# Android View System

CSCI3310 Mobile Computing & Application Development

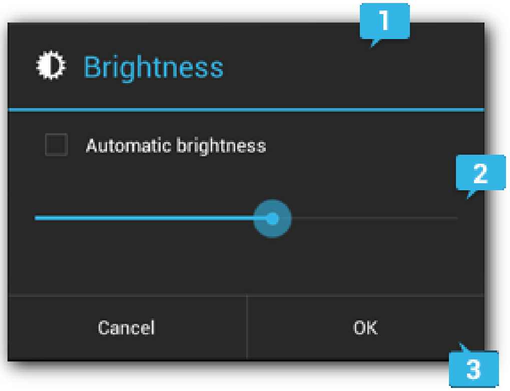
# Outline

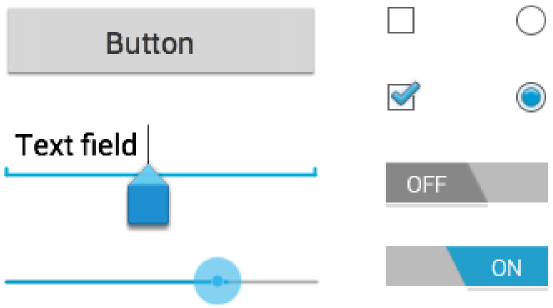
- View and Layout Inflation
- View and ViewGroup
- View System
- Design Patterns in View

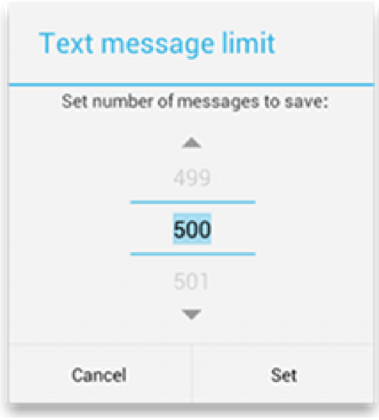
# Views – UI Widgets

Over 100 UI Controls :

- TextView, EditText, Button, CheckBox, RadioButton, ToggleButton, and Spinners for multiple options

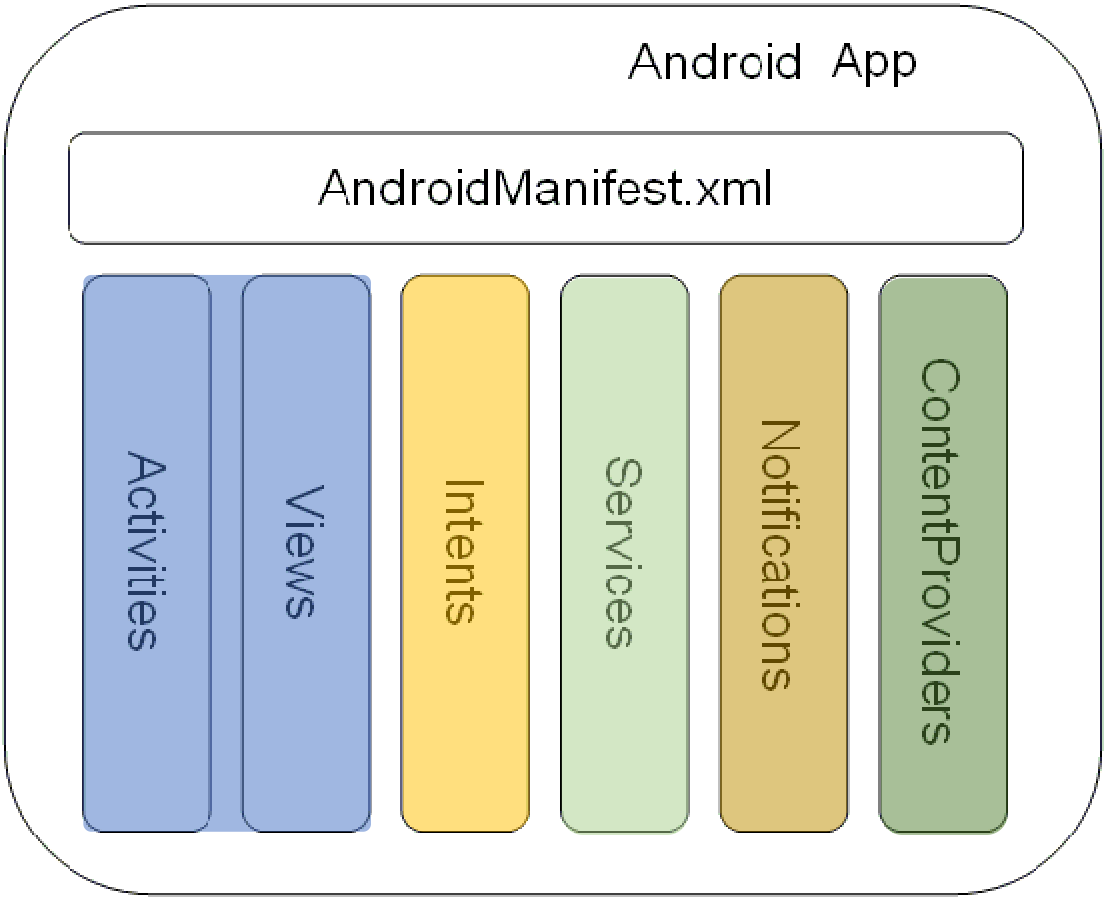






# Android App Anatomy

- Android App is **components** based



Revisited

# AndroidManifest.xml

- Describes the fundamental characteristics of an app and each of its components, the default looks like this:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.cuhk.ypchui.helloworld">

    <uses-sdk
        android:minSdkVersion="20"
        android:targetSdkVersion="26" />

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Revisited

# Android Project Structure

Project structure in Android Studio

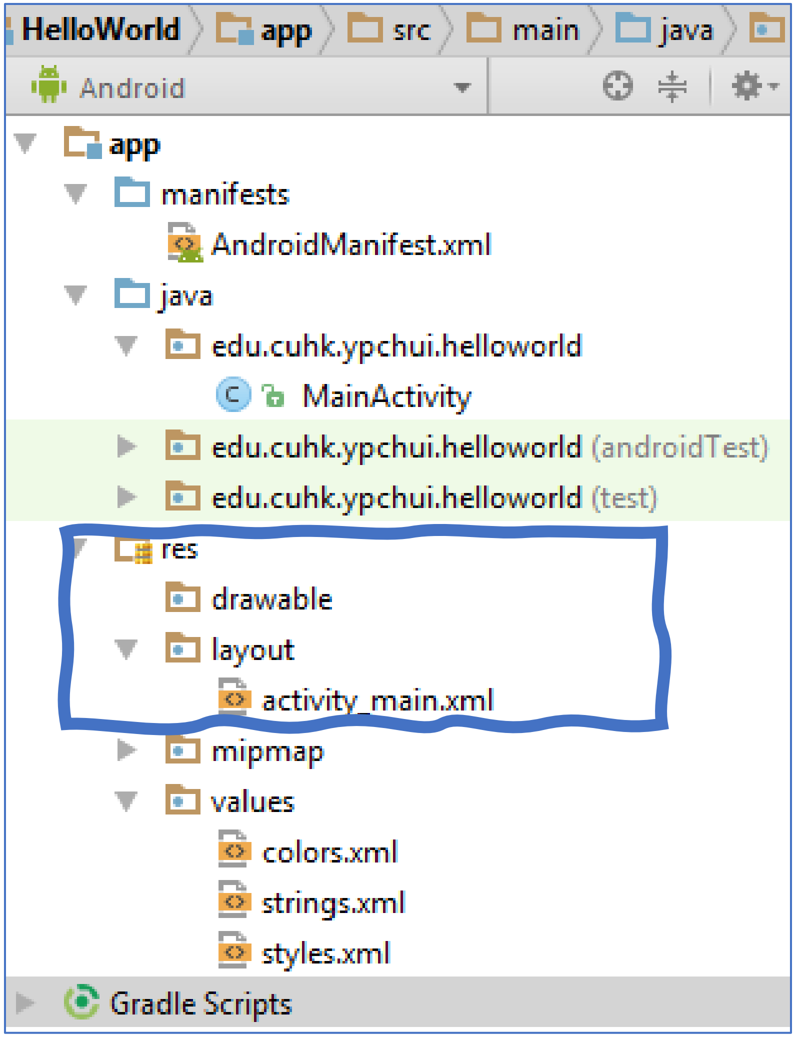
- AndroidManifest.xml
- java
  - MainActivity.java

