# The X Course: Android

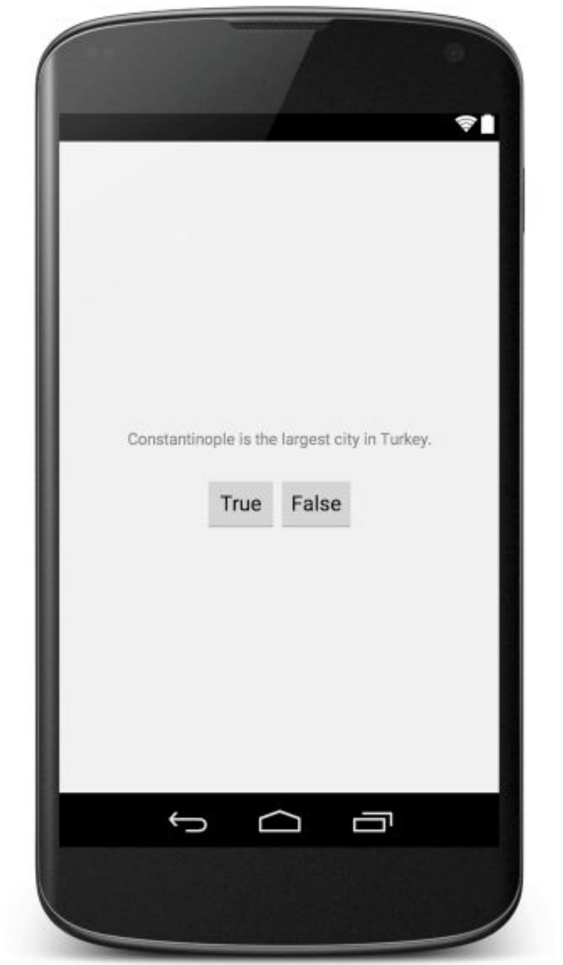
Session 2

# Agenda

- Create our First app: GeoQuiz.
- More on Layouts.
- The Activity Lifecycle.

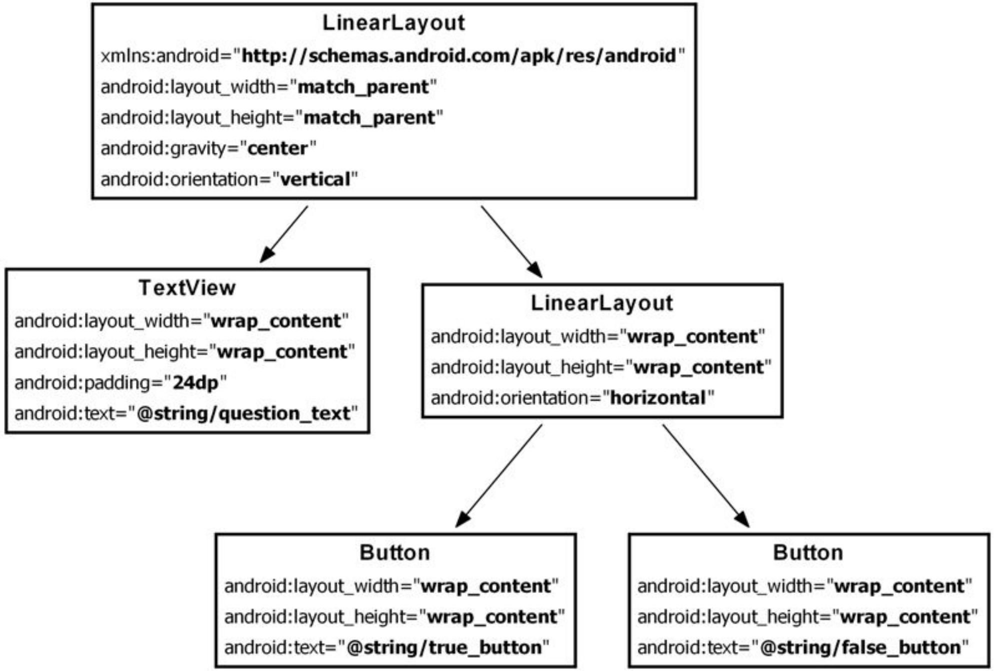
# GeoQuiz: Description

The user presses True or False to answer the question on screen, and GeoQuiz provides instant feedback.



Constantinople is the largest city in Turkey.

