Android Application Development, COMP 10073

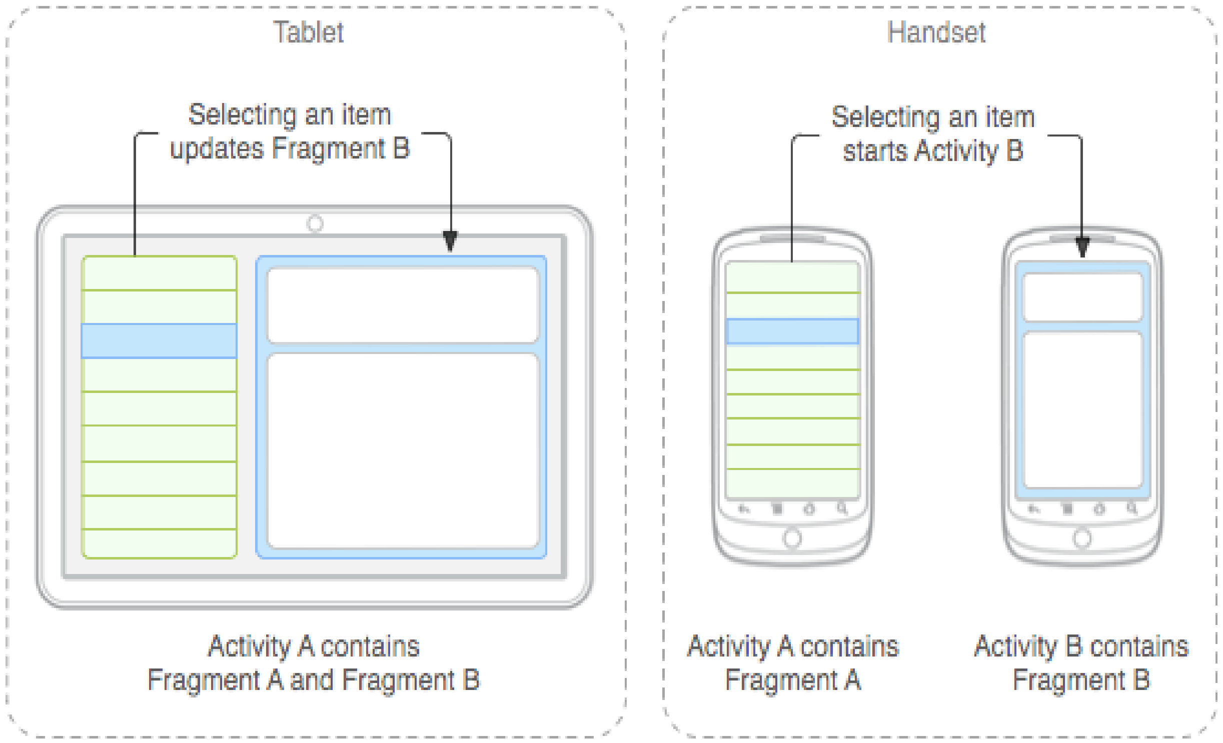
Mohawk College, Winter 2021

Fragments

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

Not as common on small screens. Very common on Tablets. Think of it as a sub-view, a portion of a screen.

Activities are an ideal place to put global elements around your app's user interface, such as a navigation drawer. Conversely, fragments are better suited to define and manage the UI of a single screen or portion of a screen.