    *[the main **Controller** entry]*
- res
  - layout/activity_main.xml

    *[the **Views** are here]*
  - values/strings.xml

    *[for different languages]*
  - values/styles.xml

    *[for different UI styles]*



HelloWorld › app › src › main › java ›

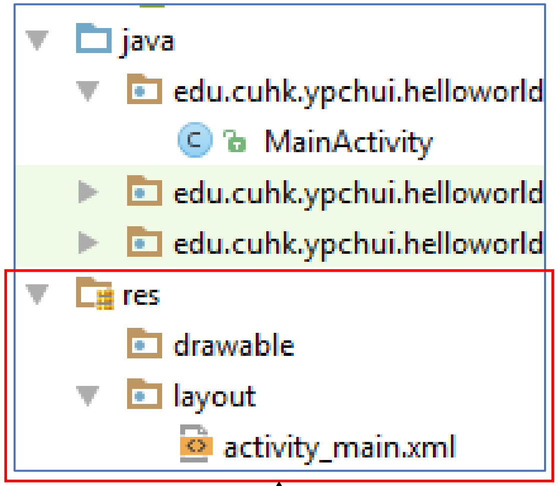
Android

```
app
  manifests
    AndroidManifest.xml
  java
    edu.cuhk.ypchui.helloworld
      MainActivity
    edu.cuhk.ypchui.helloworld (androidTest)
    edu.cuhk.ypchui.helloworld (test)
  res
    drawable
    layout
      activity_main.xml
    mipmap
    values
      colors.xml
      strings.xml
      styles.xml
Gradle Scripts
```

Revisited

# Android Java (java/)

- Contains the source, separated by package names
- All the Activity class are in java/ e.g. the default empty activity MainActivity.java looks like this:



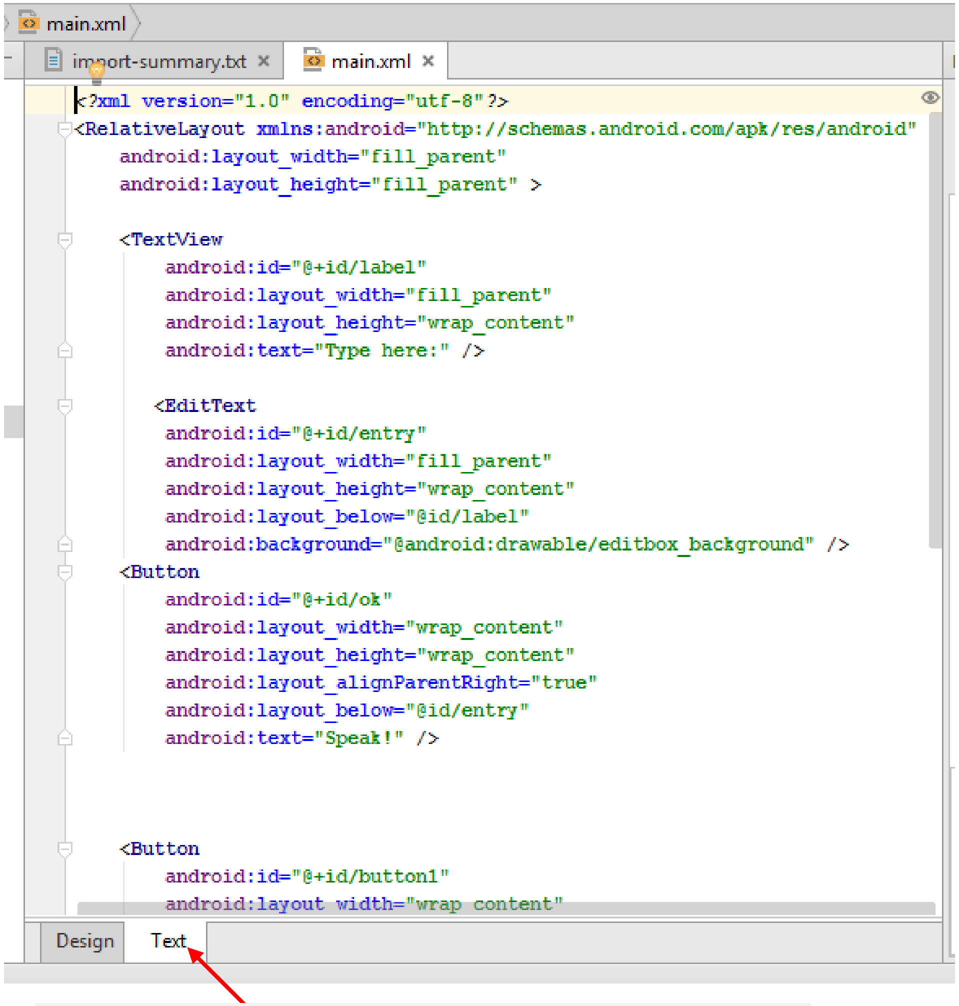
java
edu.cuhk.ypchui.helloworld
MainActivity
edu.cuhk.ypchui.helloworld
edu.cuhk.ypchui.helloworld
res
drawable
layout
activity_main.xml

```java
package edu.cuhk.csci3310.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

link to res/

# XML

- Most UI elements & other assets are specified using XML files
- IDE can also display the actual display by parsing the corresponding file
- decouple the presentation view from the application logic

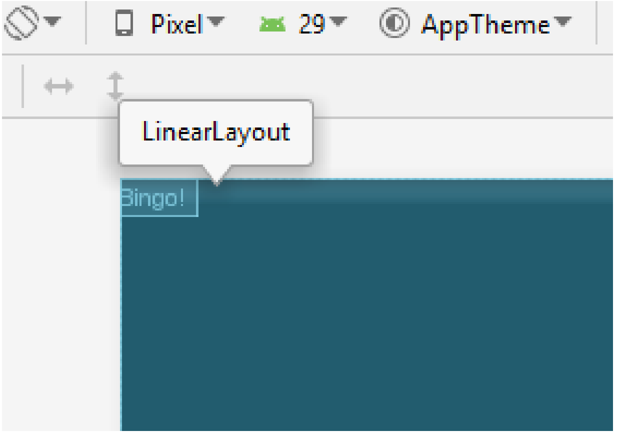


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:" />

       <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label"
        android:background="@android:drawable/editbox_background" />
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/entry"
        android:text="Speak!" />


    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
```

Switch between xml & graphical view

# Layout created in XML

```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <TextView
      android:text="Bingo!"
       ... />
</LinearLayout
```



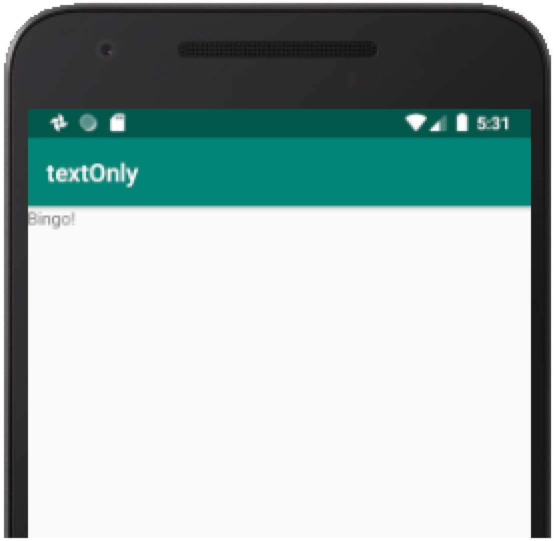
When the app is compiled, each XML layout file is compiled into a View resource. The layout resource loaded by calling setContentView() in the Activity.onCreate() callback:

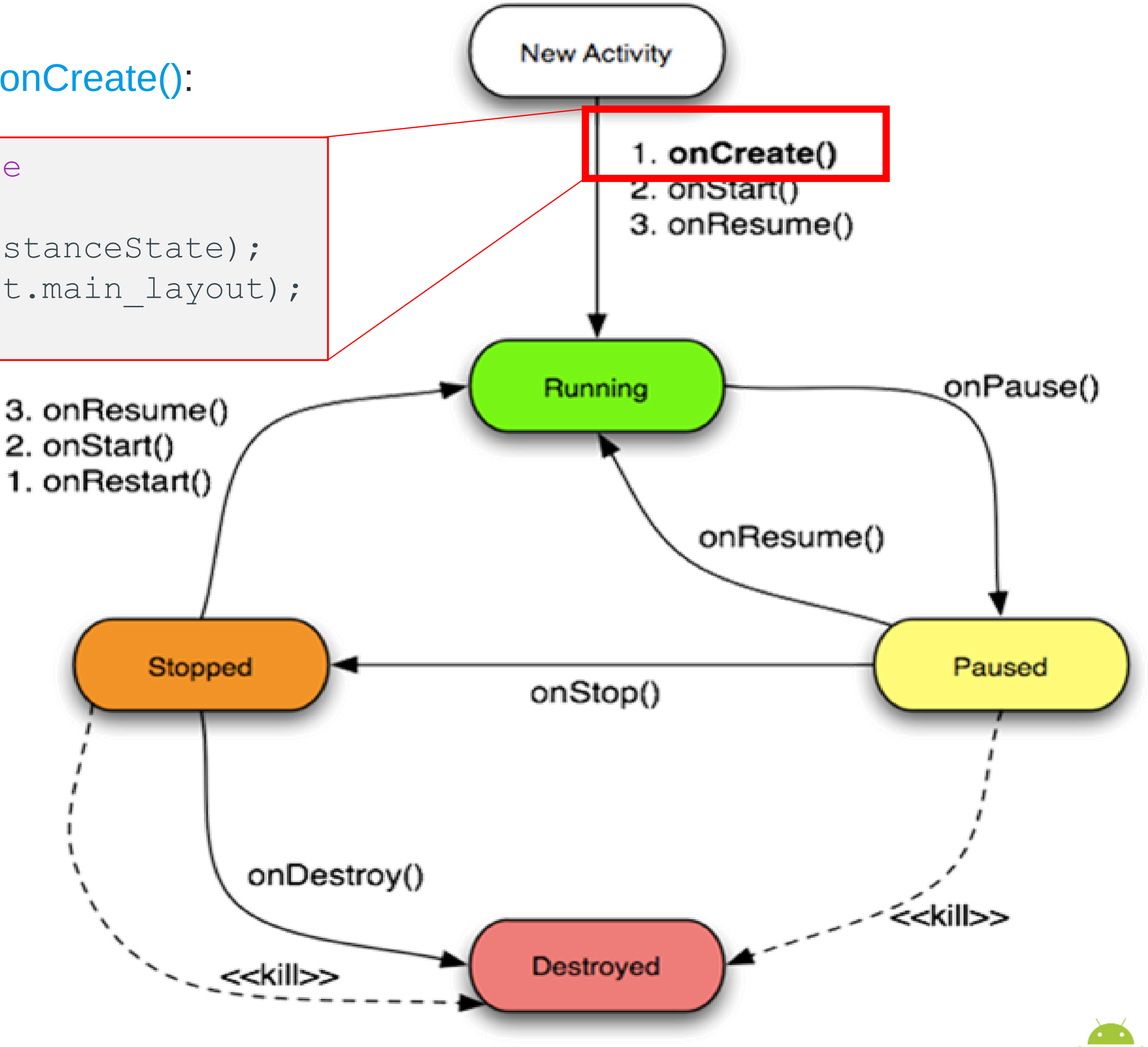
```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

# Activity Lifecycle

Loading XML layout file Activity.onCreate():

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

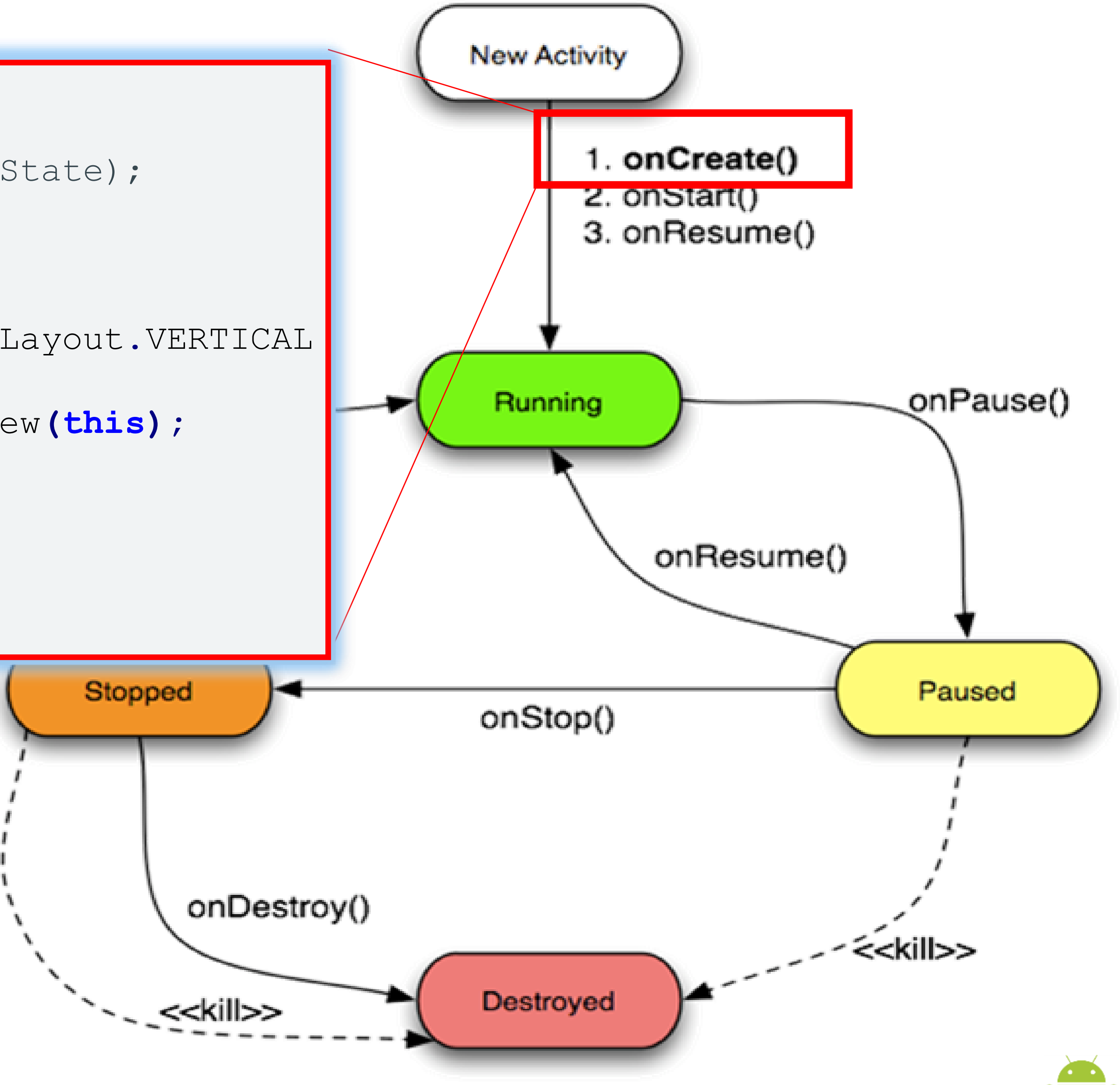




# Activity Lifecycle

Code layout directly in Activity.onCreate():

```
public void onCreate(Bundle
savedInstanceState) {
  super.onCreate(savedInstanceState);

