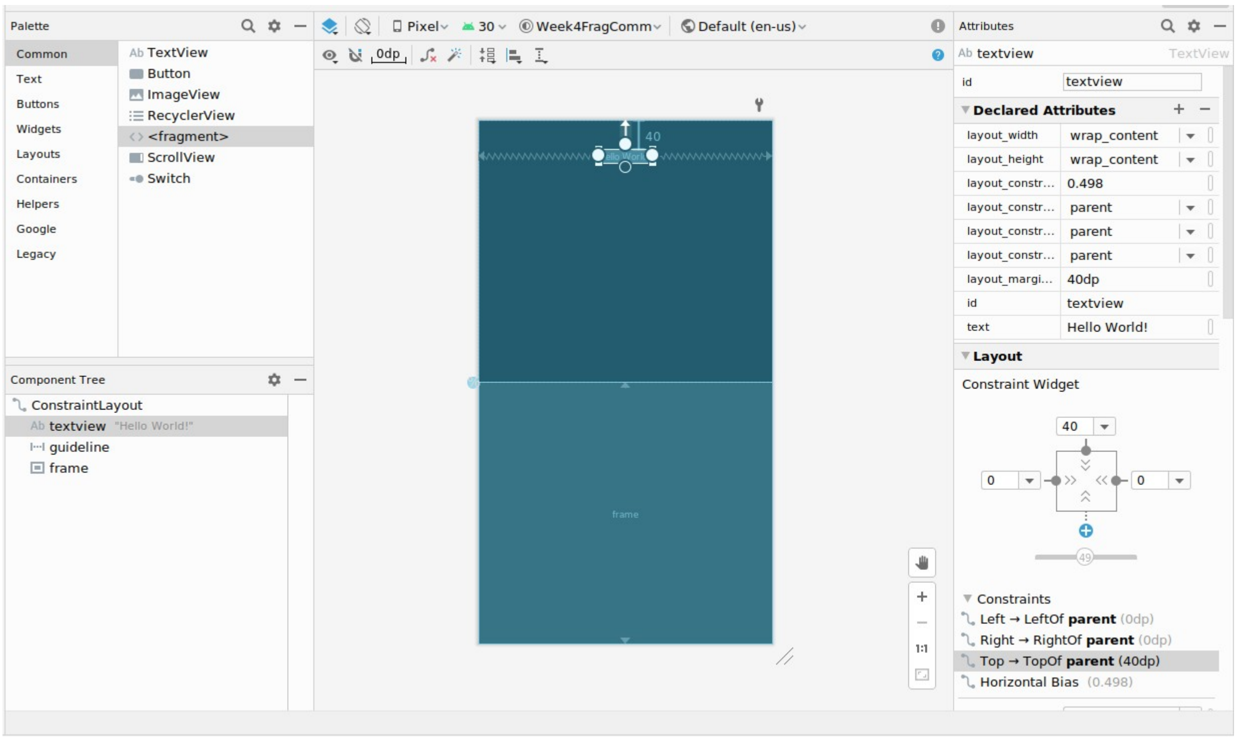
Android Application Development, COMP 10073

Mohawk College, Winter 2021

In the last lecture we set up a simple scenario where a switch in the main activity was used to control the content found in a text view object embedded in a fragment. What happens if we reverse the role of the widgets, i.e. put the switch in the fragment, and put the text view in the main activity?

For the following layout, use a guideline to split the activity, put a text view widget at the top of the activity, and place a frame at the bottom of the activity constrained to the guide. Create a layout for a fragment that includes a switch.



