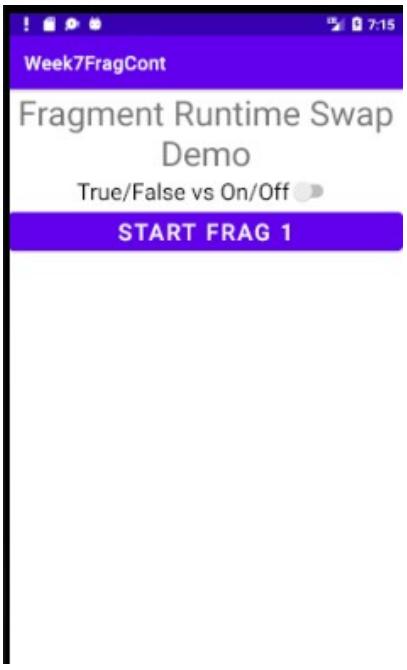
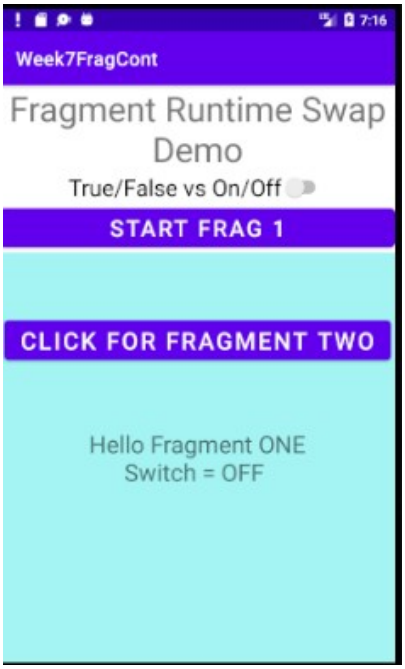
Android Application Development, COMP 10073

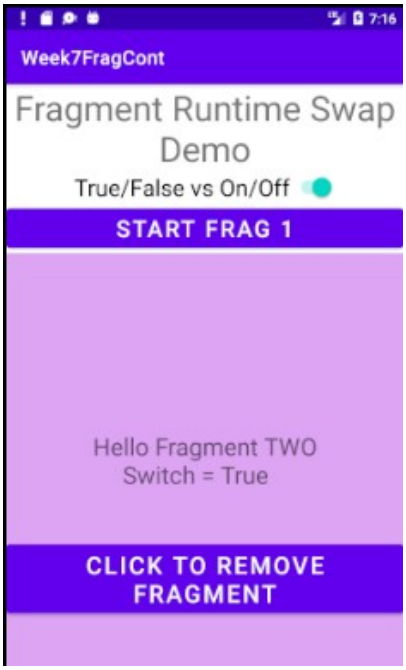
Mohawk College, Winter 2021

Fragment Manager Continued

For this weeks demo we will reuse much of what we have seen before. The advantage of fragments is the reusability of code that they allow. The Fragment Manager can be used to enable the dynamic swapping of layouts within layouts. This weeks example will use a single Activity and two Fragments. The Activity hosts one frame at the bottom and in response to button presses loads an activity. That activity can itself load the next activity which can close itself.







In our previous examples we talked about how fragments communicate with each other, and talked about some of the problems that we can run into with customizing the constructor, for example.

Each fragment in this example needs to be able to access the main activity. Both read the switch state. The first fragment displays either On or Off depending on the state of the switch when it started. The second fragment displays either true or false when the fragment is started.

In addition to having different behaviours the fragments use different background colors and different layouts. We will also introduce the use of gravity in this demonstration for adjusting layouts.