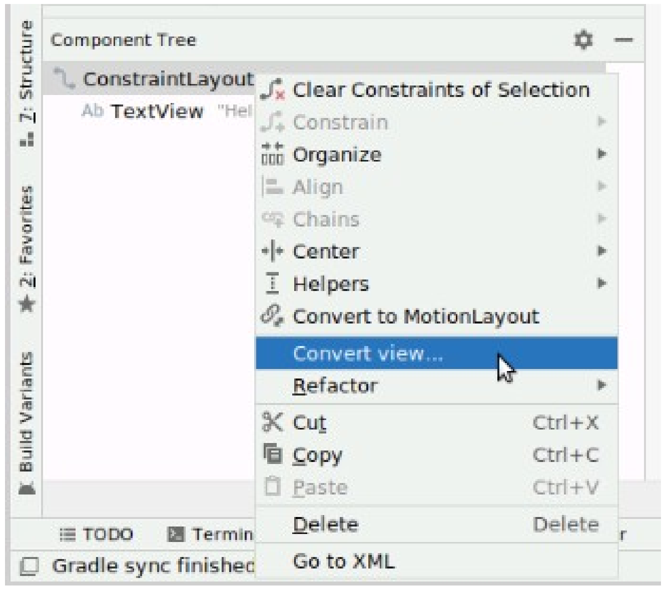

Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

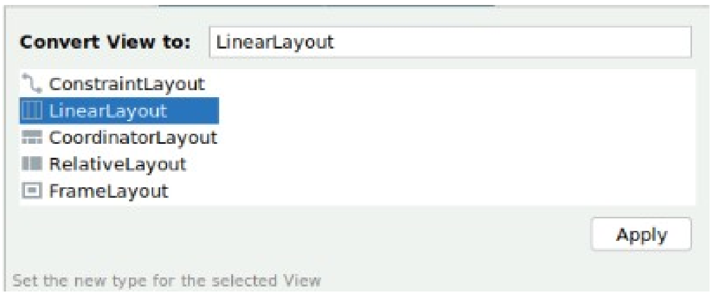
The same code can be designed to work on two devices which have different capacities. Fragments promote modular organization and reuse. Fragments are always in part of an activity. Can have several fragments in one activity. Fragments have their own lifecycle and UI event callbacks, they are affected by the parent activity lifecycle, for example if the activity is destroyed all the fragments in it are also destroyed.

Linear Layout

https://developer.android.com/guide/topics/ui/layout/linear

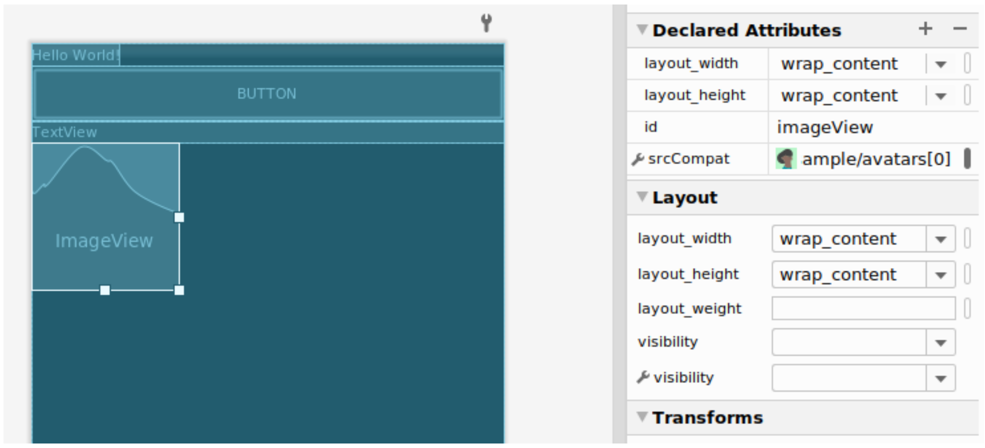
For this example we will experiment with a Linear weighted Layout. Start by creating a new project with a blank activity. Go to the layout editor, open the Component Tree, and right click on the top level ConstraintLayout and select Convert view... in the submenu select LinearLayout.





By default the linear layout will be horizontal, convert it to a vertical layout by a similar operation, i.e. right click on LinearLayout in the component tree, from the first item select convert orientation to vertical.

Previously we said that the linear layout was less flexible than the constrained layout, but it still has some nice features. You should notice that elements are positioned one after the other without any margins.



Declared Attributes	
layout_width	wrap_content
layout_height	wrap_content
id	imageView
srcCompat	ample/avatars[0]

Layout	
layout_width	wrap_content
layout_height	wrap_content
layout_weight	
visibility	
visibility	

Using Weights with the Linear Layout

LinearLayout supports assigning a weight to individual children with the android:layout_weight attribute. This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen. A larger weight value allows it to expand to fill any remaining space in the parent view.

To create a linear layout in which each child uses the same amount of vertical space on the screen, set the android:layout_height of each view to "0dp". Then set the android:layout_weight of each view to "1".

You can create unequal distributions by using different relative weights, i.e. if you set the weight of one element to 2, and the other element to 1, the smaller element will take up 1/2 as much space on the screen.

Fragments live in Frames

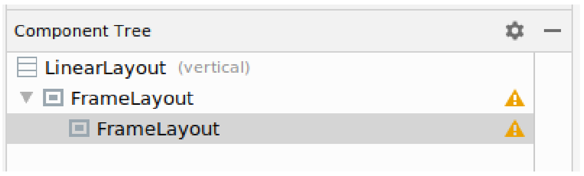
https://developer.android.com/reference/android/widget/FrameLayout

The fragment is a general abstraction like an activity, except that multiple fragments can live inside an activity. When we create an activity it takes over the whole user interface so its location is implied.

