

# Hello Android!

Dynamic UIs using Fragments

# Fragments

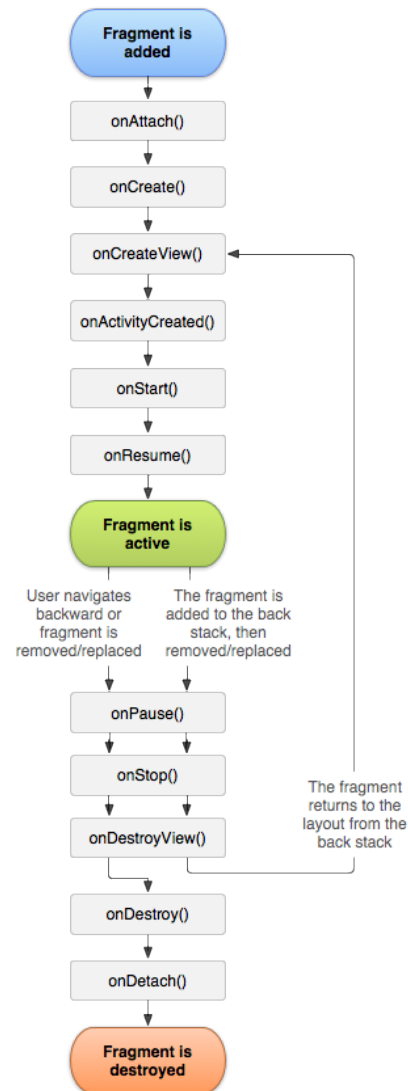
Fragments are:

- Reusable, modular UI components.
- For creating dynamic and multi-pane interfaces.
- Can be swapped into/out of activities based on things like screen size.

Each instance of a Fragment class must be associated with a parent Activity.

If Activity is a controller, then Fragments are mini-controllers.

# Fragment Lifecycle



# Associating Fragment with Activity

Fragment can be associated with parent activity:

- Using the activity's layout XML file.
- Added dynamically at run-time

Fragments do not need to be registered in the manifest. (because they can **\*not\*** exist on their own. They are embedded in other activities).

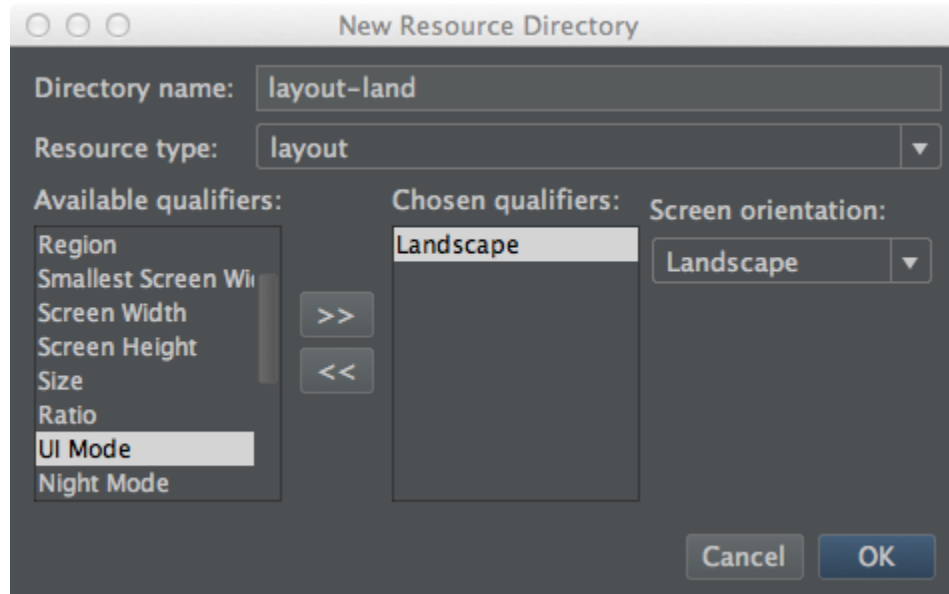
# Demo: Fragments using XML

**Source:** <https://github.com/almalkawi/AndroidClass/tree/master/FragmentsXML>

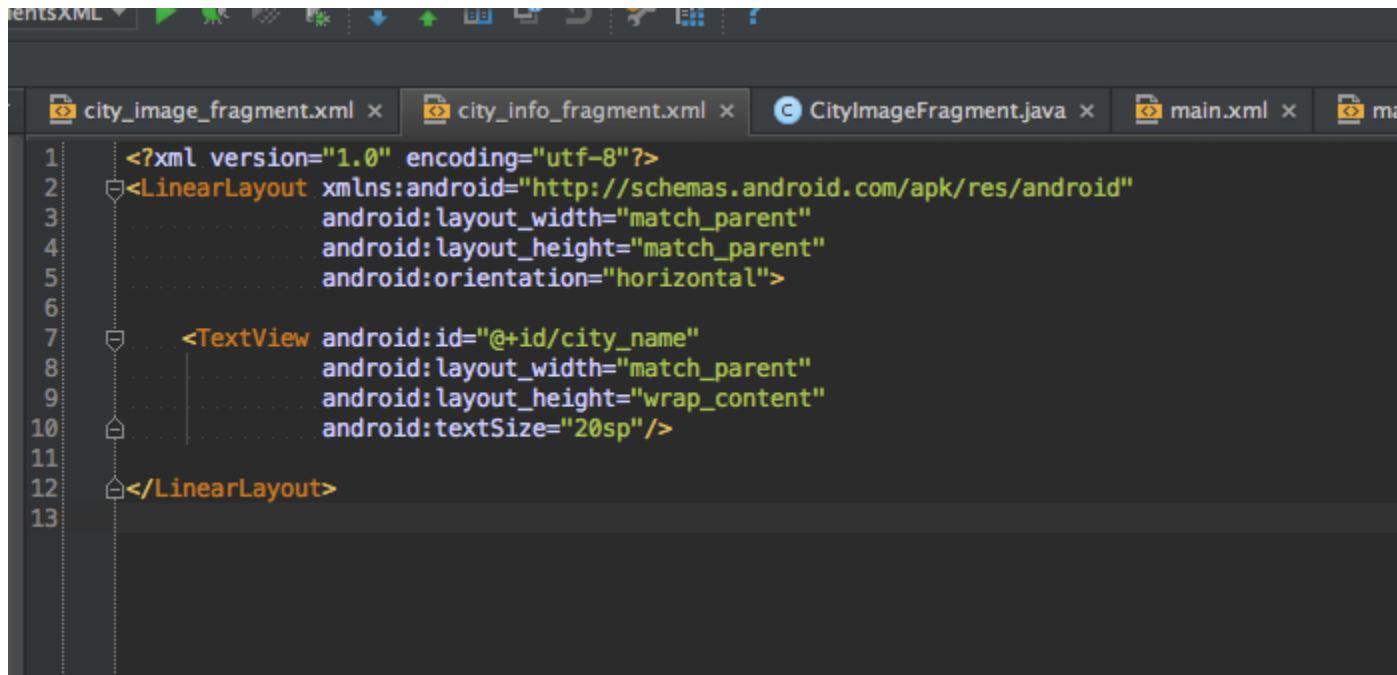
Demo demonstrates handling different screen sizes/orientation.