Attributes — textview (TextView)

id	textview

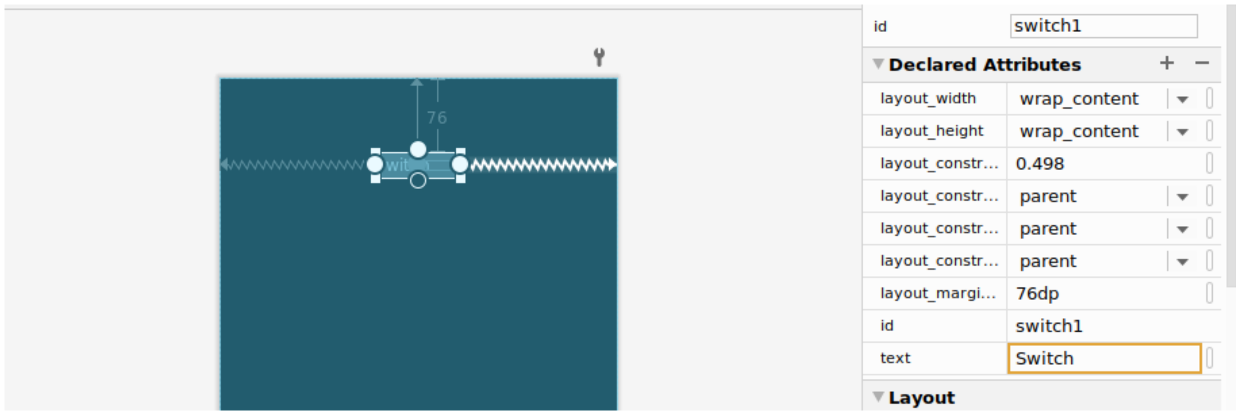
Declared Attributes

layout_width	wrap_content
layout_height	wrap_content
layout_constr...	0.498
layout_constr...	parent
layout_constr...	parent
layout_constr...	parent
layout_margi...	40dp
id	textview
text	Hello World!

Layout

Constraint Widget: 40, 0, 0

Constraints
- Left → LeftOf **parent** (0dp)
- Right → RightOf **parent** (0dp)
- Top → TopOf **parent** (40dp)
- Horizontal Bias (0.498)



id	switch1

Declared Attributes

layout_width	wrap_content
layout_height	wrap_content
layout_constr...	0.498
layout_constr...	parent
layout_constr...	parent
layout_constr...	parent
layout_margi...	76dp
id	switch1
text	Switch

Layout

activity_main.xml

```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=".MainActivity">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text="Hello World!"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <FrameLayout
        android:id="@+id/frame"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline"
        tools:layout_editor_absoluteX="199dp"/>

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

fragment_blank.xml

```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">

    <Switch
        android:id="@+id/switch1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="76dp"
        android:text="Switch"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Add the Switch Handler

https://developer.android.com/reference/android/view/LayoutInflater#inflate(int,%20android.view.ViewGroup,%20boolean)

Simplify the default code for the fragment, strip it down to just one method, like so:

```
public class BlankFragment extends Fragment {
    @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);
    }
}
```

The obvious thing to do is add a handler similar to the one used in the previous discussion.

```
public void onSwitch(View v) {

    Switch sv = (Switch) v;
    TextView tview = findViewById(R.id.textview);

    if (sv.isChecked()) {
        tview.setText("On");
    } else {
        tview.setText("Off");
    }
}
```

When you try to add this code to the fragment, you will quickly discover that findViewById() is not available. Why did it work last time? Previously our fragment was embedded in our activity, effectively it was a child of that activity and so we could lookup its ID through the global context. With the scenario reversed, the child element does not have access to the parent element, the context is unclear.

What to do?

Previously we said that using getters and setters to communicate between Activities was a bad idea because there was no guarantee regarding the lifetime of the activity. The same case does not necessarily hold for fragments.

The activity that spawns a fragment should know the duration of its lifecycle, so while communicating between activities using shared references is unsafe, allowing fragments to communicate via the main activity is more reasonable.

Pass TextView in Through Constructor

On the fragment side we need to provide a reference to the MainActivity reference for the text view. Add a constructor to the fragment that accepts a reference for the TextView widget, i.e.

```
public class BlankFragment extends Fragment {

    private TextView tview;
    public BlankFragment(TextView tview) {
        this.tview = tview;
    }

    @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);
    }

    public void onSwitch(View v) {

        Switch sv = (Switch) v;

        if (sv.isChecked()) {
            tview.setText("On");
        } else {
            tview.setText("Off");
        }
    }
}
```

Now we see better justification for initializing fragments through code. Our initialization is similar to the previous version, except that when we create the fragment we give it our reference to the textview widget.