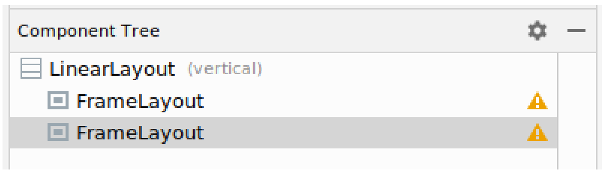
The fragment can use a portion of the activity, and so we have to define exactly where the fragment will be drawn. We can do that via XML or programmatically.

Starting with an empty activity, add two FrameLayouts. The FrameLayout is designed to block out an area on the screen to display a single item. FrameLayouts are used to hold a single child view and make it easier to organize that child's view within the Frame. Make sure that the 2nd FrameLayout is not a child of the first, but is rather a child of the LinearLayout.

You can reposition these elements in the component tree if necessary



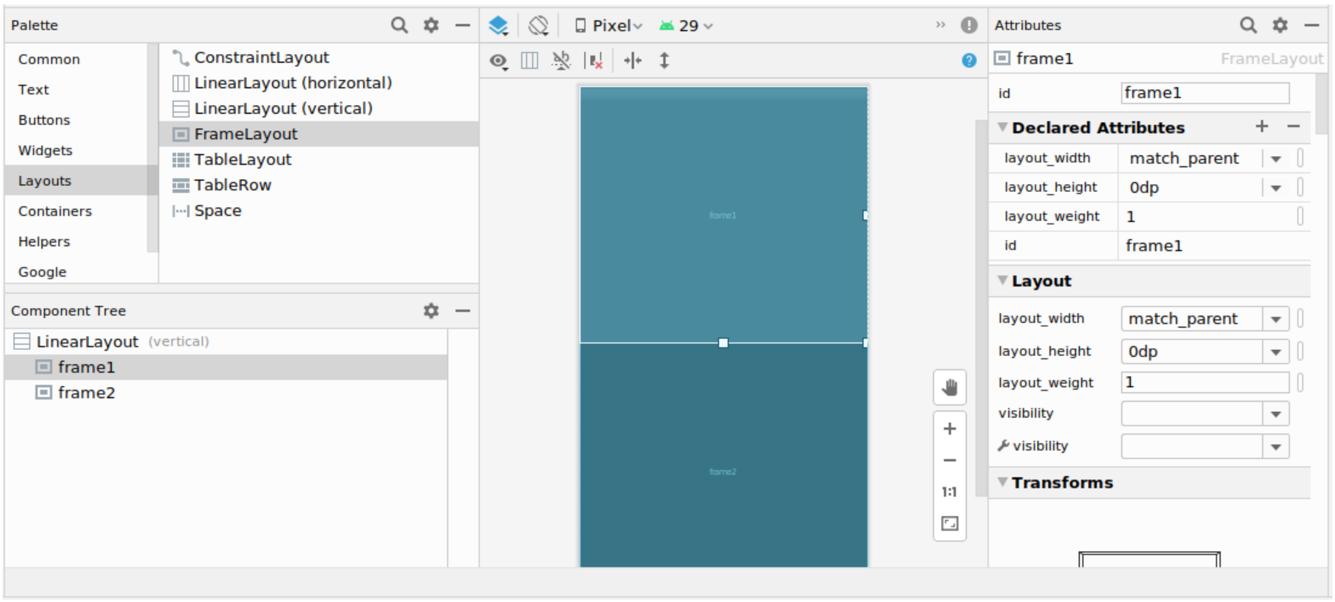
==>



Set the height of both FrameLayouts to 0dp, and set the weight of each to 1.

Give them different ids, frame1 and frame2, for example.