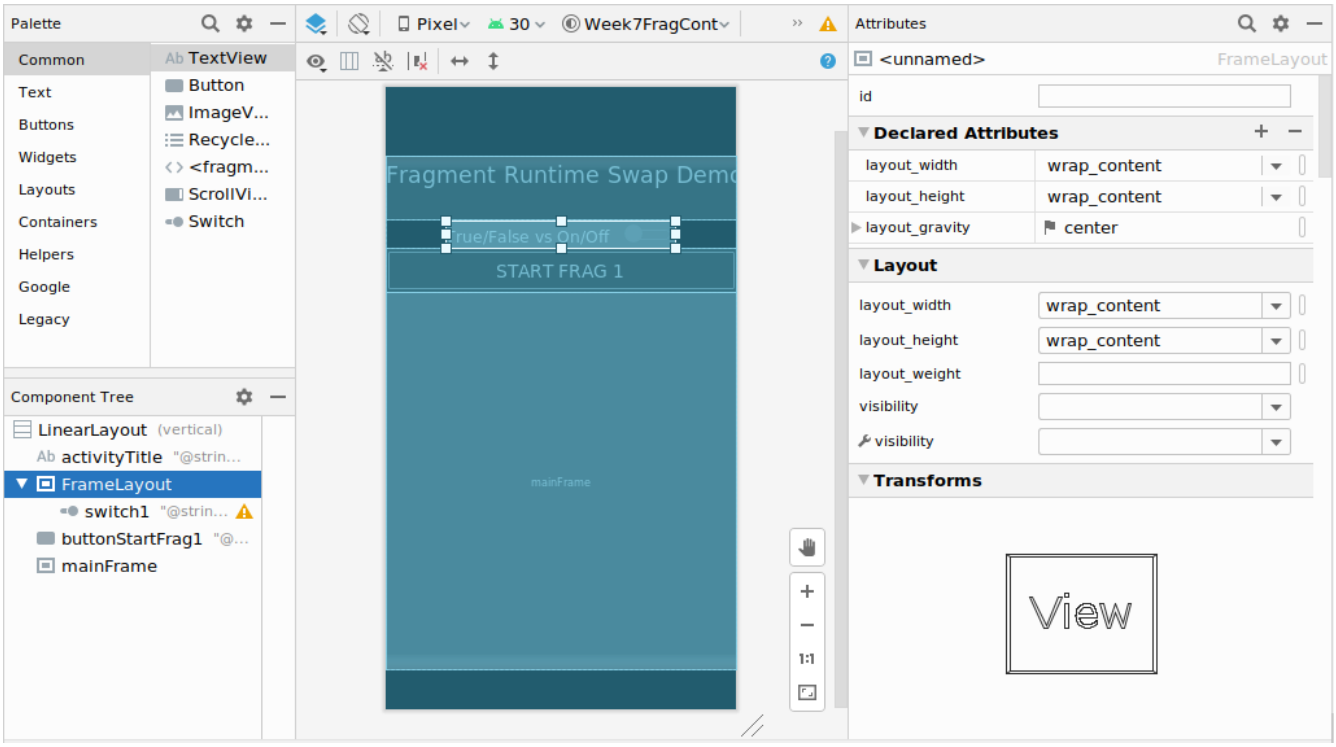
It is good practice to define your strings in the strings resource. The following strings are used in this example:

```
<resources>
    <string name="app_name">Week7FragCont</string>
    <string name="titleFrag">Fragment Runtime Swap Demo</string>
    <string name="bStart">Start Frag 1</string>
    <string name="switchLabel">True/False vs On/Off</string>
    <string name="hello1">Hello Fragment ONE</string>
    <string name="hello2">Hello Fragment TWO</string>
    <string name="nextFrag2">Click for Fragment TWO</string>
    <string name="remFrag">Click to remove fragment</string>
</resources>
```

Define a few custom colors for this example, but them in the colors.xml file.

```
<color name="frag1">#6000E0E0</color>
<color name="frag2">#60A200E0</color>
```

For the MainActivity use a linear layout. Include a TextView for the title, followed by a Frame to host the switch. Set the frame gravity to center to orient the switch within the frame. Following this create a button to start the first fragment, and underneath these widgets include the main frame.



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">

    <TextView
        android:id="@+id/activityTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/titleFrag"
        android:textSize="30sp" />

    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <Switch
            android:id="@+id/switch1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/switchLabel"
            android:textSize="24sp" />
    </FrameLayout>

    <Button
        android:id="@+id/buttonStartFrag1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="@string/bStart"
        android:textSize="24sp" />

    <FrameLayout
        android:id="@+id/mainFrame"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/activityTitle">