Run project in emulator and observe lifecycle of Activity and Fragment through Toast messages.

# Create landscape layout



# CityInfo Fragment UI (city\_info\_fragment.xml)



The screenshot shows the Android Studio interface with the XML editor open for the file `city_info_fragment.xml`. The editor displays the following XML code:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                android:layout_width="match_parent"
4                android:layout_height="match_parent"
5                android:orientation="horizontal">
6
7      <TextView android:id="@+id/city_name"
8                android:layout_width="match_parent"
9                android:layout_height="wrap_content"
10               android:textSize="20sp"/>
11
12 </LinearLayout>
13
```

The code defines a horizontal `LinearLayout` containing a single `TextView` with the ID `@+id/city_name`. The `TextView` has a width of `match_parent`, a height of `wrap_content`, and a text size of `20sp`.

# CityInfo Fragment (CityInfoFragment.java)

The screenshot displays the CityInfoFragment.java file in an IDE. The left sidebar shows the project structure, including the 'me.almalkawi.FragmentsXML' package. The main editor shows the code for the CityInfoFragment class, which extends Fragment. Red arrows point from text annotations on the right to specific methods in the code.

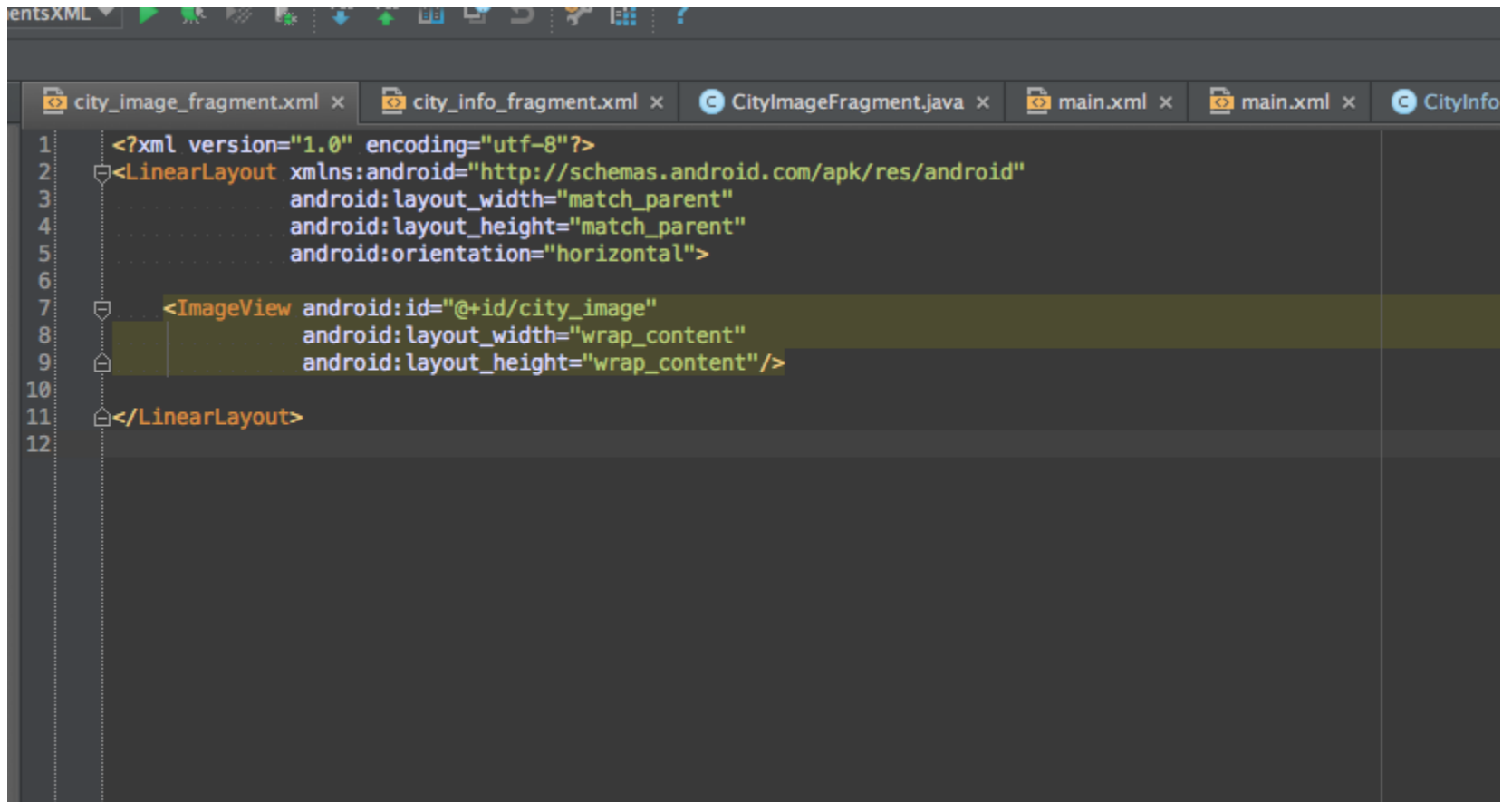
```
public class CityInfoFragment extends Fragment {  
    @Override  
    public void onAttach(Activity activity) {  
        super.onAttach(activity);  
        Toast.makeText(getActivity(), "Fragment: onAttach", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        Toast.makeText(getActivity(), "Fragment: onCreate", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
        Toast.makeText(getActivity(), "Fragment: onCreateView", Toast.LENGTH_SHORT).show();  
        final View ui = inflater.inflate(R.layout.city_info_fragment, container, false);  
        cityName = (TextView) ui.findViewById(R.id.city_name);  
        cityName.setText("Chicago");  
        return ui; // If the fragment has no UI, we return null.  
    }  
  