Two Balanced FrameLayouts



frame1 FrameLayout
id frame1

Declared Attributes

layout_width	match_parent
layout_height	0dp
layout_weight	1
id	frame1

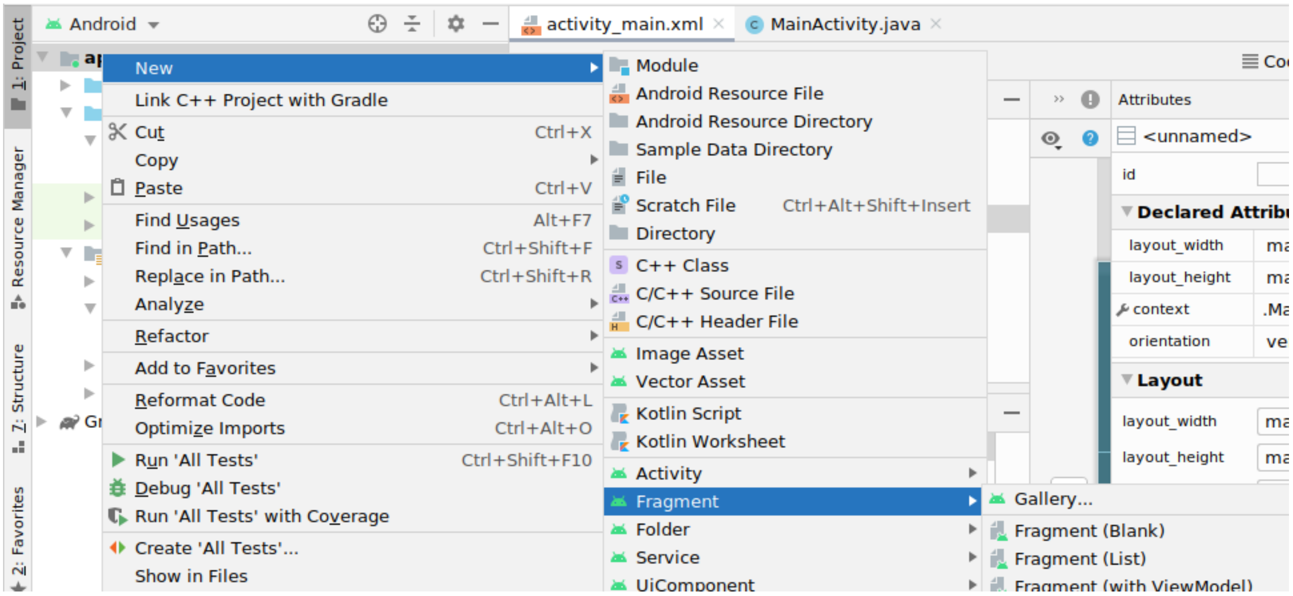
Layout

layout_width	match_parent
layout_height	0dp
layout_weight	1
visibility	
visibility	

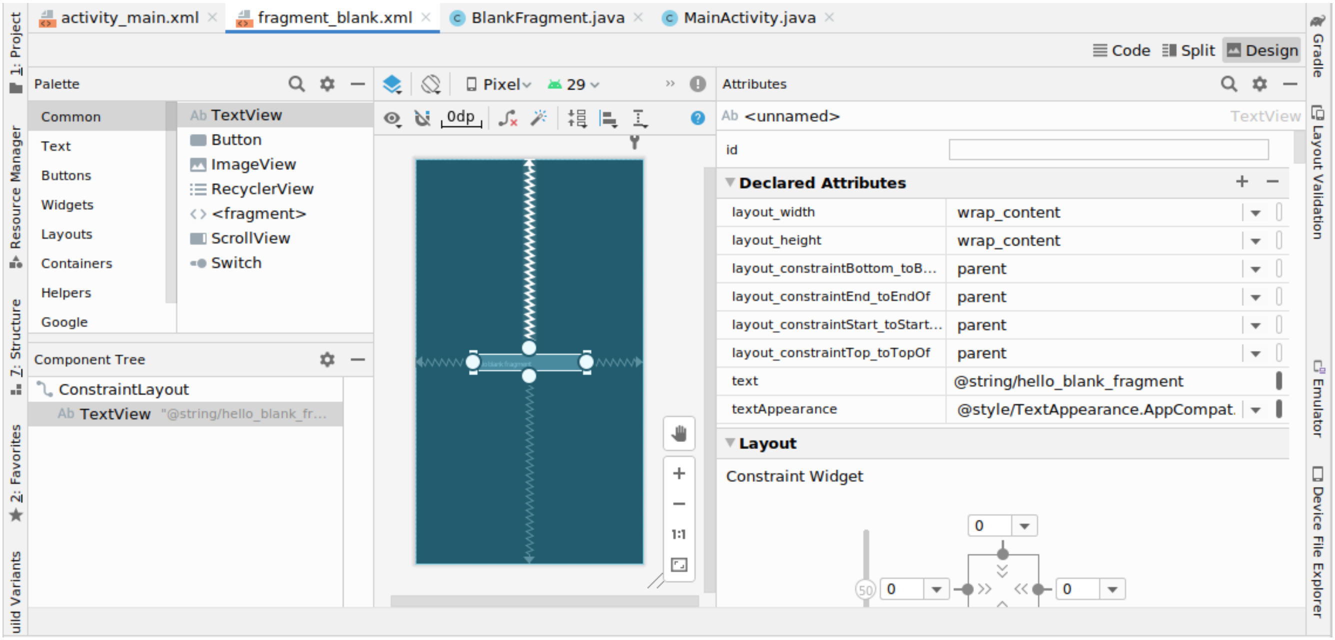
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/frame1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
       android:background="@color/purple_200">

