LUMS
A Not-for-Profit University

Android Basics

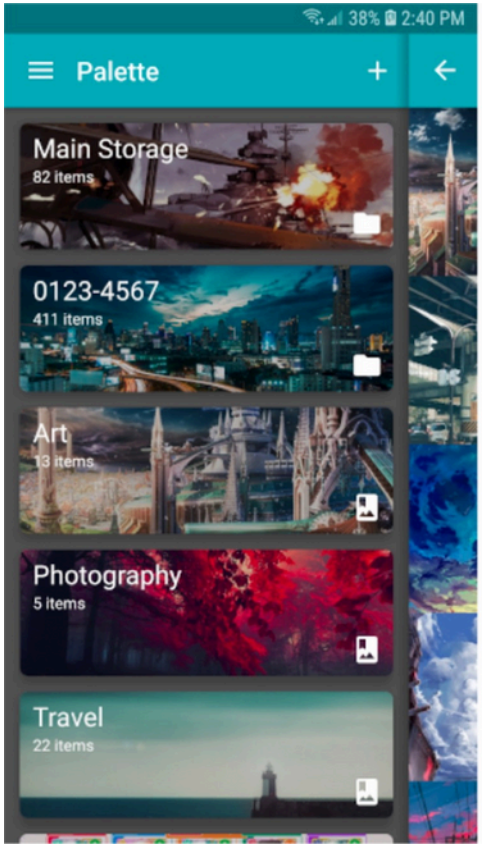
CS360 Lab 2 (Spring 2026)

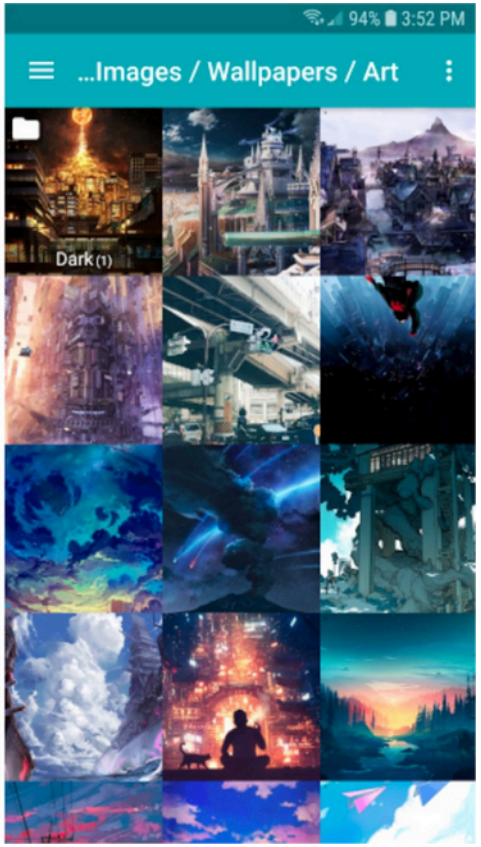
Abdul Ali Bangash

Adapted from CMPUT 301 – Software Engineering (University of Alberta)

Activities

Activity: A **single screen** in an Android app. Each screen shown below is one activity.



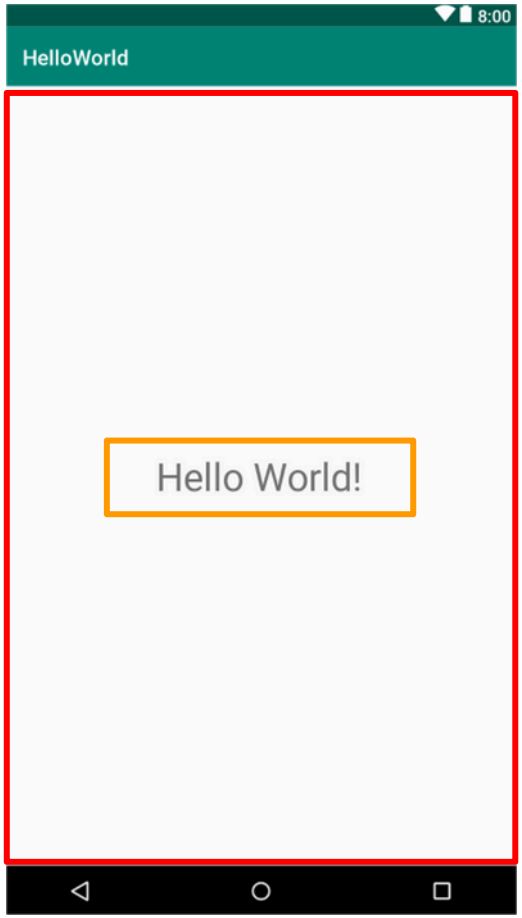


Activities

There are two parts to every Activity. First is the Activity's **layout file**.