  LinearLayout linearL = new
LinearLayout(this);
linearL.setOrientation( LinearLayout.VERTICAL
);
  TextView myText = new TextView(this);
  myText.setText("Bingo!");
  linearL.addView(myText);
  setContentView(linearL);

}
```



# Layout Inflate from XML to code

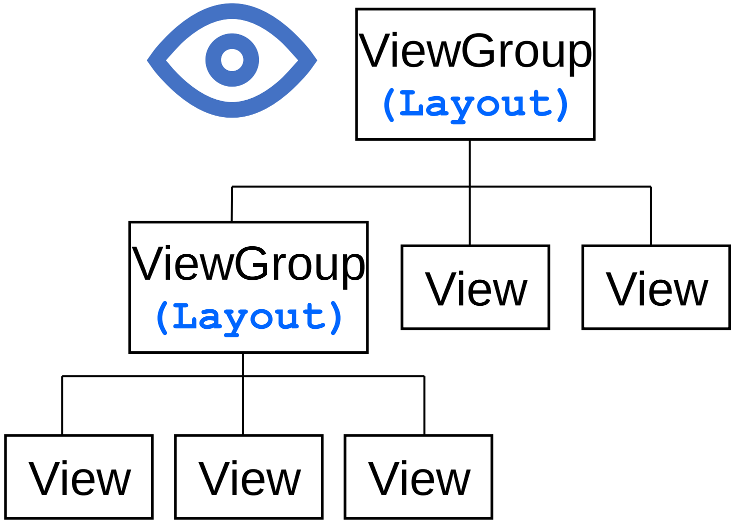
```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <TextView
      android:text="Bingo!"
       ... />
</LinearLayout
```

Internally, a LayoutInflater will Instantiate a layout XML file into its corresponding View objects.