    </FrameLayout>

    <FrameLayout
        android:id="@+id/frame2"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">

    </FrameLayout>

</LinearLayout>
```

Create A Fragment

So far we have been focused on Layout. The next step is to create a fragment, this is similar to creating a new activity. Right click on the project file tree, select new fragment, and “Blank” fragment. This will create a new layout file and it will also create a Java file with some fundamental fragment methods.



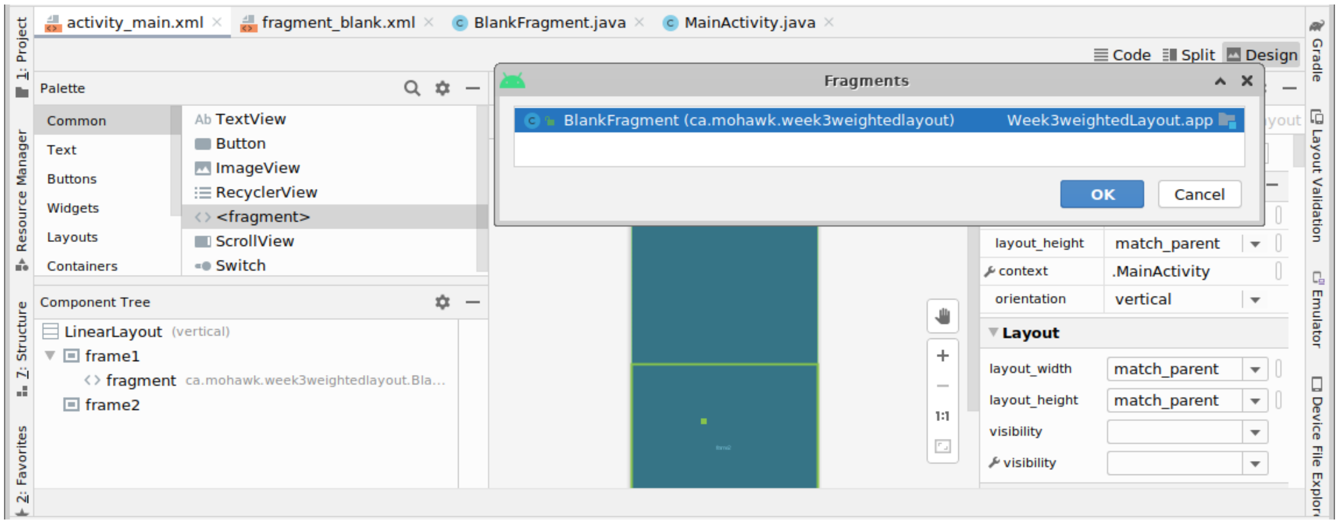
Add a TextView Widget to the fragment layout file and center the text.



Attribute	Value
id	
layout_width	wrap_content
layout_height	wrap_content
layout_constraintBottom_toB...	parent
layout_constraintEnd_toEndOf	parent
layout_constraintStart_toStart...	parent
layout_constraintTop_toTopOf	parent
text	@string/hello_blank_fragment
textAppearance	@style/TextAppearance.AppCompat.

Add the Fragment to the Frame

You will find a fragment item under the Common Palette. Take note that if you try to add this element without having first created the fragment you will get an error. The BlankFragment that is being added here was created in the previous steps.



Fragments

BlankFragment (ca.mohawk.week3weightedlayout) Week3weightedLayout.app

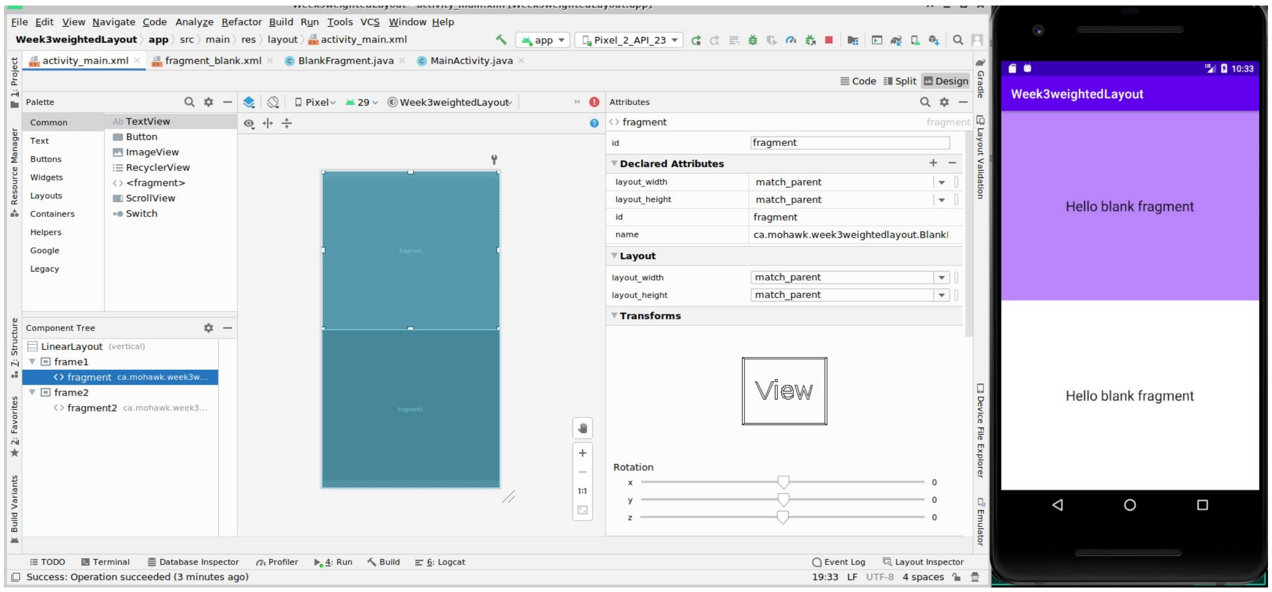
LinearLayout (vertical)
frame1
fragment ca.mohawk.week3weightedlayout.Bla...
frame2

layout_height	match_parent
context	.MainActivity
orientation	vertical

Layout

layout_width	match_parent
layout_height	match_parent
visibility	
visibility	

Set the fragment width and height to match its parent so it fills the frame. The color in the background of the top frame makes the layout more obvious.



Component Tree
LinearLayout (vertical)
frame1
fragment ca.mohawk.week3w...
frame2
fragment2 ca.mohawk.week3...

fragment

id	fragment

Declared Attributes

layout_width	match_parent
layout_height	match_parent
id	fragment
name	ca.mohawk.week3weightedlayout.BlankI

Layout

layout_width	match_parent
layout_height	match_parent

Hello blank fragment

Hello blank fragment

Success: Operation succeeded (3 minutes ago)

19:33 LF UTF-8 4 spaces

This example illustrates layout reuse. We have defined a single layout for “hello blank fragment”, and drawn that layout in multiple positions.

Do we need to put fragments inside frames? No, but fragments are most useful as dynamic entities, and to use them that way they need an anchor point.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/frame1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/purple_200">

        <fragment
            android:id="@+id/fragment"
            android:name="ca.mohawk.week3weightedlayout.BlankFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </FrameLayout>

    <FrameLayout
        android:id="@+id/frame2"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">

        <fragment
            android:id="@+id/fragment2"
            android:name="ca.mohawk.week3weightedlayout.BlankFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </FrameLayout>

</LinearLayout>
```

fragment_blank.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".BlankFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_blank_fragment"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Fragment LifeCycle