Use XML to define UI widgets on the screen

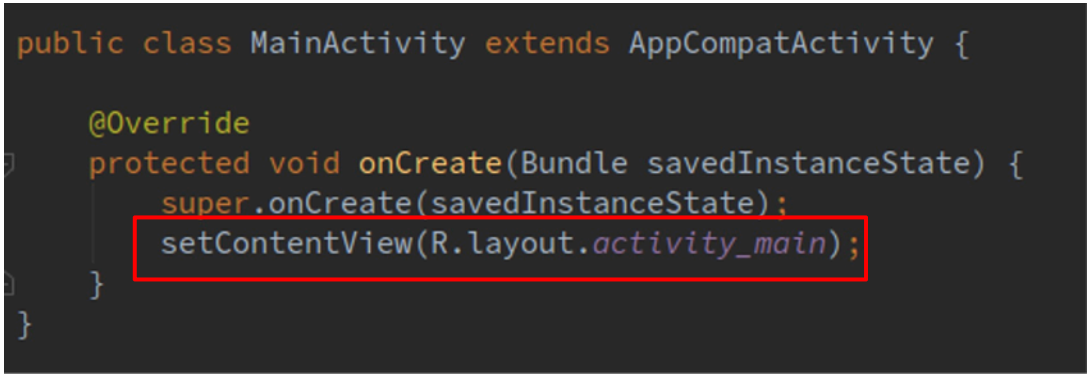
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/hello_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

</LinearLayout>
```



Activities

Next is the actual **Java/Kotlin class** for the Activity.

This is where we define **behaviour** and**logic**for the screen. (e.g. do something when a Button is clicked)



Connect the activity to the layout file using “**setContentView**”

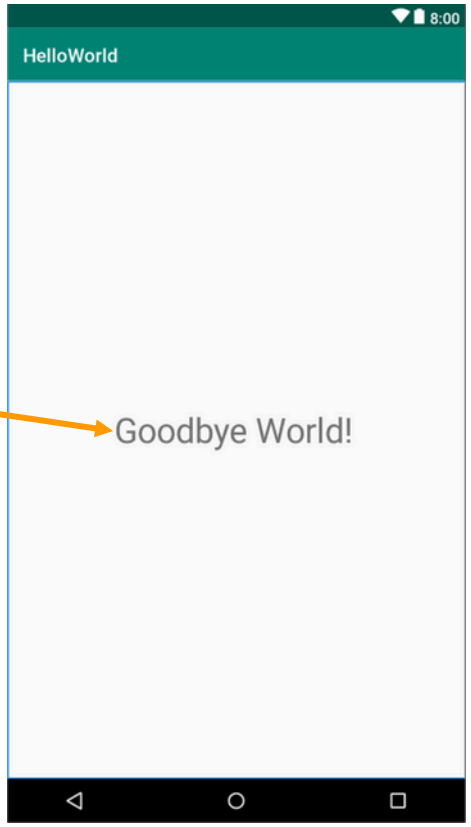
Activities

Now that we've associated the Java class with a layout file, how do we refer to the layout's UI widgets from our code? Use **findViewById.**

```
public class MainActivity extends AppCompatActivity {

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

        TextView helloTextView = findViewById(R.id.hello_text_view);
        helloTextView.setText("Goodbye World!");
    }
}
```