    </FrameLayout>


</LinearLayout>
```

fragment1.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/fragment1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/frag1">

        <TextView
            android:id="@+id/helloText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/hello1"
            android:textSize="25dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <Button
            android:id="@+id/buttonNextFrag2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/nextFrag2"
            android:textSize="25dp"
            app:layout_constraintBottom_toTopOf="@+id/helloText"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment2.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/fragment2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/frag2">

    <TextView
        android:id="@+id/helloText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello2"
        android:textSize="25dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/buttonRemFrag"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/remFrag"
        android:textSize="25dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/helloText" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

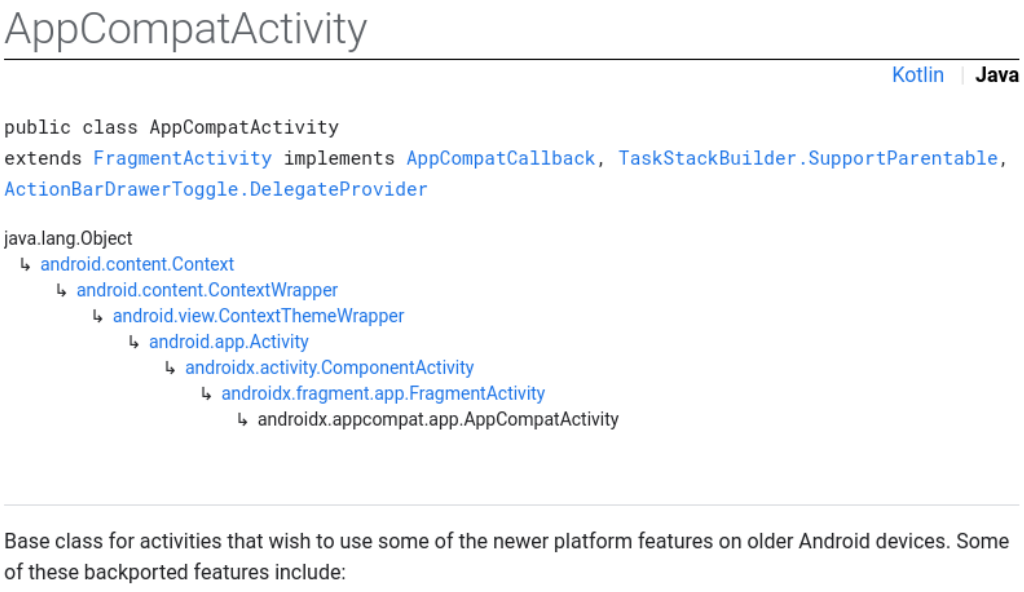
We mentioned previously that fragments promote reusability. They will do this best if we keep the code for fragments and activities separate as much as possible. Fragments should not hold on to references to each other, and since we are creating a destroying fragments within this activity we also need to be careful to not try to store state within the fragment itself.

Fragments can access the state of the hosting activity without storing any references to the main activity. Our MainActivity code for this example is quite simple. To start just adding some loggnig facilities to onCreate(), like so:

```
public class MainActivity extends AppCompatActivity {

    public static String TAG = "==MainActivity==";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "onCreate");
    }

}
```

Previously we introduced the Fragment Manager. We accessed it from the Activities context. By default Android Studio 4.1.x gives us an Activity of type AppCompatActivity. Notice the inheritance hierarchy:

https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity



getSupportFragmentManager()

https://developer.android.com/reference/androidx/fragment/app/FragmentActivity

https://developer.android.com/reference/androidx/fragment/app/FragmentManager

https://developer.android.com/reference/android/app/FragmentTransaction.html

In our main activity we need a button to load the first fragment. We can access the method getSupportFragmentManager() directly because the AppCompatActivity class extends the FragmentActivity class which provides this getter. With the fragment manager we can create fragment transactions.

Fragment transactions provide the methods we need including add, replace, and remove. The replace method is considered to be “safer” because it removes a fragment before adding it.

```
public void onClick(View v) {
    FragmentManager fm = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fm.beginTransaction();
    fragmentTransaction.replace(R.id.mainFrame, new Fragment1());
    fragmentTransaction.commit();
    Log.d(TAG, "onClick");
}
```

getActivity()

We need a way for the new fragments to communicate with the MainActivity. For this discussion we will focus on getActivity().

https://developer.android.com/reference/android/app/Fragment#getActivity()

We can use this method, accessible from within the fragment to access widgets, or other activity related methods, from within the fragment. This method gives us acess to findViewById().

N.B. the getActivity() method is not available in all versions of Android. It was added in API level 11, but then deprecated in API level 28.

Reading the Android Developer documentation is often complicated in part because of the many changes to the interface that have ocurred over the years.

From the fragments we can use getActivity() to access views like this:

```
Switch actSw = getActivity().findViewById(R.id.switch1);
```

TODOs

We haven't been using the template factory methods supplied with the fragment code. **You Should Remove or Fix TODO Code**.