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

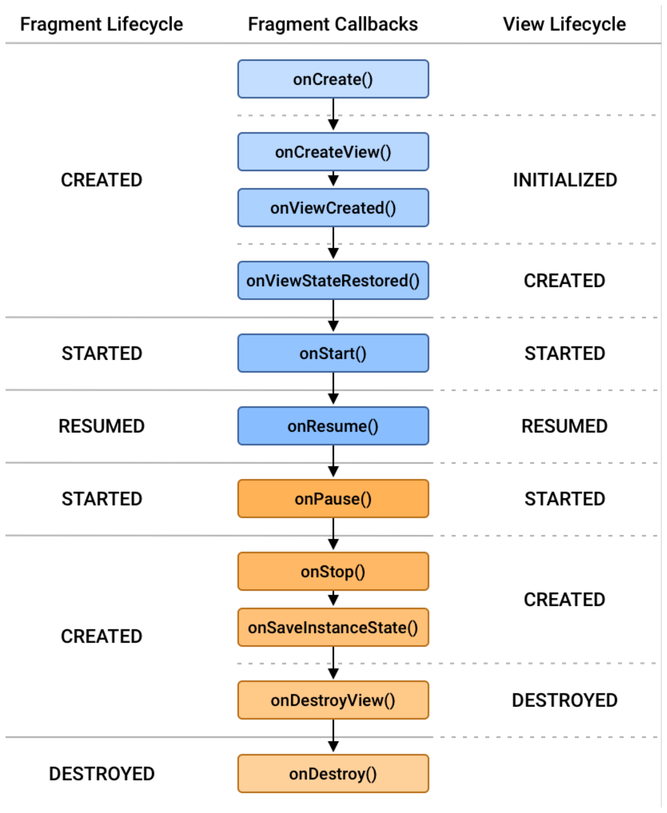
The blank fragment that we created comes with some template code. Fragments have a lifecycle similar to the activity lifecycle.

```java
/**
 * A simple {@link Fragment} subclass.
 * Use the {@link BlankFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class BlankFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public BlankFragment() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment BlankFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static BlankFragment newInstance(String param1, String param2) {
        BlankFragment fragment = new BlankFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

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

Fragment LifeCycle

Simplify the code by removing the two argument factory constructor. Add some logging facilities and a static counter. We will get back to the two argument factory constructor later.

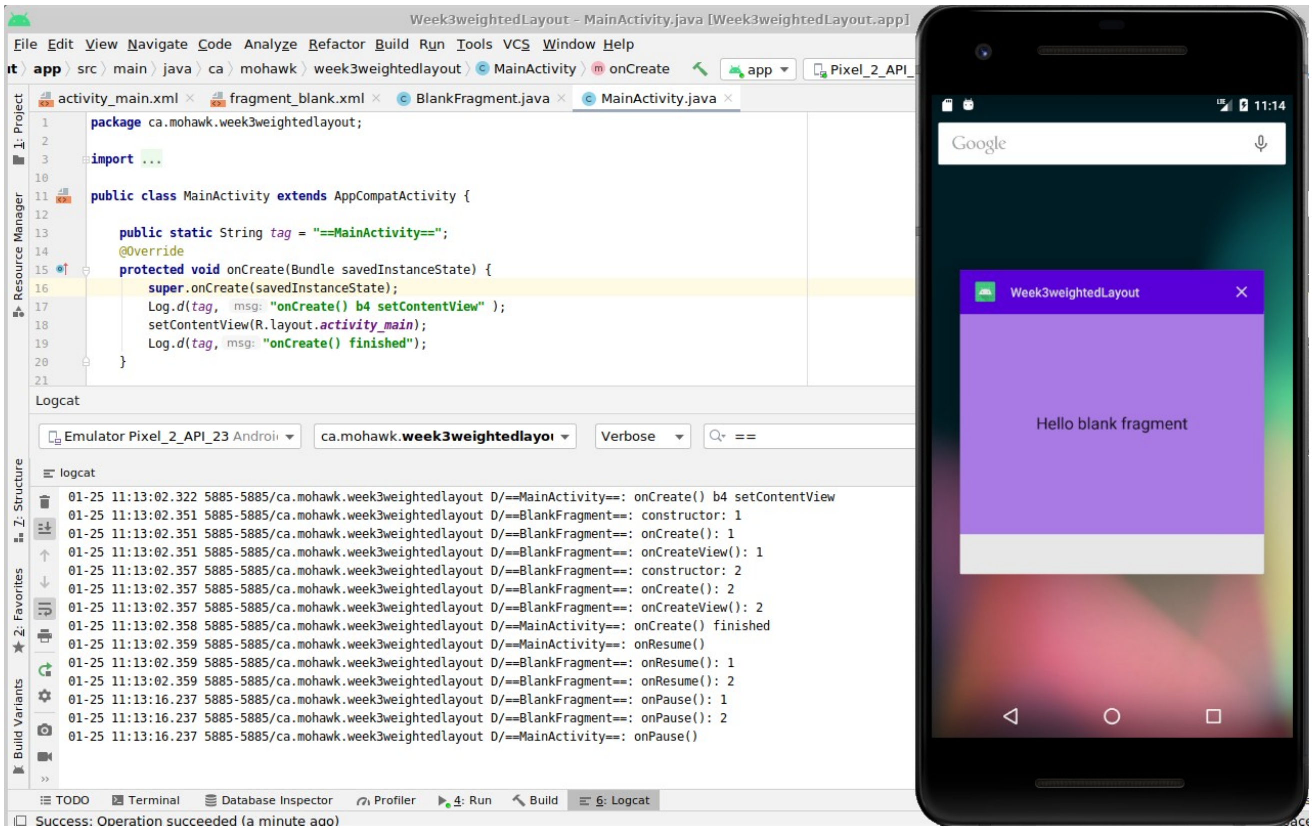
```java
public class BlankFragment extends Fragment {