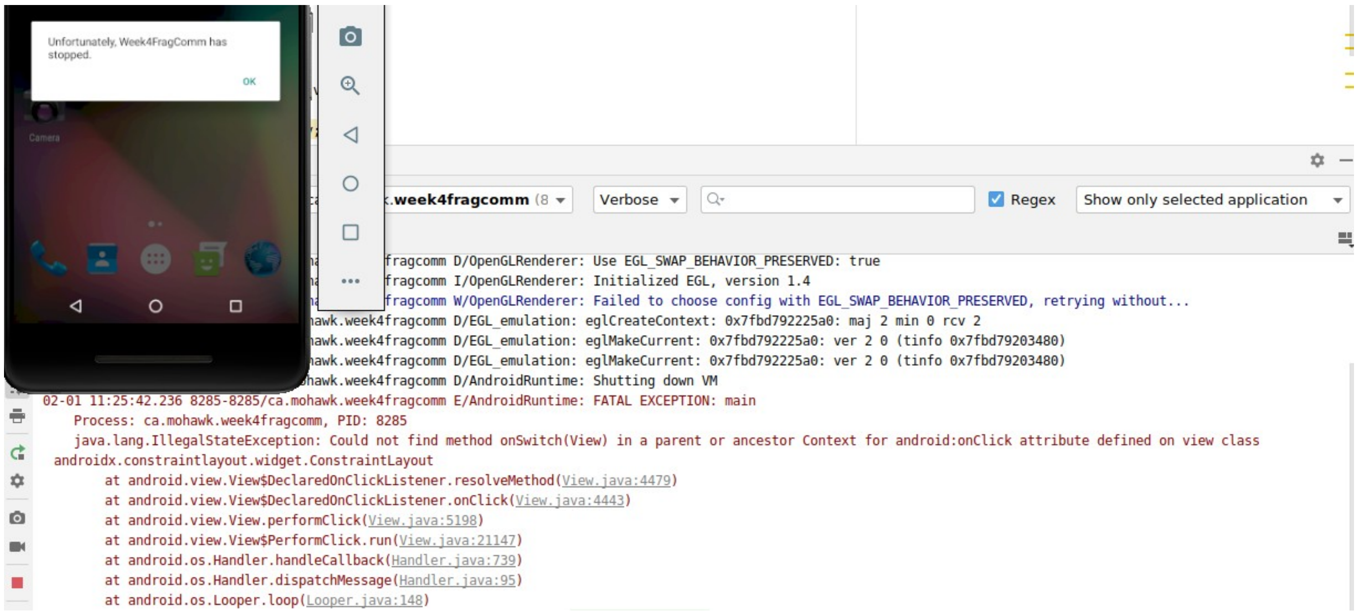
```
public class MainActivity extends AppCompatActivity {

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

        FragmentManager fm = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fm.beginTransaction();
        Fragment myFragment1 = new BlankFragment(findViewById(R.id.textview));
        fragmentTransaction.replace(R.id.frame , myFragment1);
        fragmentTransaction.commit();
    }
}
```

Still not quite right

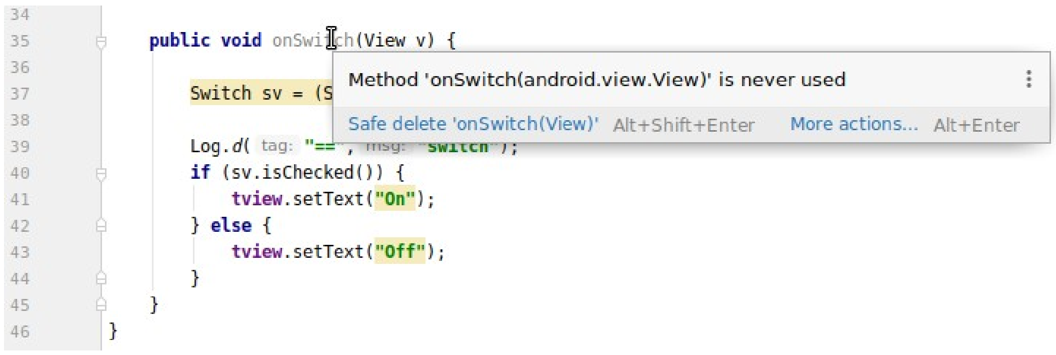
If you set up the button handler through XML, you will find that this version will compile but it won't work. The switch isn't active and if you click anywhere on the bottom fragment it will crash.