    @Override  
    public void onActivityCreated(Bundle savedInstanceState) {  
        super.onActivityCreated(savedInstanceState);  
        Toast.makeText(getActivity(), "Fragment: onActivityCreated", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onStart() {  
        super.onStart();  
        Toast.makeText(getActivity(), "Fragment: onStart", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onResume() {  
        super.onResume();  
        Toast.makeText(getActivity(), "Fragment: onResume", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onPause() {  
        super.onPause();  
        Toast.makeText(getActivity(), "Fragment: onPause", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onStop() {  
        super.onStop();  
        Toast.makeText(getActivity(), "Fragment: onStop", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onDestroyView() {  
        super.onDestroyView();  
        Toast.makeText(getActivity(), "Fragment: onDestroyView", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onDetach() {  
        super.onDetach();  
        Toast.makeText(getActivity(), "Fragment: onDetach", Toast.LENGTH_SHORT).show();  
    }  
}
```

Annotations and their corresponding methods:

- Called when Fragment is attached to its parent Activity (usually used to get reference to parent)** points to `onAttach`.
- Used to create Fragment's UI** points to `onCreateView`.
- Inflate UI** points to the `inflater.inflate` call within `onCreateView`.
- Called once the parent Activity and the Fragment's UI have been created** points to `onActivityCreated`.
- Same meaning as in Activity lifecycle** points to `onStart`, `onResume`, `onPause`, and `onStop`.
- Equivalent to Activity's onDestroy** points to `onDestroyView`.
- Called when the Fragment has been detached from its parent Activity** points to `onDetach`.

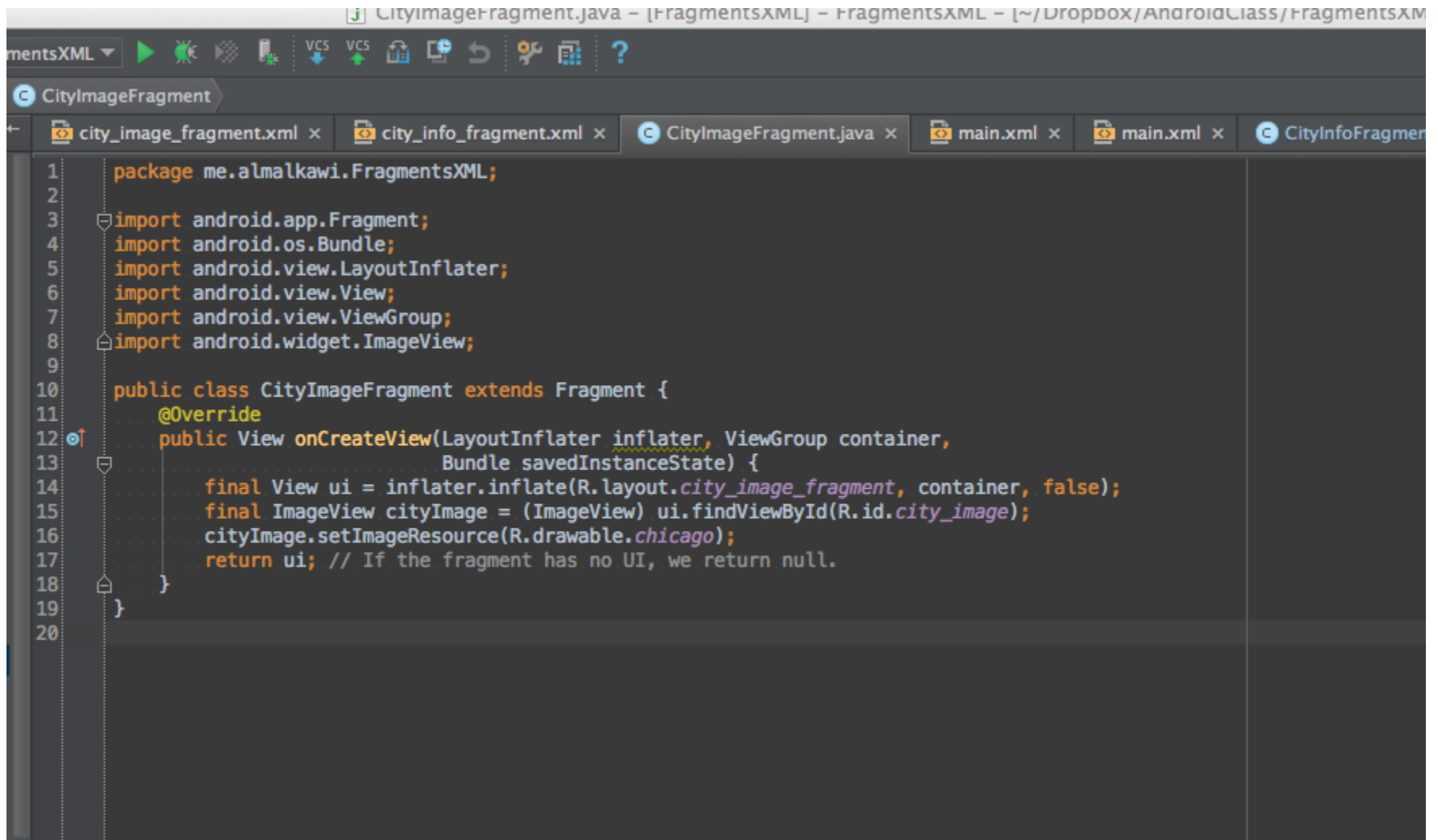


# CityImage Fragment UI (city\_image\_fragment.xml)



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     . . . . . android:layout_width="match_parent"
4     . . . . . android:layout_height="match_parent"
5     . . . . . android:orientation="horizontal">
6
7     <ImageView android:id="@+id/city_image"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"/>
10
11 </LinearLayout>
12
```