# Building the Layout !



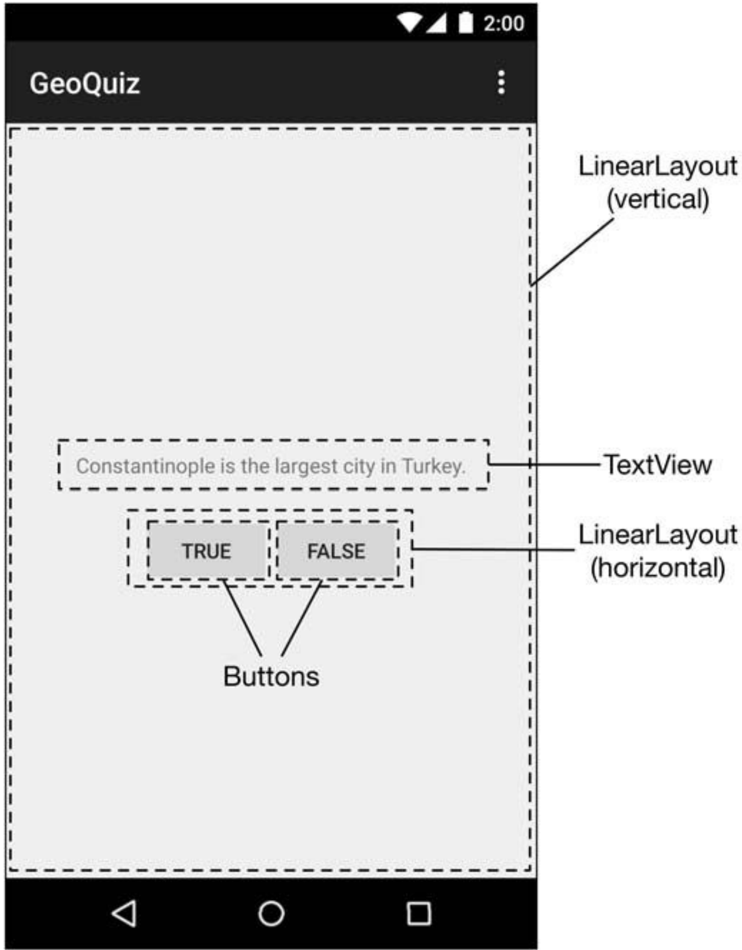
LinearLayout
xmlns:android="**http://schemas.android.com/apk/res/android**"
android:layout_width="**match_parent**"
android:layout_height="**match_parent**"
android:gravity="**center**"
android:orientation="**vertical**"

TextView
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:padding="**24dp**"
android:text="**@string/question_text**"

LinearLayout
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:orientation="**horizontal**"

Button
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:text="**@string/true_button**"

Button
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:text="**@string/false_button**"

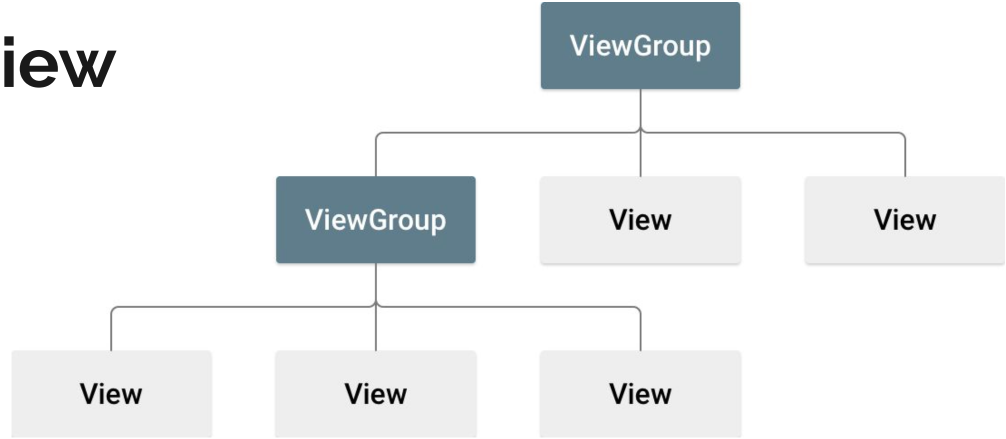

2:00
GeoQuiz
Constantinople is the largest city in Turkey.
TRUE
FALSE
LinearLayout (vertical)
TextView
LinearLayout (horizontal)
Buttons

# Coding the way through the App !

- Need to assign onClicklisteners on our two buttons.
- Add appropriate code for our listeners. (Show Pop-ups for now )

# More on Layouts: Overview

- Layout simply defines UI of screen.
- Layouts is Tree Hierarchical.
- View is a single element on screen.
- Viewgroup is an invisible container for elements