```
...
  LinearLayout linearL = new
LinearLayout(this);
linearL.setOrientation( LinearLayout.VERTICAL
);
  TextView myText = new TextView(this);
  myText.setText("Bingo!");
  linearL.addView(myText);
  setContentView(linearL);
```

# Framework

- Activity and Task Design
  - Activities : basic, independent building blocks of applications
- ViewGroup
  - a special view that can contain other views (called children)
  - ViewGroup arranges their children by Layouts.
- View
  - base class for layouts and views containers.

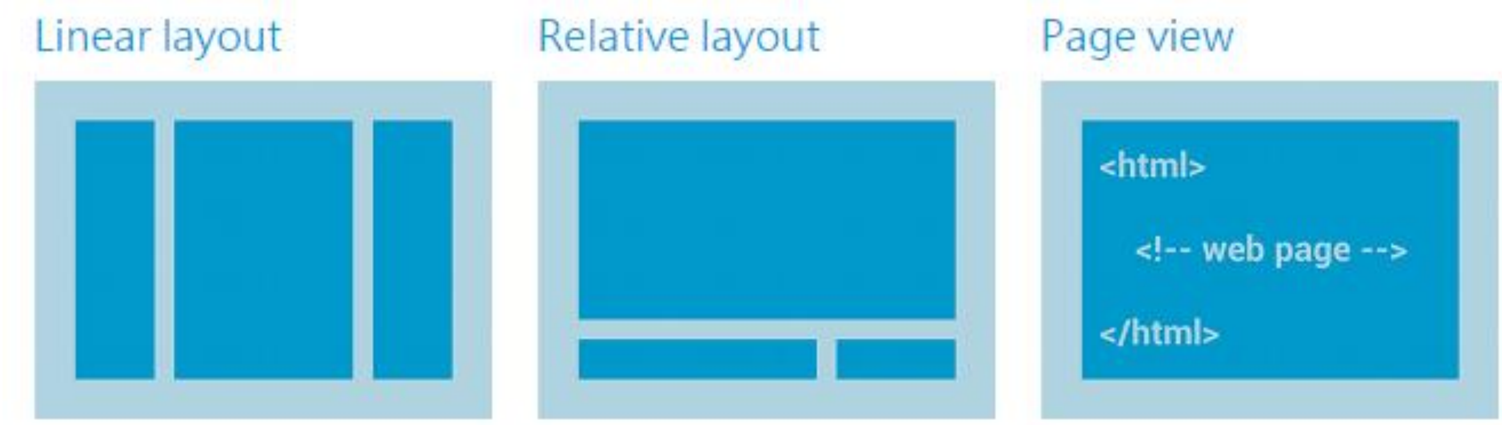


# layout

- Layouts- solution for different pixel densities, dimensions, or aspect ratios
- Typical Android devices allow changing the screen orientation (portrait or landscape) while applications are running, so the layout infrastructure needs to be able to respond on the fly.
- As Android inflates the Layout, it uses the developer requests to come up with a screen layout that best approximates what the developer has asked for.

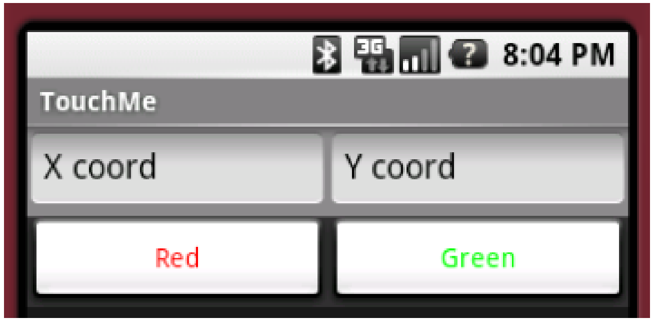
# layout

- Linear layout configures underlying objects into a single **horizontal or vertical** column
- Relative will specify the **relative position between lower or upper** objects e.g. A is located left of B
- Page view **display web** pages



# Cascaded Layouts

- *View using a cascaded layout resource*



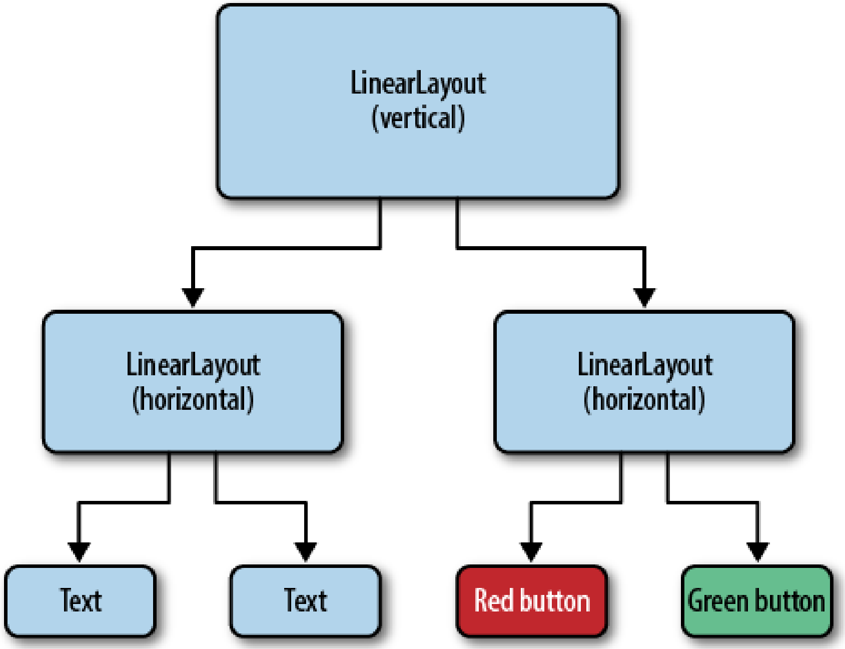
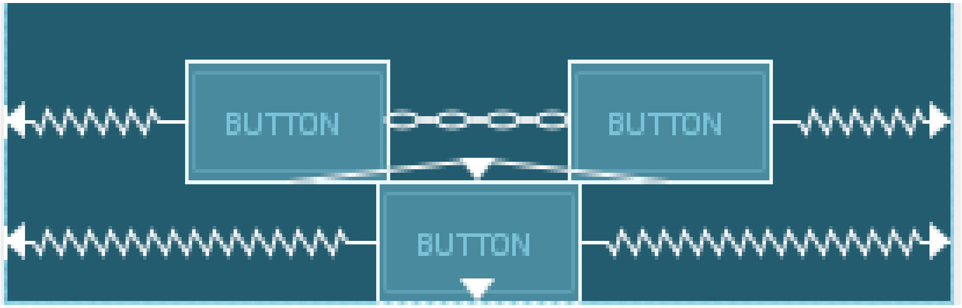


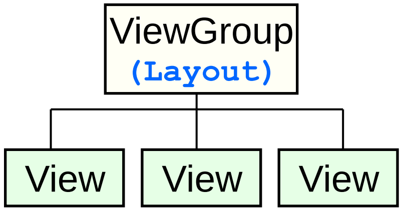
*Image Source: Programming Android(second edition) Chapter 6*

# Constraint Layout

- layout with a flat view hierarchy
- views are laid out according to relationships between sibling views and the parent layout



# The Tree of Views



- View: object that draw itself to the screen
- ViewGroups: containers of views
- Layout: Views are arranged and displayed on the screen according to.
  - Views and Layouts both have attributes that can **either be defined in Java source code or in the XML** file associated with the Activity
  - When the attributes are in an XML file, they are "inflated" at runtime, meaning that they are applied to their respective Views by the Android framework to **determine how the Views look and operate.**

# How drawing of Views may look like?

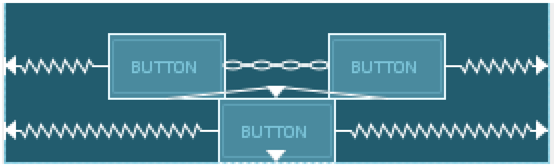
To draw a button, at least go through:

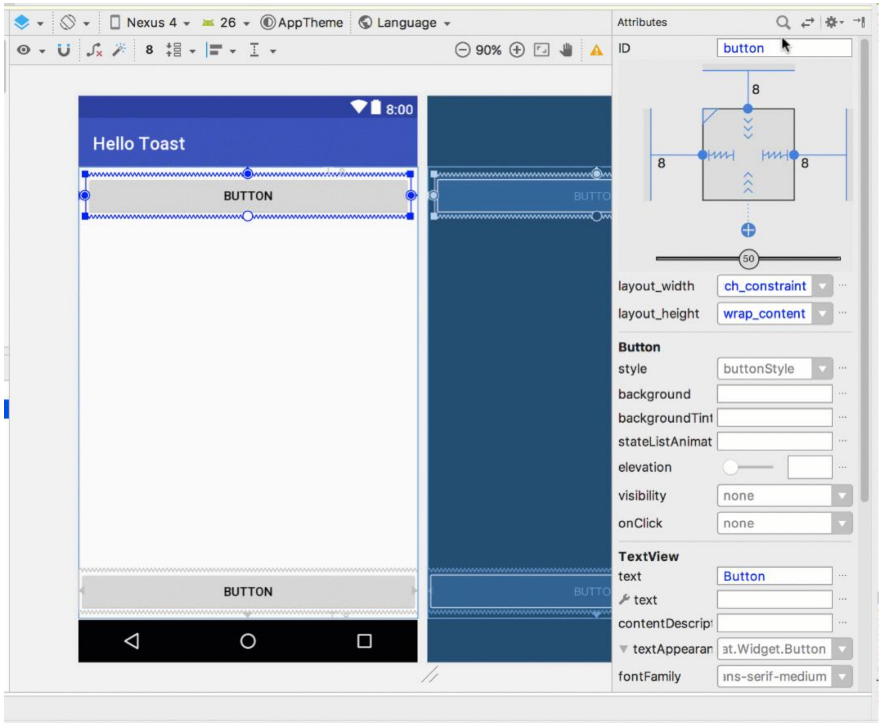
**Measure pass**

1. Calculate the width / height based on text size and number of characters
2. Add margin to calculate the overall space required

**Draw pass**

1. Check the background/text color
2. Draw the button based on all color & sizing info





Attributes

ID: button

layout_width: ch_constraint

layout_height: wrap_content

Button

style: buttonStyle

background

backgroundTint

stateListAnimat

elevation

visibility: none

onClick: none

TextView

text: Button

text

contentDescript

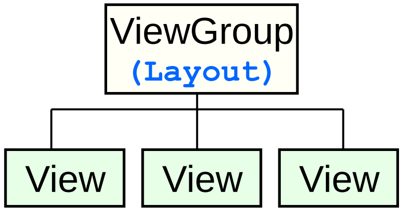
textAppearan: at.Widget.Button

fontFamily: ans-serif-medium

# How to draw the tree of Views?

Given the tree of Views, e.g. **Tree**<View>, the drawing pseudocode may look like:

1. walking the tree from root node
2. for each node of the tree
   a) call View's **measure()**
   b) call View's **draw()**



A pre-order traversal can ensure parent is drawn before its children

# View Lifecycle

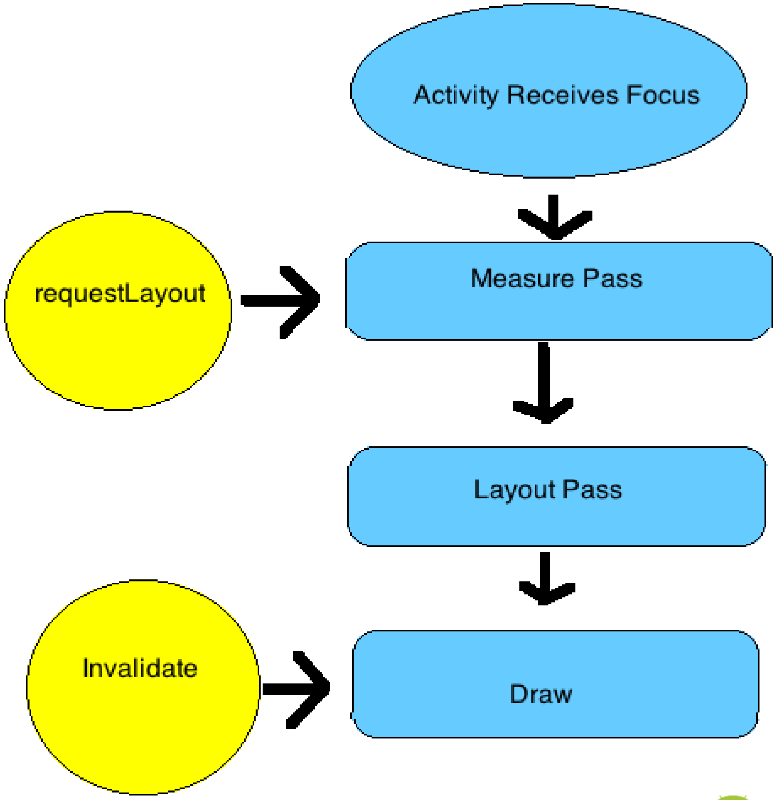
Typically, not necessary to work on View lifecycle details except for implementing a custom view

Minimally, needs an Override on

- onMeasure()
- onDraw()

might also need to work with

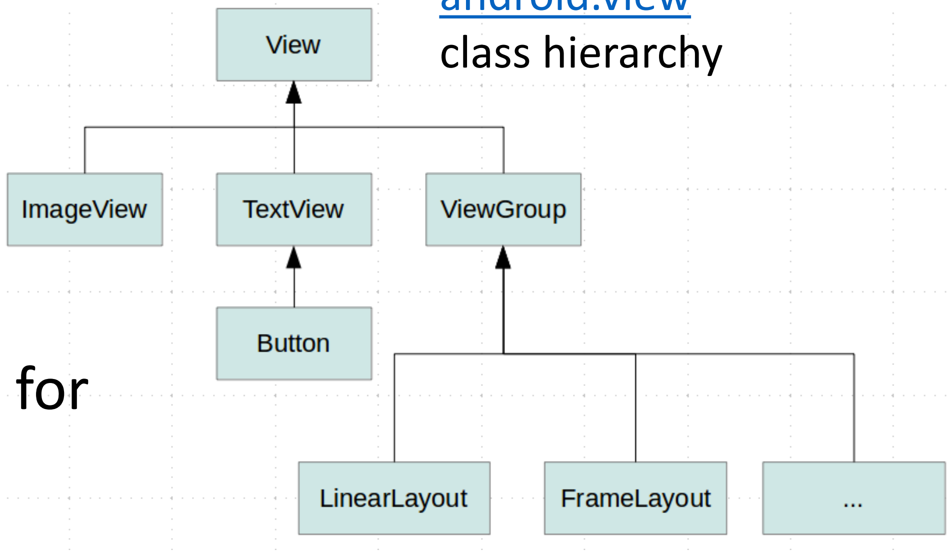
- onSizeChanged()
- onAttach()



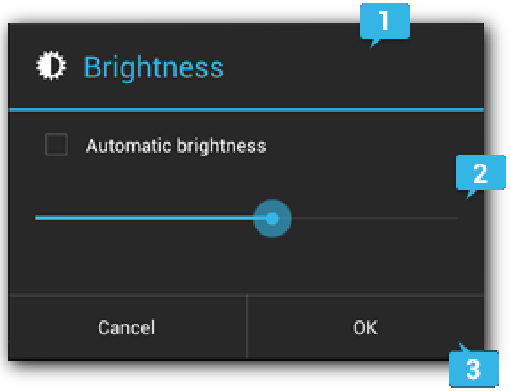
# Views – UI Widgets

Over 100 UI Controls :

- TextView, EditText, Button, CheckBox, RadioButton, ToggleButton, and Spinners for multiple options

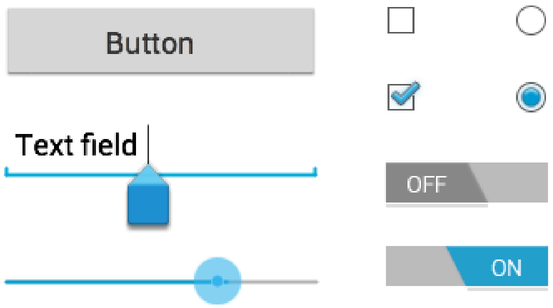


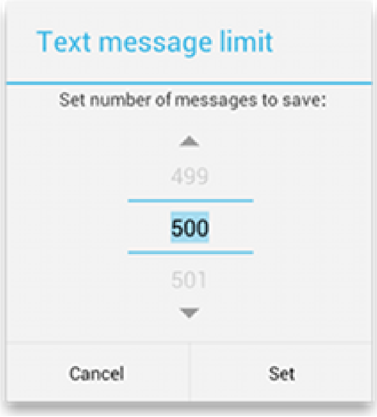
android.view
class hierarchy



Brightness

□ Automatic brightness





Text message limit

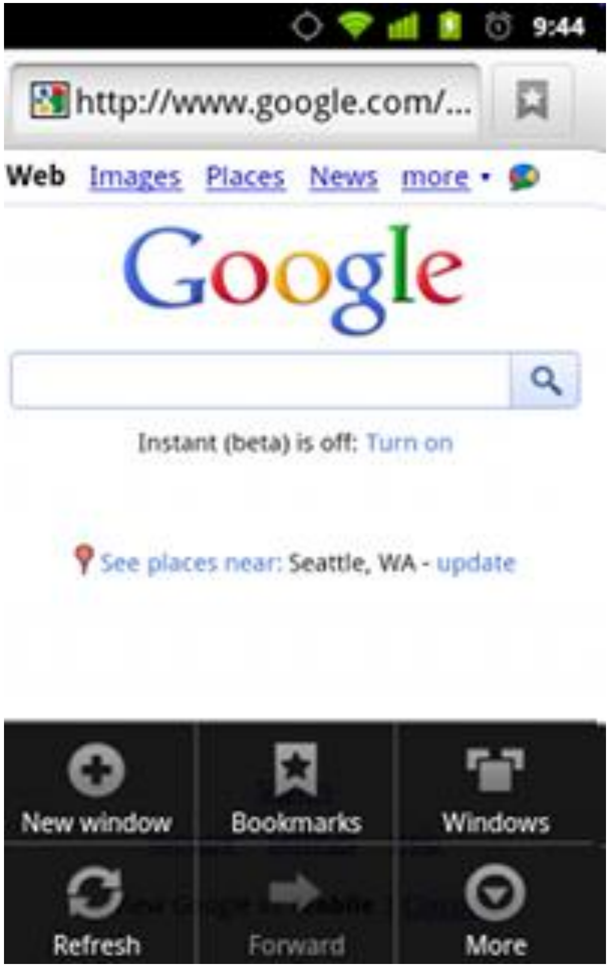
Set number of messages to save:

499
500
501

Revisited

# menu

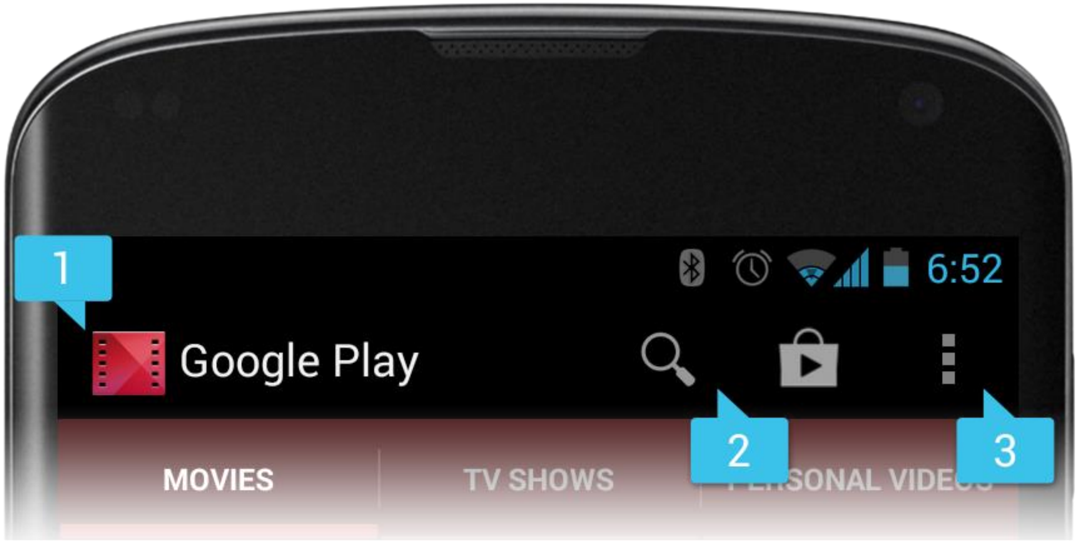
- Three menu:
  - **options menu**: main menu.
  - **context menu**: floating menu. By push on a view that registered for context menu more than 2 seconds.
  - **popup menu**: vertical list anchored to the view
- Define menu & its items in XML menu resources, then **inflate** it in code


9:44
http://www.google.com/...
Web
Images
Places
News
more
Google
Instant (beta) is off: Turn on
See places near: Seattle, WA - update
New window
Bookmarks
Windows
Refresh
Forward
More

Options Menu

# menu

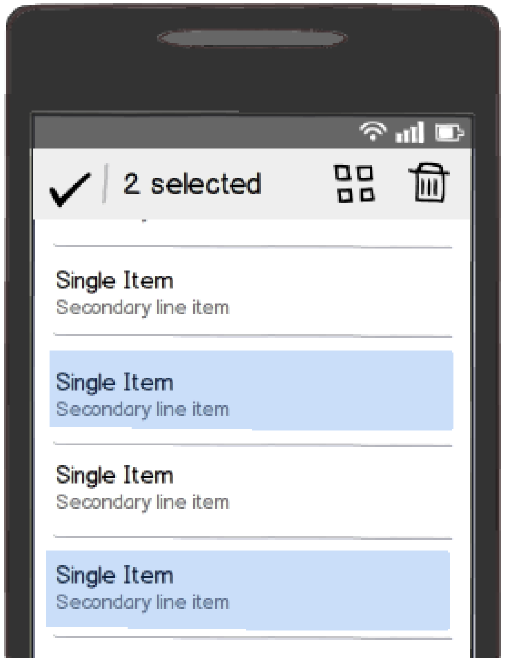