# CityImage Fragment (CityImageFragment.java)



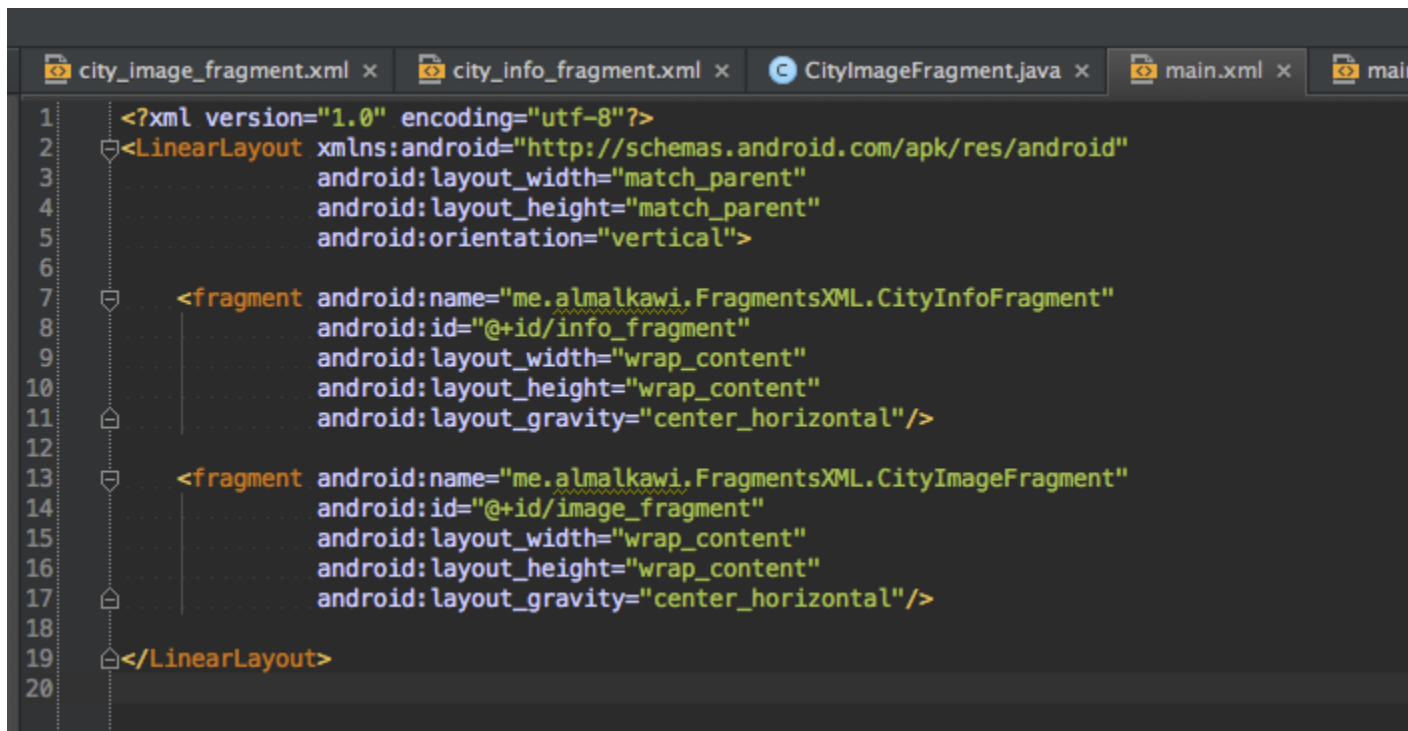
```
CityImageFragment.java - [FragmentsXML] - FragmentsXML - [~/Dropbox/AndroidClass/FragmentsXML]
CityImageFragment
city_image_fragment.xml x city_info_fragment.xml x CityImageFragment.java x main.xml x main.xml x CityInfoFragmen
1 package me.almalkawi.FragmentsXML;
2
3 import android.app.Fragment;
4 import android.os.Bundle;
5 import android.view.LayoutInflater;
6 import android.view.View;
7 import android.view.ViewGroup;
8 import android.widget.ImageView;
9
10 public class CityImageFragment extends Fragment {
11     @Override
12     public View onCreateView(LayoutInflater inflater, ViewGroup container,
13                             Bundle savedInstanceState) {
14         final View ui = inflater.inflate(R.layout.city_image_fragment, container, false);
15         final ImageView cityImage = (ImageView) ui.findViewById(R.id.city_image);
16         cityImage.setImageResource(R.drawable.chicago);
17         return ui; // If the fragment has no UI, we return null.
18     }
19 }
20
```

# MainActivity.java

```
MainActivity
city_image_fragment.xml × city_info_fragment.xml × CityImageFragment.java × main.xml ×

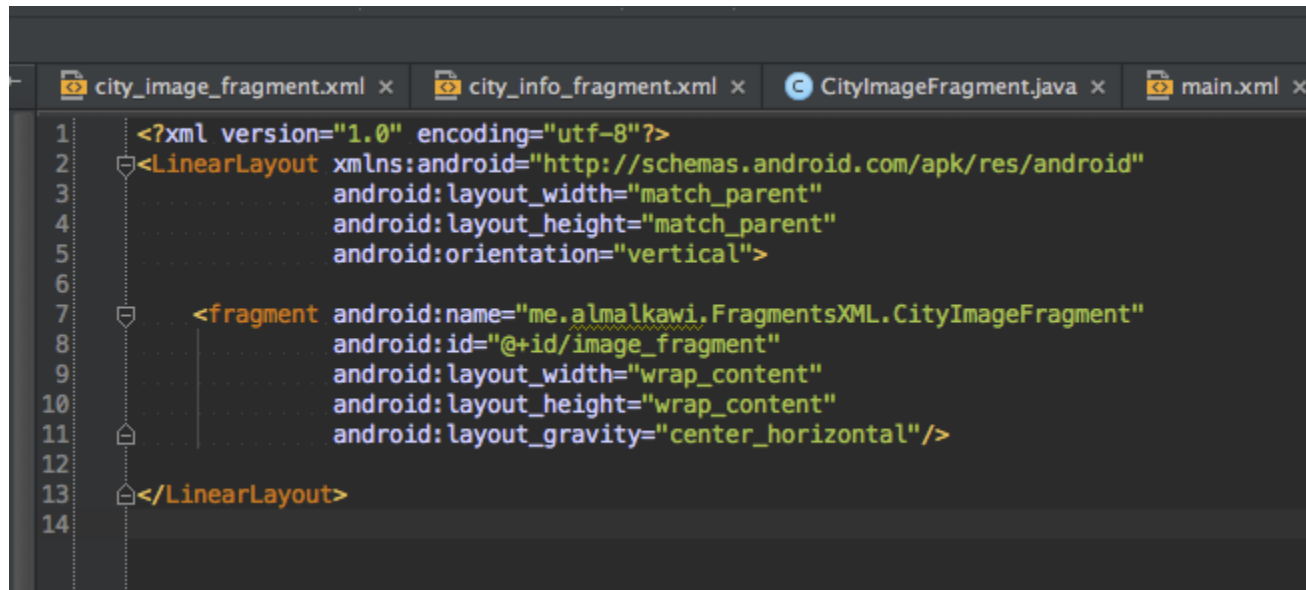
1 package me.almalkawi.FragmentsXML;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.widget.Toast;
6
7 public class MainActivity extends Activity {
8     @Override
9     public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         Toast.makeText(this, "Activity: onCreate", Toast.LENGTH_SHORT).show();
12         setContentView(R.layout.main);
13     }
14
15     @Override
16     protected void onStart() {
17         super.onStart();
18         Toast.makeText(this, "Activity: onStart", Toast.LENGTH_SHORT).show();
19     }
20
21     @Override
22     protected void onRestart() {
23         super.onRestart();
24         Toast.makeText(this, "Activity: onRestart", Toast.LENGTH_SHORT).show();
25     }
26
27     @Override
28     protected void onResume() {
29         super.onResume();
30         Toast.makeText(this, "Activity: onResume", Toast.LENGTH_SHORT).show();
31     }
32
33     @Override
34     protected void onPause() {
35         super.onPause();
36         Toast.makeText(this, "Activity: onPause", Toast.LENGTH_SHORT).show();
37     }
38
39     @Override
40     protected void onStop() {
41         super.onStop();
42         Toast.makeText(this, "Activity: onStop", Toast.LENGTH_SHORT).show();
43     }
44
45     @Override
46     protected void onDestroy() {
47         super.onDestroy();
48         Toast.makeText(this, "Activity: onDestroy", Toast.LENGTH_SHORT).show();
49     }
50 }
```