```
/**
 * 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);
    }
}
```

Strip the template code and comments out of the fragment and reduce it to just the basic details. Rename the fragment by right clicking on the class name and selecting "Refactor". Start with something simple, like the following:

```
public class Fragment1 extends Fragment {


    public Fragment1() {
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment1, container, false);
    }
}
```

Setting Fragment onClick handler

The fragment view is returned by onCreateView, we need to store it in a temporary variable. We need the fragment view so that we can set up the onClickListener. If you want to use “this” as an unqualified argument you need to implement the interface View.OnClickListener as part of your fragment declaration.

```
View v = inflater.inflate(R.layout.fragment1, container, false);

Button nextFrag2 = v.findViewById(R.id.buttonNextFrag2);
nextFrag2.setOnClickListener(this);
```

For the first fragment we will access the switch and use it as part of setting up our text. Avoid hardcoding text where possible, in this case we can extend the string stored in the resouce with getString()

```
Switch actSw = getActivity().findViewById(R.id.switch1);
TextView tv = v.findViewById(R.id.helloText);

if (actSw.isChecked()) {
    tv.setText(getString(R.string.hello1) + "\n      Switch = ON");
} else {
    tv.setText(getString(R.string.hello1) + "\n      Switch = OFF");
}
```

The onClick method will access the fragment manager directly. Notice that within the fragment we are not calling “getSupportFragmentManager”. We could access this method through the activity, but we don’t have to. Again, be cautious, the “getFragmentManager” method is part of an older API, and deprecated in API level 28.

https://developer.android.com/reference/android/app/Fragment#getFragmentManager()

The onClick handler for our fragment looks like this:

```
@Override
public void onClick(View view) {

    FragmentManager fm = getFragmentManager();
    Fragment f2 = new Fragment2();
    FragmentTransaction fragmentTransaction = fm.beginTransaction();
    fragmentTransaction.replace(R.id.mainFrame, f2);
    fragmentTransaction.commit();
    Log.d(TAG, " onClick");
}
```

This is almost the same as as the onClick handler for the MainActivity.

Fragment1.java

```
public class Fragment1 extends Fragment
    implements View.OnClickListener {

    public static String TAG = "==Fragment1== (";

    public Fragment1() {
        Log.d(TAG, "constructor");
    }

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

        View v = inflater.inflate(R.layout.fragment1, container, false);

        Button nextFrag2 = v.findViewById(R.id.buttonNextFrag2);
        nextFrag2.setOnClickListener(this);

        Switch actSw = getActivity().findViewById(R.id.switch1);
        TextView tv = v.findViewById(R.id.helloText);

        if (actSw.isChecked()) {
            tv.setText(getString(R.string.hello1) + "\n      Switch = ON");
        } else {
            tv.setText(getString(R.string.hello1) + "\n      Switch = OFF");
        }

        Log.d(TAG, "onCreateView");
        return v;
    }

    @Override
    public void onClick(View view) {

        FragmentManager fm = getFragmentManager();
        Fragment f2 = new Fragment2();
        FragmentTransaction fragmentTransaction = fm.beginTransaction();
        fragmentTransaction.replace(R.id.mainFrame, f2);
        fragmentTransaction.commit();
        Log.d(TAG, "onClick");
    }

    @Override
    public void onPause() {
        super.onPause();
        Log.d(TAG, "onPause");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy");
    }
}
```

Fragment2.java

The second fragment is similar to the first. It uses a different layout file, has a different interpretation of the switch in the parent class, and rather than starting a new fragment it removes the current fragment.

```
public class Fragment2 extends Fragment
        implements View.OnClickListener{

    public static String TAG = "==Fragment2==";

    public Fragment2() {
        Log.d(TAG,"constructor");
    }

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

        View v =  inflater.inflate(R.layout.fragment2, container, false);

        Button remFrag = v.findViewById(R.id.buttonRemFrag);
        remFrag.setOnClickListener(this);

        Switch actSw = getActivity().findViewById(R.id.switch1);
        TextView tv = v.findViewById(R.id.helloText);

        if (actSw.isChecked()) {
            tv.setText(getString(R.string.hello2) + "\n     Switch = True");
        } else {
            tv.setText(getString(R.string.hello2) + "\n     Switch = False");
        }

        Log.d(TAG,"onCreateView");

        return v;
    }

    public void onClick(View v) {

        FragmentManager fm = getFragmentManager();
        FragmentTransaction fragmentTransaction = fm.beginTransaction();
        fragmentTransaction.remove(this);
        fragmentTransaction.commit();
        Log.d(TAG,"onClick");
    }

    @Override
    public void onPause() {
        super.onPause();
        Log.d(TAG, "onPause");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy");
    }
}
```