# More on Layouts: Some Rules

- Each layout file must contain exactly one root element (View or ViewGroup).
- You should load the layout resource from your app code like this.

```
public void onCreate(savedInstanceState: Bundle) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

# More on Layouts: Attributes

- Every element supports their own variety of XML attributes.
- Attribute "ID" is used to get elements in code and you will use it very often !

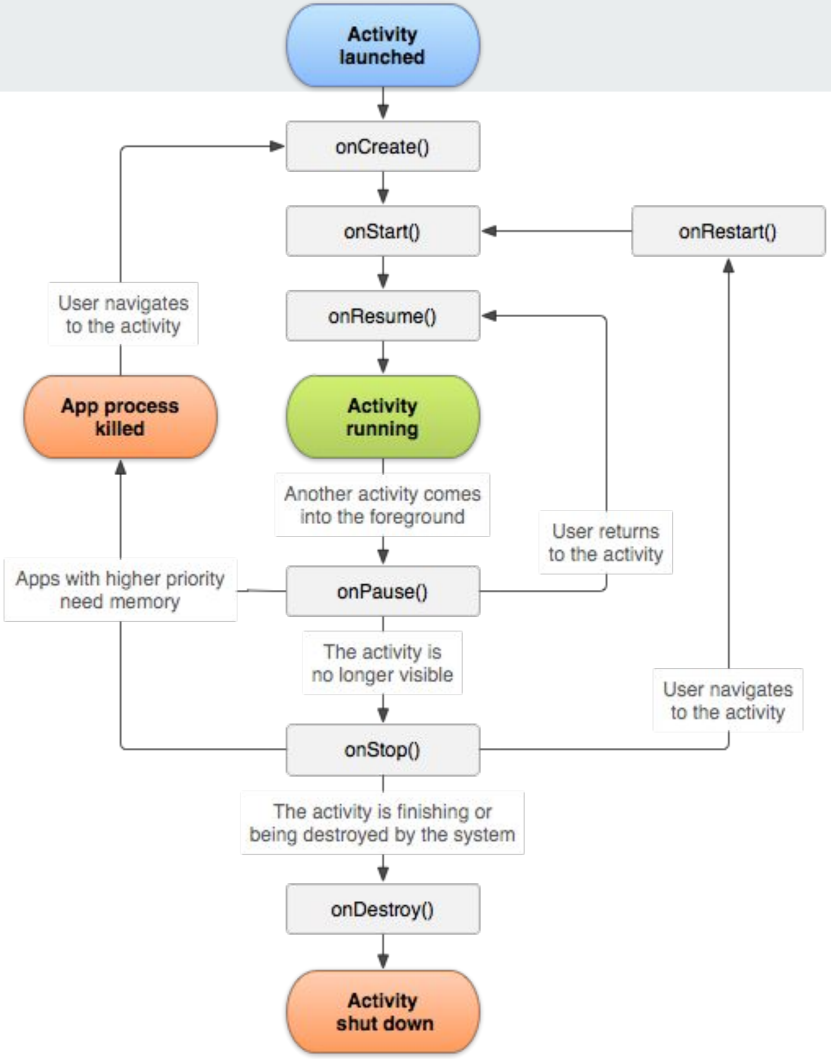
```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

```
Button myButton = (Button) findViewById(R.id.my_button);
```

# More on Layouts: Further Readings

- https://developer.android.com/guide/topics/ui/declaring-layout#java
- https://medium.com/androiddevelopers/layouts-attributes-and-you-9e5a4b4fe32c
- https://www.freecodecamp.org/news/how-to-make-sense-of-the-many-android-layouts-693b262706e0/

# The Activity Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

# States and Callbacks

- Callbacks allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.
- Within the lifecycle callback methods, you can declare how your activity behaves.

# onCreate()

- Must implement this callback.
- Fires when the system first creates the activity.
- Perform basic application startup logic that should happen only once.

# onStart()

- Activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive.
- Where the app initializes the code that maintains the UI.
- Completes very quickly.

# onResume()

- Activity comes to the foreground.
- The app stays in this state until something happens to take focus away from the app.
- When an interruptive event occurs, the activity enters the *Paused* state.
- Initialize components that you release during onPause().

# onPause()

- First indication that the user is leaving your activity. (no longer in the foreground)
- Use it to pause or adjust operations that should not continue whole not in foreground.
- onPause() execution is very brief, and does not necessarily afford enough time to perform save operations. DO NOT USE FOR INTENSIVE SHUT DOWN OPERATIONS!

# onStop()

- When your activity is no longer visible to the user.
- In the onStop() method, the app should release or adjust resources that are not needed while the app is not visible to the user.
- You should also use onStop() to perform relatively CPU-intensive shutdown operations.

# onDestroy()

- The system invokes this callback either because the activity is finishing or the system is temporarily destroying the activity due to a configuration change. ( isFinishing() method)
- If the activity is finishing, onDestroy() is the final lifecycle callback the activity receives.
- If onDestroy() is called as the result of a configuration change, the system immediately creates a new activity instance and then calls onCreate() on that new instance in the new configuration.

# The Activity Lifecycle: Further Readings

- https://developer.android.com/guide/components/activities/activity-lifecycle
- http://blog.dancrisan.com/a-tiny-intro-to-android-activities-lifecycle
- https://blog.mindorks.com/android-activity-lifecycle