- Starting from 3.0 or above, action bar is recommended for user menu actions
- Provide a dedicated space for giving app an identity & user's location in app
- Makes important actions prominent and accessible



[1] app icon
[2] two action items
[3] action overflow

# Contextual Action Bar

- Content menu is a floating menu that appears when user presses an element
- This **Contextual action mode** will show a contextual action bar at top/bottom of screen to show actions the user can perform

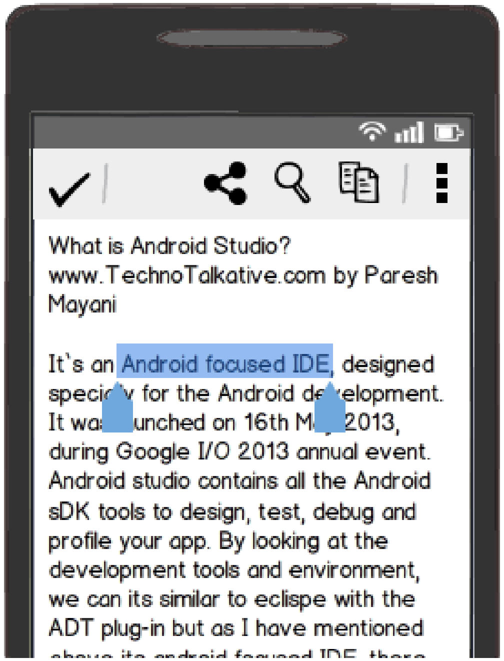


Single Item
Secondary line item

Single Item
Secondary line item

Single Item
Secondary line item

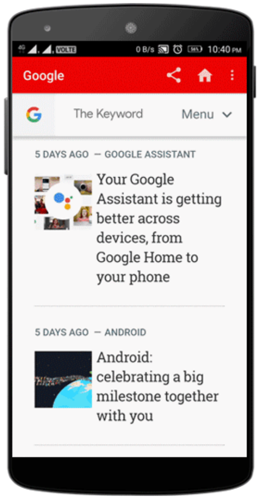
Single Item
Secondary line item


What is Android Studio?
www.TechnoTalkative.com by Paresh Mayani

It`s an Android focused IDE, designed specially for the Android development. It was launched on 16th May 2013, during Google I/O 2013 annual event. Android studio contains all the Android sDK tools to design, test, debug and profile your app. By looking at the development tools and environment, we can its similar to eclispe with the ADT plug-in but as I have mentioned above its android focused IDE, there

# Container Views

- Examples of Container Views:
  - WebView
  - SearchView
  - GridView
  - ListView
  - ScrollView
  - RecyclerView

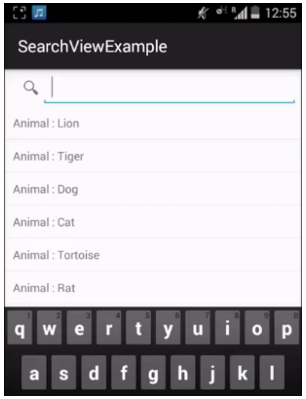


5 DAYS AGO — GOOGLE ASSISTANT

Your Google Assistant is getting better across devices, from Google Home to your phone

5 DAYS AGO — ANDROID

Android: celebrating a big milestone together with you



Animal : Lion

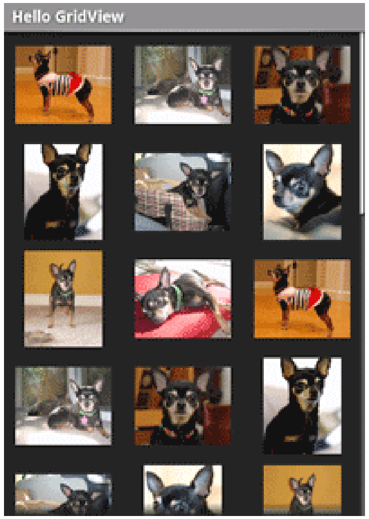
Animal : Tiger

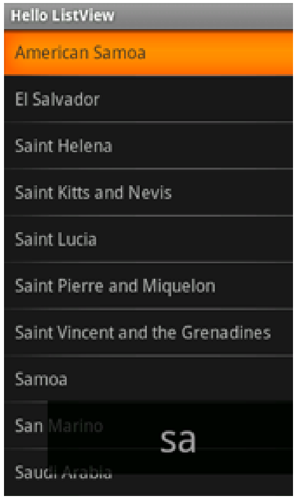
Animal : Dog

Animal : Cat

Animal : Tortoise

Animal : Rat





American Samoa

El Salvador

Saint Helena

Saint Kitts and Nevis

Saint Lucia

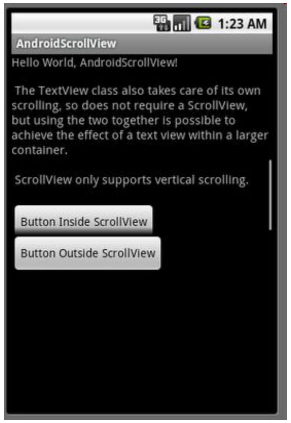
Saint Pierre and Miquelon

Saint Vincent and the Grenadines

Samoa

San Marino

Saudi Arabia



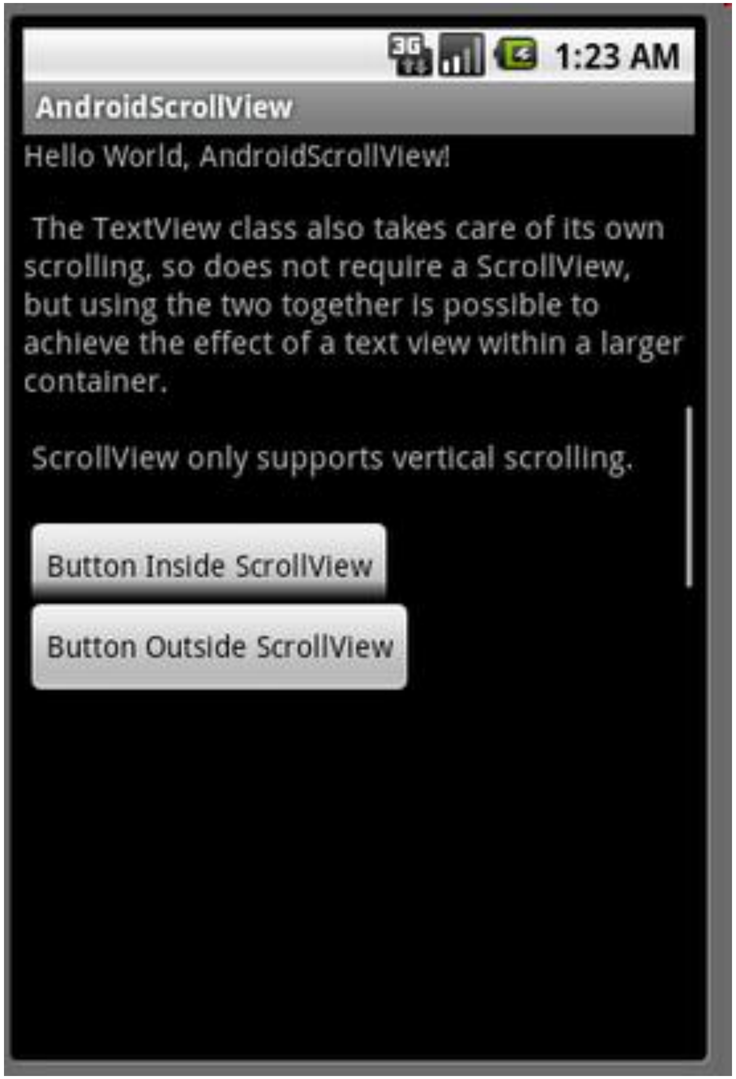
Hello World, AndroidScrollView!

The TextView class also takes care of its own scrolling, so does not require a ScrollView, but using the two together is possible to achieve the effect of a text view within a larger container.

ScrollView only supports vertical scrolling.

# ScrollView

- Not to house a ListView or RecyclerView within a ScrollView, because that defeats the performance optimizations of a ListView



