagerAdapter
- FragmentPagerAdapter (Click here!)
  - best for a fixed, small number of pages, such as a set of tabs
  - fragment of each page will be kept in memory
  - its view may be destroyed when not visible
- FragmentStatePagerAdapter (Click here!!!)
  - best for paging across a collection of objects
  - number of pages is not determined
  - fragments may be destroyed when not visible (when user navigates to other pages)
  - minimize memory usage

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      android:fitsSystemWindows="true"
9      tools:context="com.example.mina.third.TabViewPagerActivity">
10
11     <android.support.design.widget.AppBarLayout
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:theme="@style/AppTheme.AppBarOverlay">
15
16         <android.support.v7.widget.Toolbar
17             android:id="@+id/toolbar"
18             android:layout_width="match_parent"
19             android:layout_height="?attr/actionBarSize"
20             android:background="?attr/colorPrimary"
21             app:popupTheme="@style/AppTheme.PopupOverlay" />
22
23     </android.support.design.widget.AppBarLayout>
24
25
```

```
26      <LinearLayout
27          android:id="@+id/ll"
28          android:orientation="horizontal"
29          android:layout_width="match_parent"
30          android:layout_height="wrap_content">
31
32          <TextView
33              android:id="@+id/tab_first"
34              android:layout_width="0dip"
35              android:layout_height="50dp"
36              android:layout_weight="1"
37              android:gravity="center"
38              android:textColor="@drawable/tab_color_selector"
39              android:background="@drawable/tab_bg_selector"
40              android:text="First Tab" />
41
42          <TextView
43              android:id="@+id/tab_second"
44              android:layout_width="0dip"
45              android:layout_height="50dp"
46              android:layout_weight="1"
47              android:gravity="center"
48              android:textColor="@drawable/tab_color_selector"
49              android:background="@drawable/tab_bg_selector"
50              android:text="Second Tab" />
51
```

```
52          <TextView
53              android:id="@+id/tab_third"
54              android:layout_width="0dip"
55              android:layout_height="50dp"
56              android:layout_weight="1"
57              android:gravity="center"
58              android:textColor="@drawable/tab_color_selector"
59              android:background="@drawable/tab_bg_selector"
60              android:text="Third Tab" />
61      </LinearLayout>
62
63      <android.support.v4.view.ViewPager
64          android:id="@+id/tvp"
65          android:layout_width="match_parent"
66          android:layout_height="match_parent">
67      </android.support.v4.view.ViewPager>
68  </LinearLayout>
```

activity_tab_view_pager.xml

```
 1 package com.example.mina.third;
 2
 3 import android.os.Bundle;
 4 import android.support.v4.app.FragmentStatePagerAdapter;
 5 import android.support.v4.view.ViewPager;
 6 import android.support.v7.app.AppCompatActivity;
 7 import android.support.v7.widget.Toolbar;
 8 import android.view.View;
 9 import android.widget.LinearLayout;
10 import android.widget.TextView;
11
12 public class TabViewPagerActivity extends AppCompatActivity {
13
14     ViewPager vp;
15     LinearLayout ll;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_tab_view_pager);
21         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
22  //         toolbar.setTitle("ViewPager with Button Tab");
23         setSupportActionBar(toolbar);
24
25         vp = (ViewPager)findViewById(R.id.tvp);
26         ll = (LinearLayout)findViewById(R.id.ll);
```

```
27
28          TextView tab_first = (TextView)findViewById(R.id.tab_first);
29          TextView tab_second =
                (TextView)findViewById(R.id.tab_second);
30          TextView tab_third = (TextView)findViewById(R.id.tab_third);
31
32          vp.setAdapter(new
                MyPagerAdapter(getSupportFragmentManager()));
33          vp.setCurrentItem(0);
34
35          tab_first.setOnClickListener(mPageListener);
36          tab_first.setTag(0);
37          tab_second.setOnClickListener(mPageListener);
38          tab_second.setTag(1);
39          tab_third.setOnClickListener(mPageListener);
40          tab_third.setTag(2);
41
42
43          tab_first.setSelected(true);
44
45          vp.addOnPageChangeListener(new
                ViewPager.OnPageChangeListener()
46          {
47              @Override
48              public void onPageScrolled(int position, float
                    positionOffset, int positionOffsetPixels)
```

```
49              {
50
51              }
52
53              @Override
54              public void onPageSelected(int position)
55              {
56                  int i = 0;
57                  while(i<3)
58                  {
59                      if(position==i)
60                      {
61                          ll.findViewWithTag(i).setSelected(true);
62                      }
63                      else
64                      {
65                          ll.findViewWithTag(i).setSelected(false);
66                      }
67                      i++;
68                  }
69              }
70
71              @Override
72              public void onPageScrollStateChanged(int state)
73              {
74
```

```
75              }
76          });
77
78          vp.setPageTransformer(false, new
                ViewPager.PageTransformer(){
79              @Override
80              public void transformPage(View page, float position){
81                  final float normalized_position =
                        Math.abs(Math.abs(position)-1);
82                  page.setScaleX(normalized_position/2 + 0.5f);
83                  page.setScaleY(normalized_position/2 + 0.5f);
84              }
85          });
86      }
87
88      View.OnClickListener mPageListener = new View.OnClickListener()
89      {
90          @Override
91          public void onClick(View v)
92          {
93              int tag = (int) v.getTag();
94
95              int i = 0;
96              while(i<3)
97              {
98                  if(tag==i)
```

```
 99                  {
100                      ll.findViewWithTag(i).setSelected(true);
101                  }
102                  else
103                  {
104                      ll.findViewWithTag(i).setSelected(false);
105                  }
106                  i++;
107              }
108
109              vp.setCurrentItem(tag);
110          }
111      };
112
113      private class MyPagerAdapter extends FragmentStatePagerAdapter
114      {
115          public
                 MyPagerAdapter(android.support.v4.app.FragmentManager
                 fm)
116          {
117              super(fm);
118          }
119          @Override
120          public android.support.v4.app.Fragment getItem(int position)
121          {
122              switch(position)
```

```
123                 {
124                     case 0:
125                         return new FragmentFirst();
126                     case 1:
127                         return new FragmentSecond();
128                     case 2:
129                         return new FragmentThird();
130                     default:
131                         return null;
132                 }
133             }
134             @Override
135             public int getCount()
136             {
137                 return 3;
138             }
139         }
140 }
```

TabViewPagerActivity.java