# res/layout/main.xml



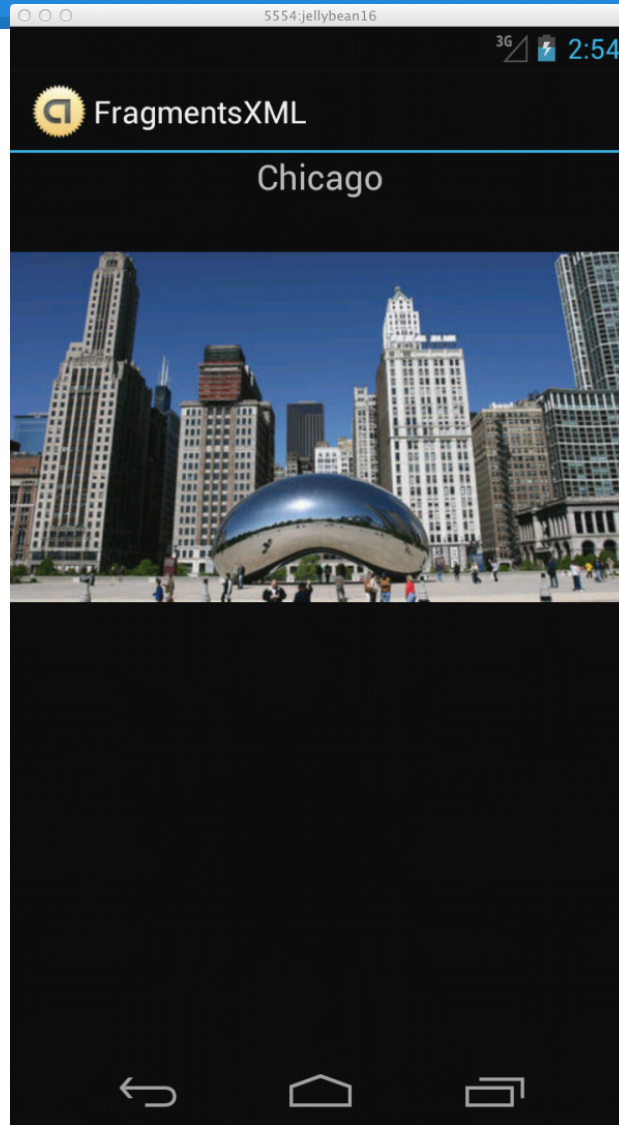
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6
7      <fragment android:name="me.almalkawi.FragmentsXML.CityInfoFragment"
8          android:id="@+id/info_fragment"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_gravity="center_horizontal"/>
12
13     <fragment android:name="me.almalkawi.FragmentsXML.CityImageFragment"
14         android:id="@+id/image_fragment"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:layout_gravity="center_horizontal"/>
18
19 </LinearLayout>
20
```

# res/layout-land/main.xml

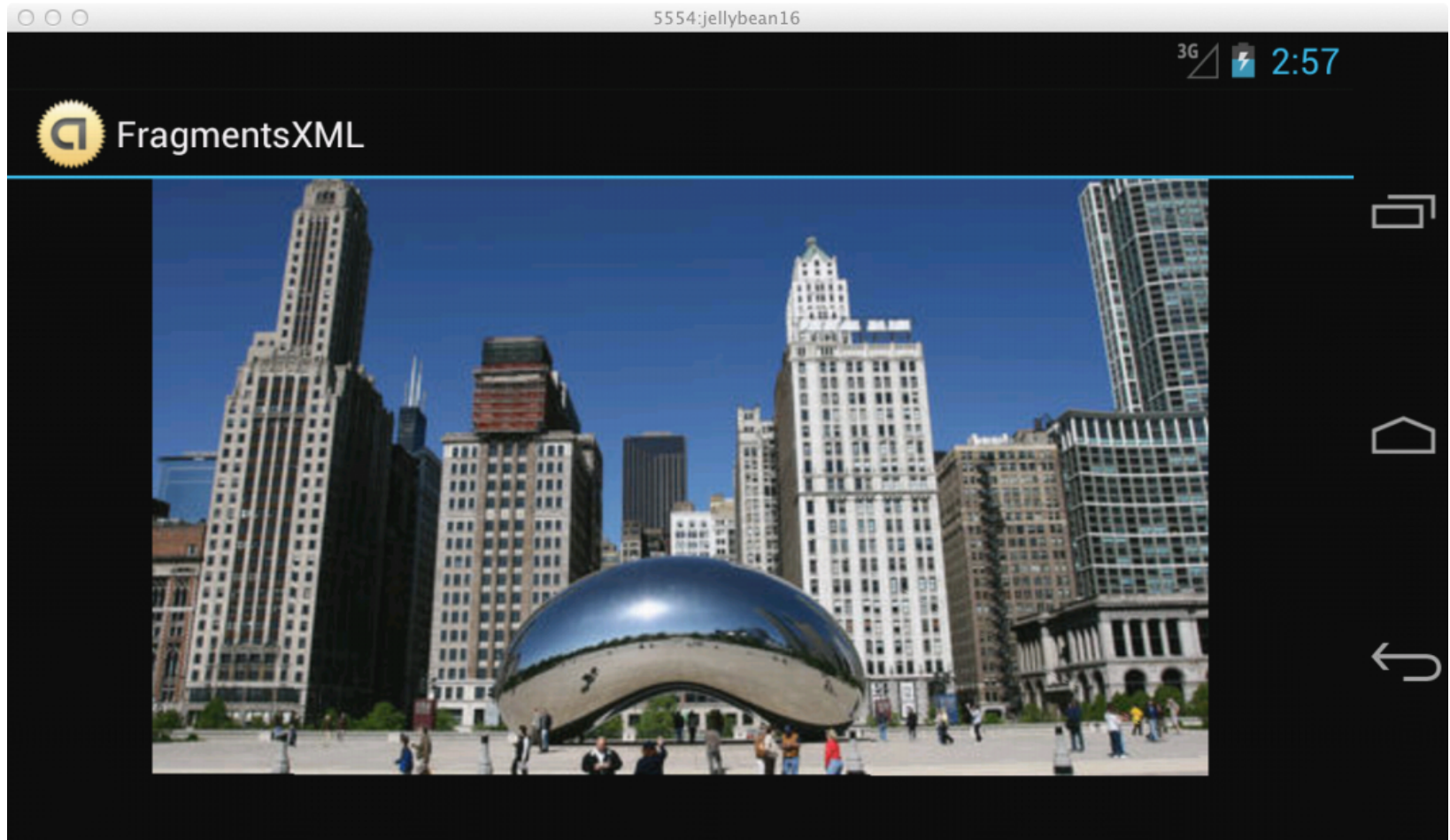