    public static String tag = "==BlankFragment==";

    private static int fragCount = 0;
    private int fragID;
    public BlankFragment() {
        // Required empty public constructor
        fragCount += 1;
        fragID = fragCount;
        Log.d(tag, "constructor: "+ fragID);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(tag, "onCreate(): " + fragID);
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
             ViewGroup container,
             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        Log.d(tag, "onCreateView(): " + fragID);
        return inflater.inflate(R.layout.fragment_blank,
                container, false);
    }

    // called immediately before fragment becomes active
    @Override
    public void onResume() {
        super.onResume();
        Log.d(tag,"onResume(): " + fragID);
    }
    // called when fragment or parent activity loses focus
    @Override
    public void onPause() {
        super.onPause();
        Log.d(tag,"onPause(): " + fragID);
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(tag,"onDestroy(): " + fragID);
    }
}
```



Fragment LifeCycle

Also add some logging details to the main activity, in particular lets track what happens when setContentView() is called.

```java
public class MainActivity extends AppCompatActivity {

    public static String tag = "==MainActivity==";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(tag, "onCreate() b4 setContentView" );
        setContentView(R.layout.activity_main);
        Log.d(tag,"onCreate() finished");
    }

    // called immediately before activity becomes active
    @Override
    public void onResume() {
        super.onResume();
        Log.d(tag,"onResume()");
    }
    // called when activity loses focus
    @Override
    public void onPause() {
        super.onPause();
        Log.d(tag,"onPause()");
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(tag,"onDestroy()");
    }
}
```



```java
package ca.mohawk.week3weightedlayout;

import ...

public class MainActivity extends AppCompatActivity {

    public static String tag = "==MainActivity==";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(tag, msg: "onCreate() b4 setContentView" );
        setContentView(R.layout.activity_main);
        Log.d(tag, msg: "onCreate() finished");
    }
```

```
01-25 11:13:02.322 5885-5885/ca.mohawk.week3weightedlayout D/==MainActivity==: onCreate() b4 setContentView
01-25 11:13:02.351 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: constructor: 1
01-25 11:13:02.351 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onCreate(): 1
01-25 11:13:02.351 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onCreateView(): 1
01-25 11:13:02.357 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: constructor: 2
01-25 11:13:02.357 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onCreate(): 2
01-25 11:13:02.357 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onCreateView(): 2
01-25 11:13:02.358 5885-5885/ca.mohawk.week3weightedlayout D/==MainActivity==: onCreate() finished
01-25 11:13:02.359 5885-5885/ca.mohawk.week3weightedlayout D/==MainActivity==: onResume()
01-25 11:13:02.359 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onResume(): 1
01-25 11:13:02.359 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onResume(): 2
01-25 11:13:16.237 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onPause(): 1
01-25 11:13:16.237 5885-5885/ca.mohawk.week3weightedlayout D/==BlankFragment==: onPause(): 2
01-25 11:13:16.237 5885-5885/ca.mohawk.week3weightedlayout D/==MainActivity==: onPause()
```

Week3weightedLayout - MainActivity.java [Week3weightedLayout.app]
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app src main java ca mohawk week3weightedlayout MainActivity onCreate
activity_main.xml fragment_blank.xml BlankFragment.java MainActivity.java
Logcat
Emulator Pixel_2_API_23 Android
ca.mohawk.week3weightedlayout
Verbose
==
TODO Terminal Database Inspector Profiler 4: Run Build 6: Logcat
Success: Operation succeeded (a minute ago)
11:14
Google
Week3weightedLayout
Hello blank fragment