Unfortunately, Week4FragComm has stopped.
OK
Camera
.week4fragcomm (8
Verbose
Regex
Show only selected application

```
fragcomm D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
fragcomm I/OpenGLRenderer: Initialized EGL, version 1.4
fragcomm W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
awk.week4fragcomm D/EGL_emulation: eglCreateContext: 0x7fbd792225a0: maj 2 min 0 rcv 2
awk.week4fragcomm D/EGL_emulation: eglMakeCurrent: 0x7fbd792225a0: ver 2 0 (tinfo 0x7fbd79203480)
awk.week4fragcomm D/EGL_emulation: eglMakeCurrent: 0x7fbd792225a0: ver 2 0 (tinfo 0x7fbd79203480)
hawk.week4fragcomm D/AndroidRuntime: Shutting down VM
02-01 11:25:42.236 8285-8285/ca.mohawk.week4fragcomm E/AndroidRuntime: FATAL EXCEPTION: main
    Process: ca.mohawk.week4fragcomm, PID: 8285
    java.lang.IllegalStateException: Could not find method onSwitch(View) in a parent or ancestor Context for android:onClick attribute defined on view class
 androidx.constraintlayout.widget.ConstraintLayout
        at android.view.View$DeclaredOnClickListener.resolveMethod(View.java:4479)
        at android.view.View$DeclaredOnClickListener.onClick(View.java:4443)
        at android.view.View.performClick(View.java:5198)
        at android.view.View$PerformClick.run(View.java:21147)
        at android.os.Handler.handleCallback(Handler.java:739)
        at android.os.Handler.dispatchMessage(Handler.java:95)
        at android.os.Looper.loop(Looper.java:148)
```

You should notice that in the code editor there is a warning that says that the switch handler is unused:



Safe delete 'onSwitch(View)' Alt+Shift+Enter
More actions... Alt+Enter

```
    public void onSwitch(View v) {

        Switch sv = (S

        Log.d( tag: "==", msg: "switch");
        if (sv.isChecked()) {
            tview.setText("On");
        } else {
            tview.setText("Off");
        }
    }
}
```

Method 'onSwitch(android.view.View)' is never used

The solution here is to connect the switch to the fragment programmatically, but we already had trouble with findViewById(). What we are missing here is the fragment's view.

Fragment View

The fragment view is generated by the inflater. Rather than immediately returning the result from the inflater use it to call setOnClickListener(...).

```java
@Override
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View fragView = inflater.inflate(R.layout.fragment_blank,
            container, false);
    Switch sview = fragView.findViewById(R.id.switch1);
    sview.setOnClickListener(this::onSwitch);

    return fragView;
}
```

What about Embedding the Fragment in XML

We aren’t doing anything too fancy with the fragment, so why not drop the fragment manager and embed the fragment directly in the activity_main.xml file?

```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=".MainActivity">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text="Hello World!"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <fragment
        android:id="@+id/fragment"
        android:name="ca.mohawk.week4fragcomm.BlankFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline" />

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

The view that we used to lookup a reference to the switch embedded in the fragment is generated by the inflater. If the fragment is embedded directly in the XML, then we can also look up its view, and add the onClick listener programmatically.

The main activity can be simplified to this:

```
public class MainActivity extends AppCompatActivity {

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

        View fragView = findViewById(R.id.fragment);
        Switch sview = fragView.findViewById(R.id.switch1);
        sview.setOnClickListener(this::onSwitch);
    }

    public void onSwitch(View v) {

        Switch sv = (Switch) v;
        TextView tview = findViewById((R.id.textview));
        if (sv.isChecked()) {
            tview.setText("On");
        } else {
            tview.setText("Off");
        }
    }
}
```

The fragment can be reduced to this

```
public class BlankFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container,
                             Bundle savedInstanceState) {

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