```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6
7      <fragment android:name="me.almalkawi.FragmentsXML.CityImageFragment"
8          android:id="@+id/image_fragment"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_gravity="center_horizontal"/>
12
13 </LinearLayout>
14
```

# Running project (Portrait)



# Running project (Landscape)



# Demo: Adding/Removing Fragments at runtime

**Source:** <https://github.com/almalkawi/AndroidClass/tree/master/DynamicFragments>

These are the same as in previous demo:

CityInfoFragment.java

city\_image\_fragment.xml

CityImageFragment.java

city\_image\_fragment.xml

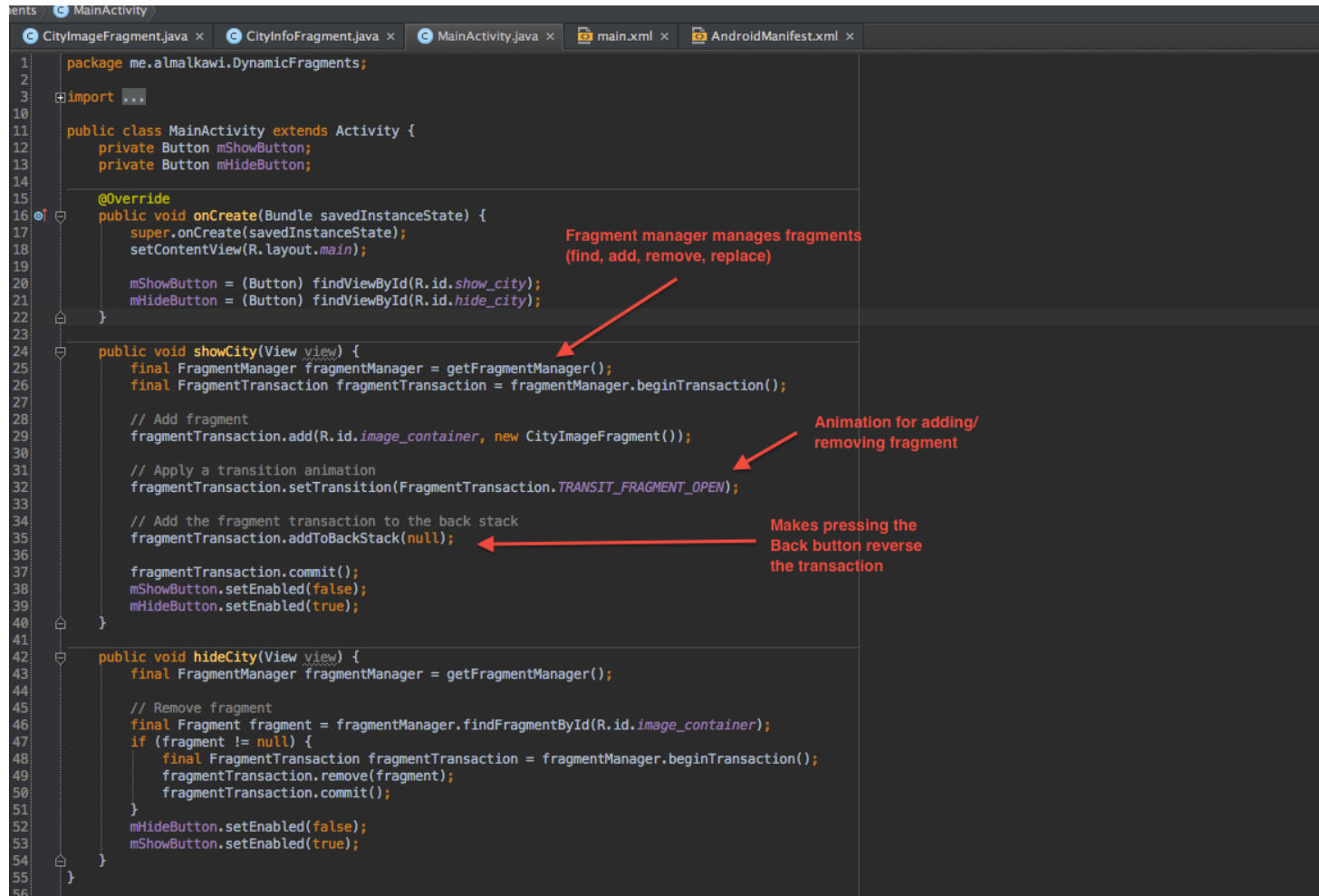


# main.xml