```
<TextView
    android:id="@+id/hello_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
```

activity_main.xml



Views

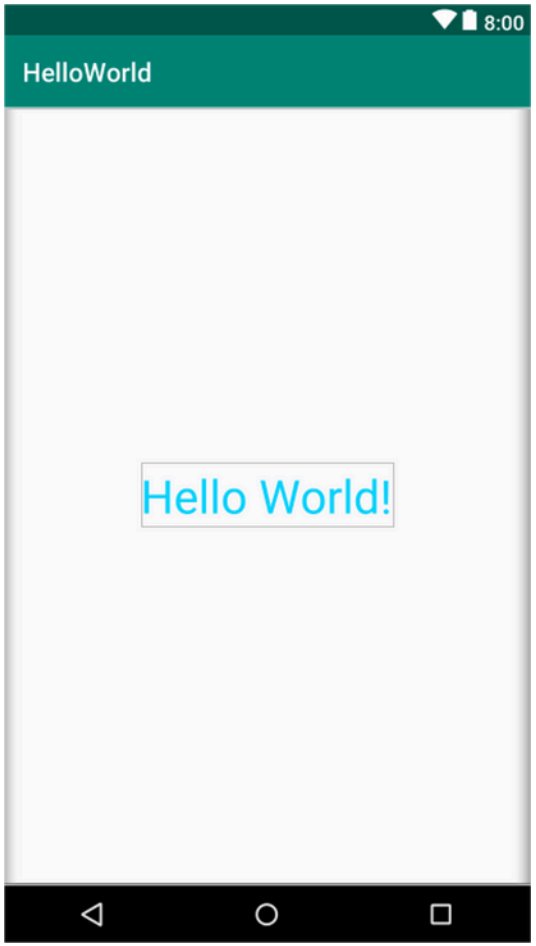
How do we change how the views look? We use XML **attributes**. This lets us control things like the **size, colour**, etc.

Note: If you want to add logic to your views, it must have an id attribute so it can be found in the Activity class using **findViewById()**

```
<TextView
    android:id="@+id/hello_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="36sp"
    android:textColor="@color/sky_blue" />
```

attribute name

attribute value



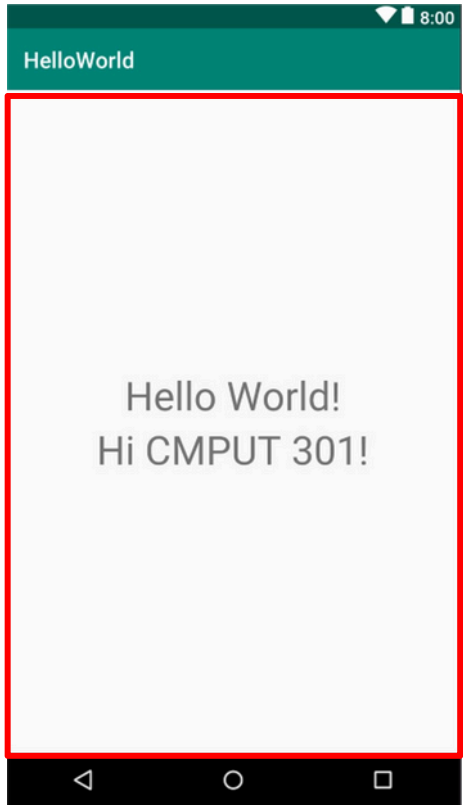
ViewGroups

How does a view know how to position itself on the screen? We put inside of a **ViewGroup** (LinearLayout, ConstraintLayout, etc).

ViewGroups **contain other views**, and **definehowitschildrenviewsshouldbe positioned**.

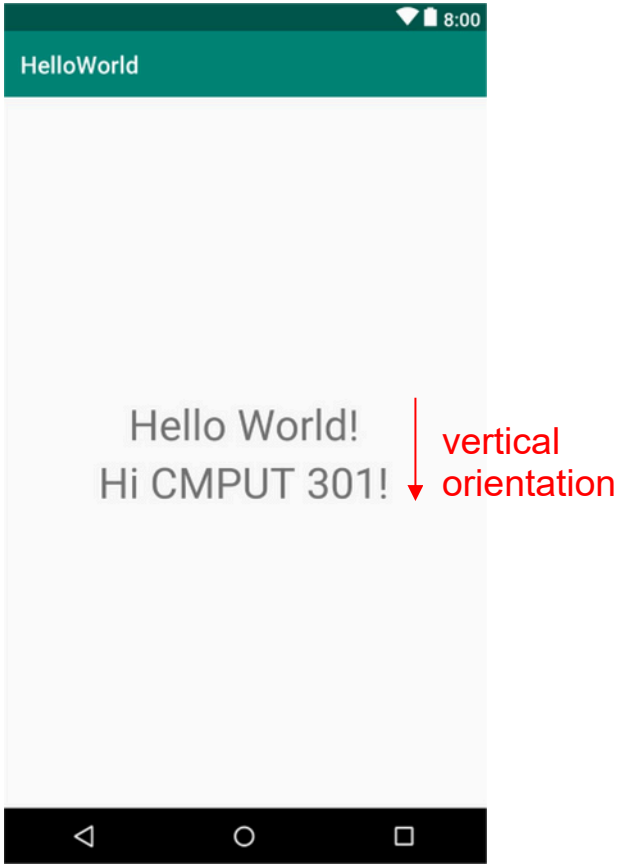
activity_main.xml

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

    <TextView
        android:id="@+id/hello_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="36sp"
        android:text="Hello World!" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hi CMPUT 301!"
        android:textSize="36sp"
        tools:ignore="HardcodedText" />

</LinearLayout>
```

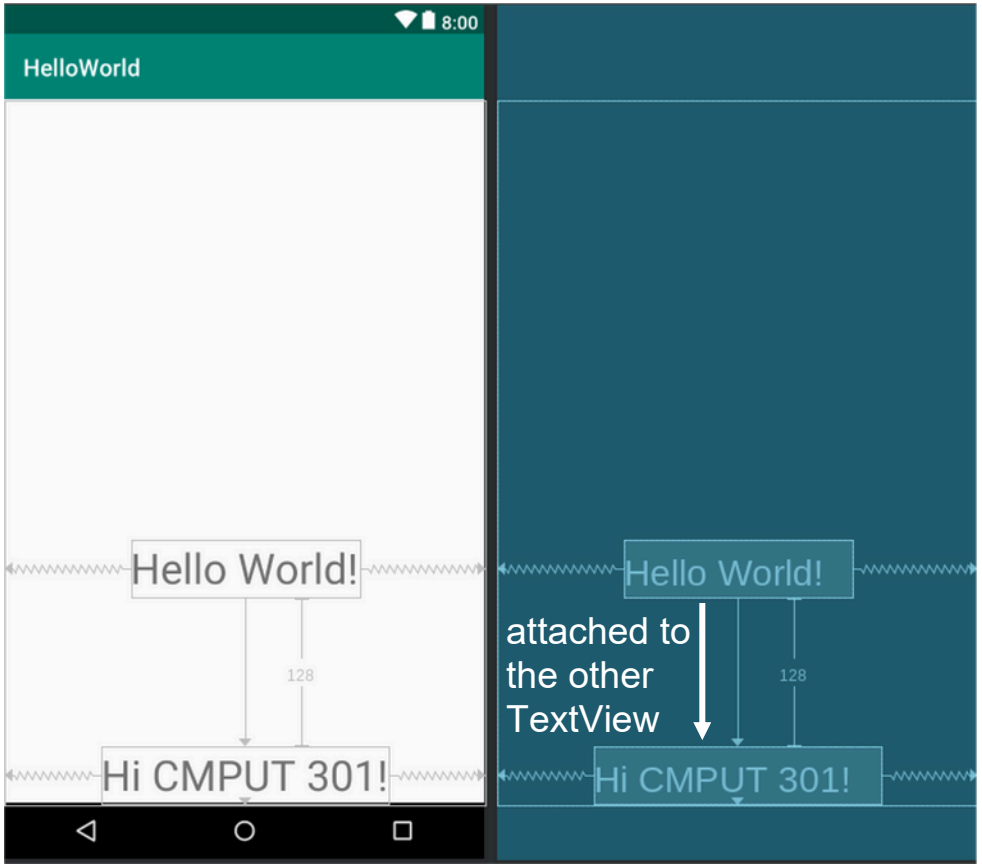


LinearLayout

Examples of ViewGroups



LinearLayout positions children one after another (horizontally or vertically)



ConstraintLayout positions children by attaching them to the edges of other views

Activity Lifecycle

How come the onCreate method runs even though **wenever called the method**?

When you start up an app or navigate to a new activity, the **Android OS calls onCreate**.