```
CityImageFragment.java x CityInfoFragment.java x MainActivity.java x main.xml x AndroidManifest.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical">
6
7     <fragment android:name="me.almalkawi.DynamicFragments.CityInfoFragment"
8         android:id="@+id/image_fragment"
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:layout_gravity="center_horizontal"/>
12
13    <LinearLayout android:layout_width="wrap_content"
14        android:layout_height="wrap_content"
15        android:orientation="horizontal"
16        android:layout_gravity="center_horizontal">
17
18        <Button android:id="@+id/show_city"
19            android:layout_width="wrap_content"
20            android:layout_height="wrap_content"
21            android:layout_gravity="center_horizontal"
22            android:text="Add fragment"
23            android:enabled="true"
24            android:onClick="showCity"/>
25
26        <Button android:id="@+id/hide_city"
27            android:layout_width="wrap_content"
28            android:layout_height="wrap_content"
29            android:layout_gravity="center_horizontal"
30            android:text="Del fragment"
31            android:enabled="false"
32            android:onClick="hideCity"/>
33
34    </LinearLayout>
35
36    <FrameLayout android:id="@+id/image_container"
37        android:layout_width="wrap_content"
38        android:layout_height="wrap_content"
39        android:layout_gravity="center_horizontal"/>
40
41 </LinearLayout>
42
```

Container in which fragment can be placed at runtime

# MainActivity.java



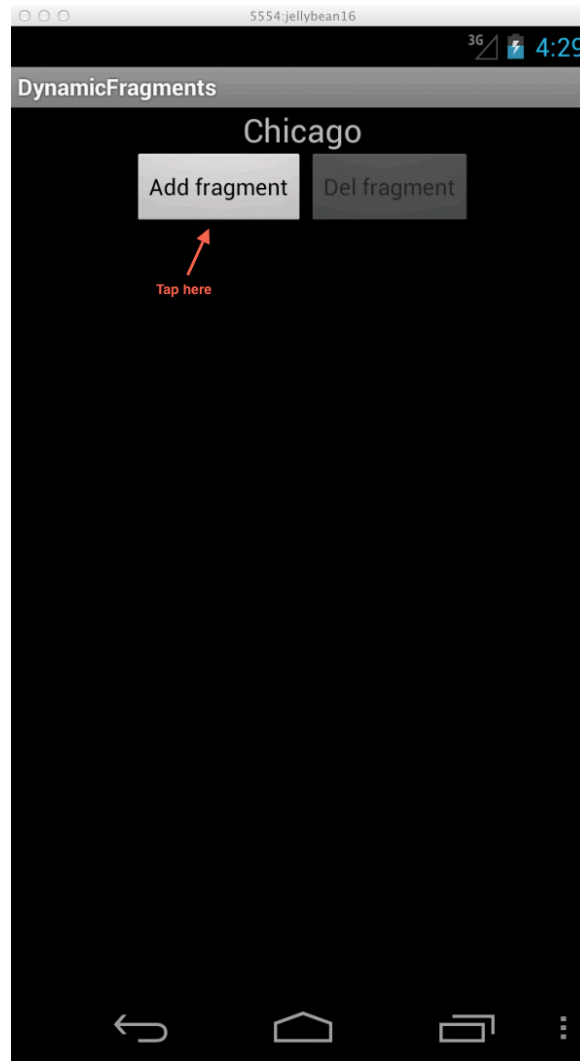
```
1 package me.almalkawi.DynamicFragments;
2
3 import ...
4
5
6
7
8
9
10
11 public class MainActivity extends Activity {
12     private Button mShowButton;
13     private Button mHideButton;
14
15     @Override
16     public void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.main);
19
20         mShowButton = (Button) findViewById(R.id.show_city);
21         mHideButton = (Button) findViewById(R.id.hide_city);