Android OS

calls

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

    TextView helloTextView = findViewById(R.id.hello_text_view);
    helloTextView.setText("Goodbye World!");
}
```

Activity Lifecycle

onCreate is similar to the **main functionin C / C++** . The OS calls it when we run the program.

OS

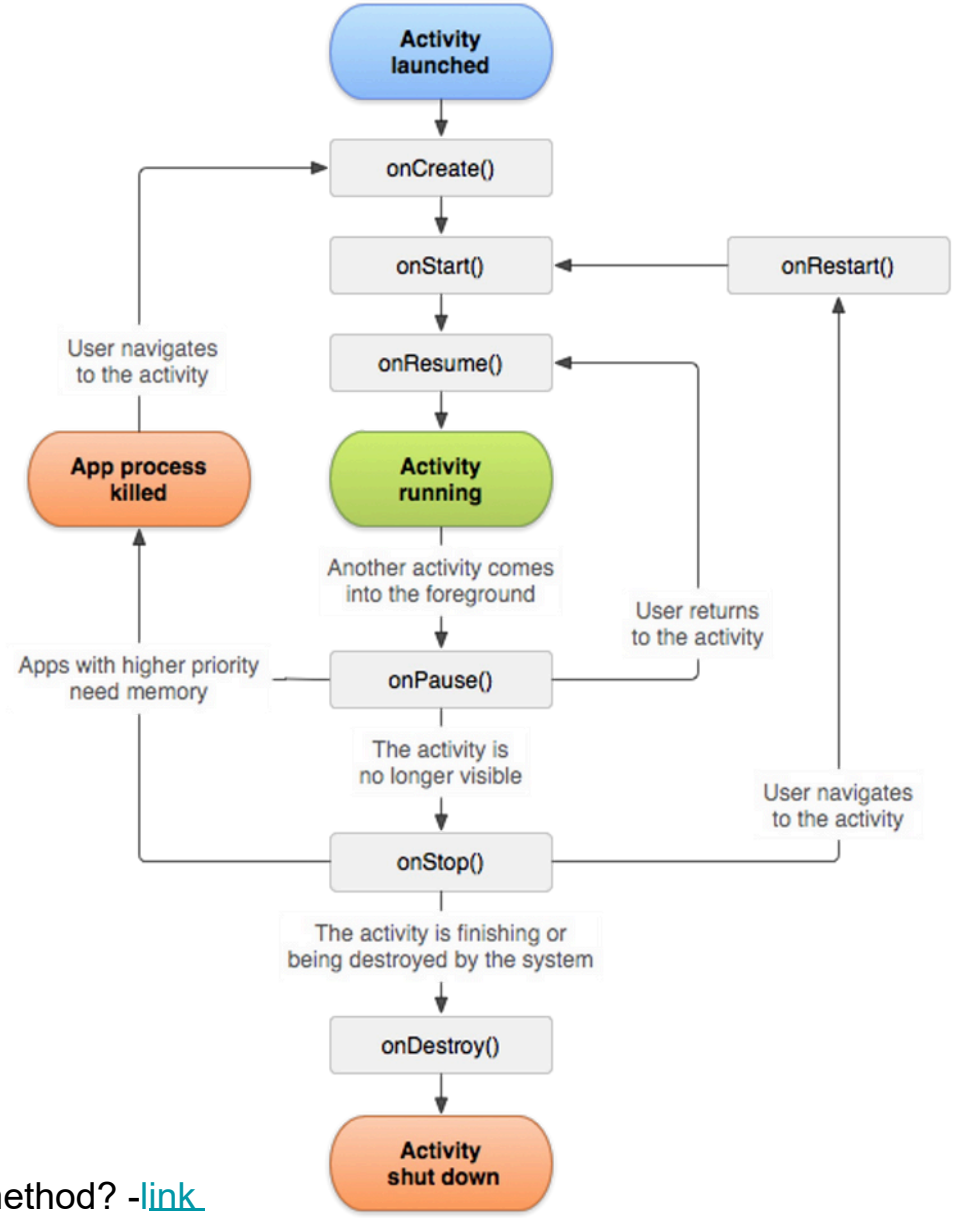
calls

```c
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
}
```

Activity Lifecycle

In fact, the **Android OS calls a bunch of methodson the Activity**for us.

These are an **Activity's lifecycle methods**.



What code should we write in each lifecycle method? -link

Activity Lifecycle

Never call these methods yourself. Leave it to the Android OS.

Our job is simply to override the methods and add our own logic.

```
public class MainActivity extends AppCompatActivity {
    override, then add additional logic
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    protected void onStart() {
        super.onStart();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onStop() {
        super.onStop();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }
}
```

Other concepts you should know for Assignment 1

- how to do something when a view is clicked (**View.OnClickListener**)
- Use of **Intents** to navigate to another activity
- How to **send data between activities** (send it through the Intent!)
- **Android Manifest**
- How to display a list of items (**ListView, ListAdapter**)

We'll go over these in the lab.

References

Android Icon-Made by Freepikfrom www.flaticon.com.

OS Icons(discdepotdundee.co.uk)

Activity Lifecycle Diagram-Understand the Activity Lifecycle