22     }
23
24     public void showCity(View view) {
25         final FragmentManager fragmentManager = getFragmentManager();
26         final FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
27
28         // Add fragment
29         fragmentTransaction.add(R.id.image_container, new CityImageFragment());
30
31         // Apply a transition animation
32         fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
33
34         // Add the fragment transaction to the back stack
35         fragmentTransaction.addToBackStack(null);
36
37         fragmentTransaction.commit();
38         mShowButton.setEnabled(false);
39         mHideButton.setEnabled(true);
40     }
41
42     public void hideCity(View view) {
43         final FragmentManager fragmentManager = getFragmentManager();
44
45         // Remove fragment
46         final Fragment fragment = fragmentManager.findFragmentById(R.id.image_container);
47         if (fragment != null) {
48             final FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
49             fragmentTransaction.remove(fragment);
50             fragmentTransaction.commit();
51         }
52         mHideButton.setEnabled(false);
53         mShowButton.setEnabled(true);
54     }
55 }
56
```

Fragment manager manages fragments (find, add, remove, replace)

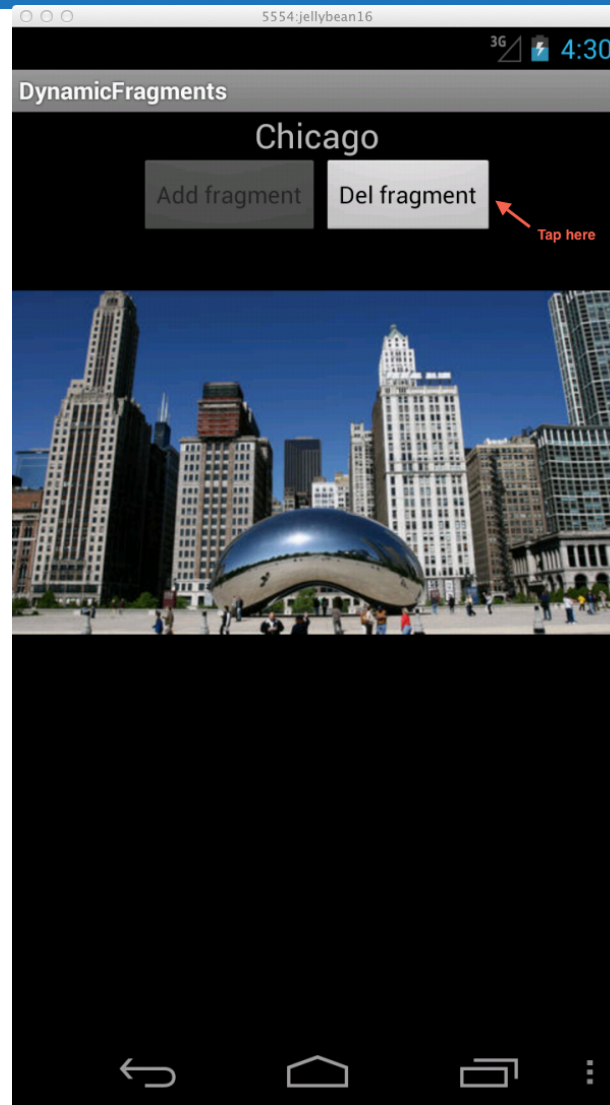
Animation for adding/removing fragment

Makes pressing the Back button reverse the transaction

# Running project (initial and after removing fragment)



# Running project (after tapping "Add fragment")



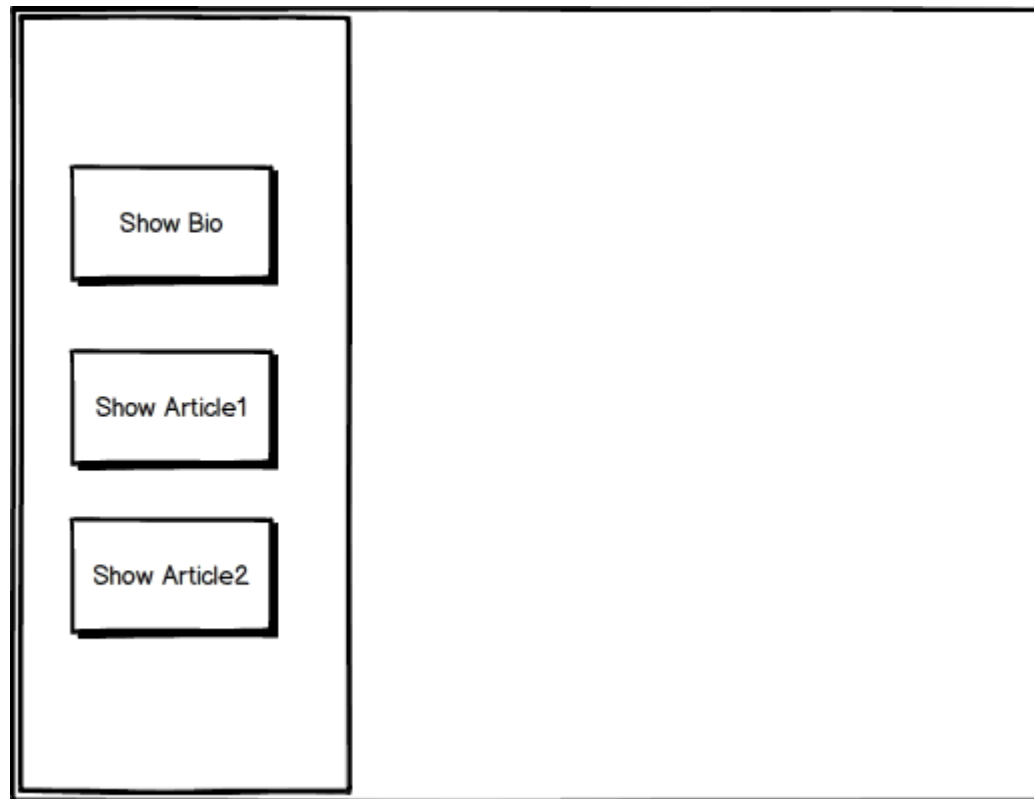
# Hello Android!

Fragments Exercise

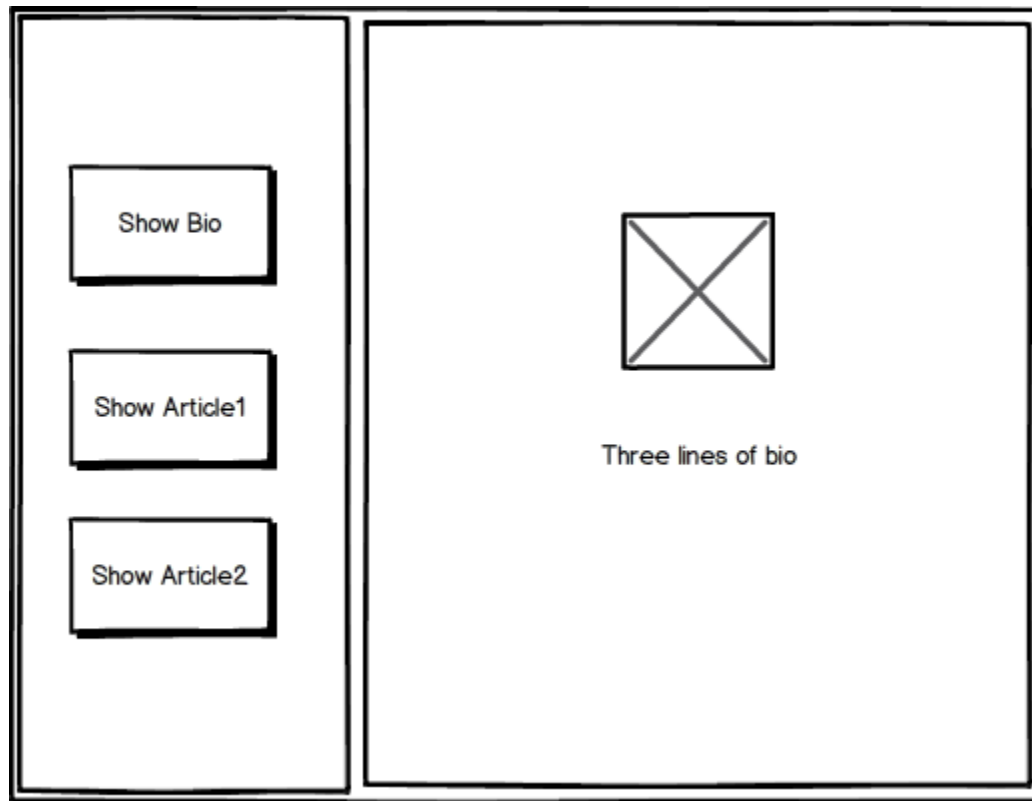
# Exercise: Reader App

- Use one static Fragment for left pane
- Use one dynamic Fragment for bio
- Use one dynamic Fragment for article
- Mockups are in next slides

# Exercise: Initial State

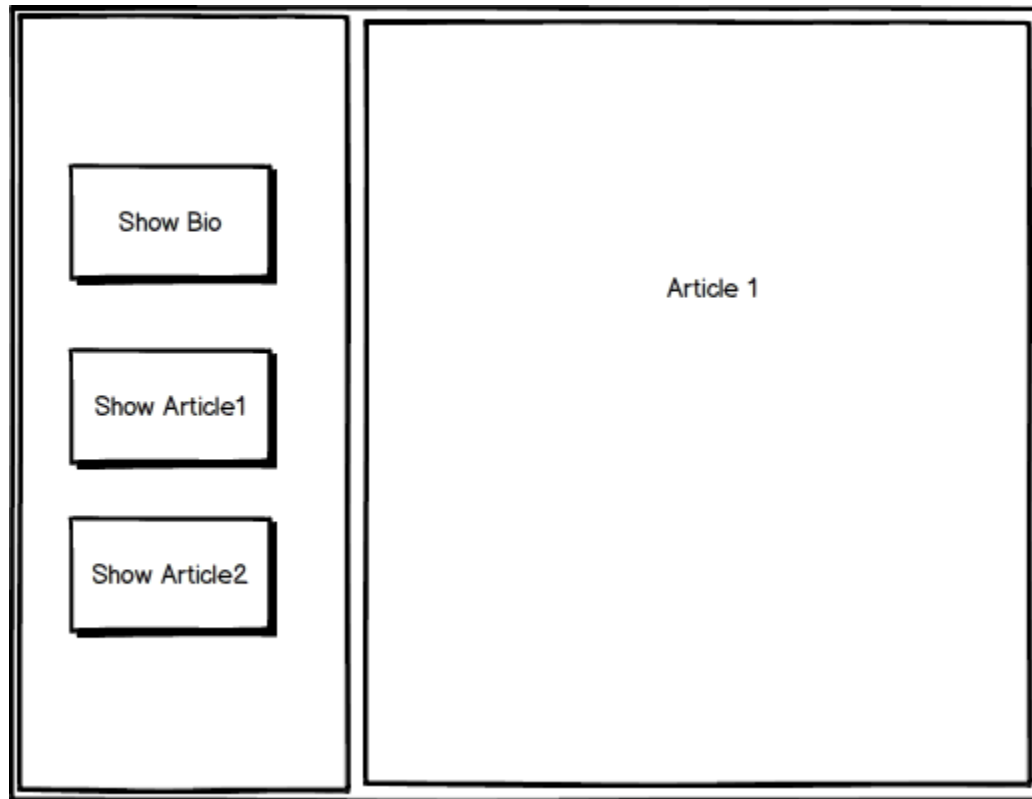


# Exercise: Showing Bio Fragment





# Exercise: Showing Article1



# Exercise: Showing Article2