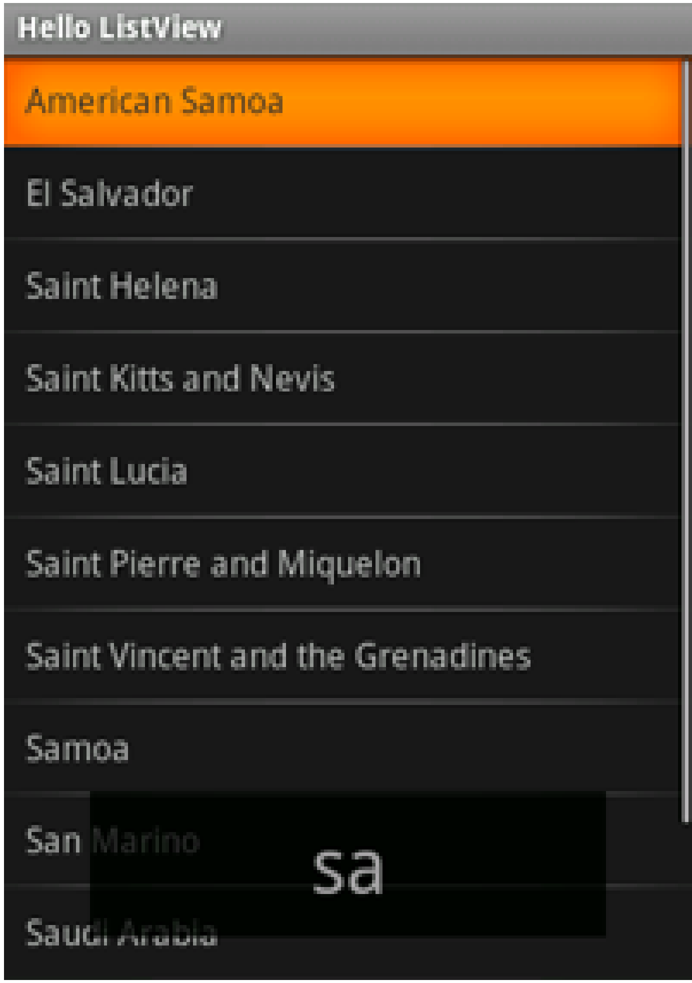
Hello World, AndroidScrollView!

The TextView class also takes care of its own scrolling, so does not require a ScrollView, but using the two together is possible to achieve the effect of a text view within a larger container.

ScrollView only supports vertical scrolling.

# ListView and ListActivity

- ListView is full screen
- Use AdapterView to bind the view to data source, thus retrieving data from source
- User interaction achieved through *ClickListener* member



Hello ListView

American Samoa

El Salvador

Saint Helena

Saint Kitts and Nevis

Saint Lucia

Saint Pierre and Miquelon

Saint Vincent and the Grenadines

Samoa

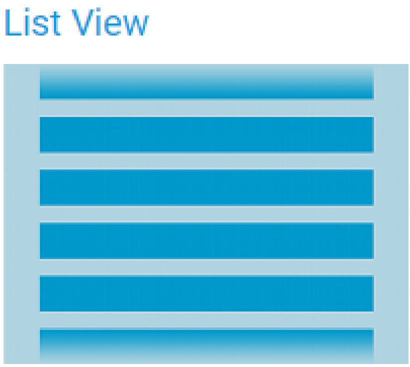
San Marino

Saudi Arabia

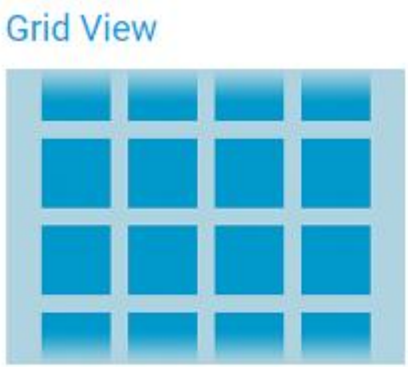
sa

# Views with an Adapter

When the content of the view is dynamic or not pre-determined, use AdapterView to generate dynamic layout.views at runtime, e.g.:



Displays a scrolling single column list.



Displays a scrolling grid of columns and rows.

Adapter is kind of *Design Pattern* in Software

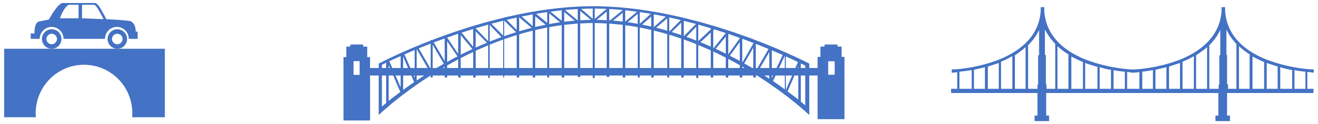
# Design Patterns

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

- a template for a design that solves a general, recurring problem in a particular context in software engineering

- problem is the goal you are trying to achieve

- abstracts the key aspects of the structure of a concrete design that has proven to be effective over time



# Design Patterns

- a kind of template or guide for a particular design

- design principles are **rules of thumb** for constructing **object-oriented** systems, such as
  - "encapsulate the aspects of system structure that vary"

- if we isolate the parts of a system that vary, and **encapsulate** them, they can **vary independently** of other parts of the system

# Design Patterns

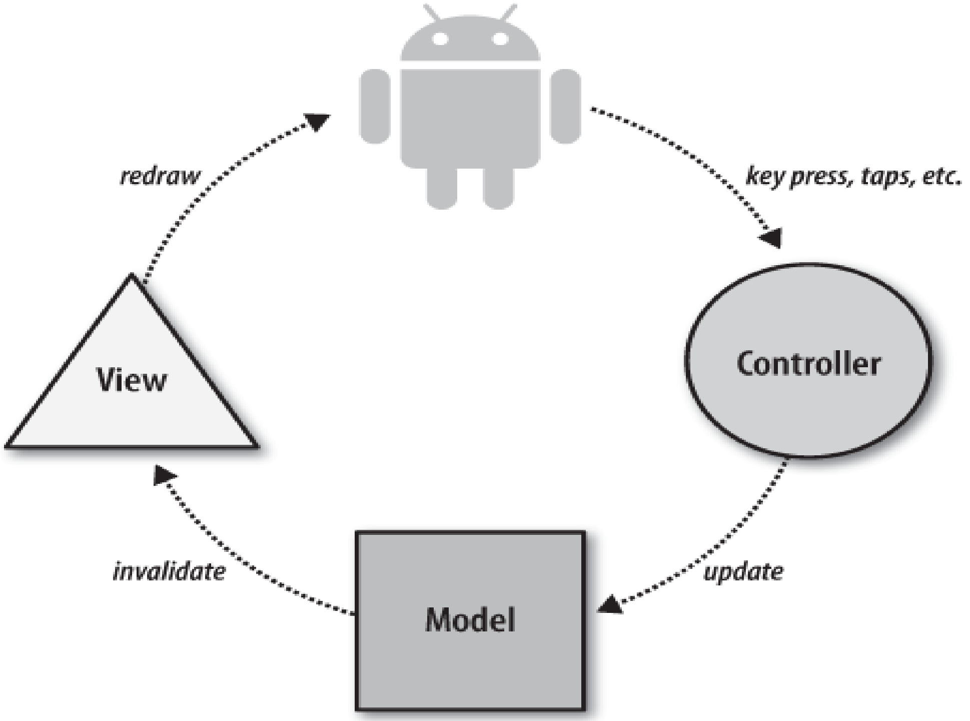
- Define interfaces for them that are not tied to implementation specifics
  - can later **alter or extend those variable parts without affecting the other** parts of the system

- **Reduce couplings** between parts, and consequently the system becomes more flexible and easier to change

- The important thing is to be aware of patterns when you are developing software and to use them in your designs

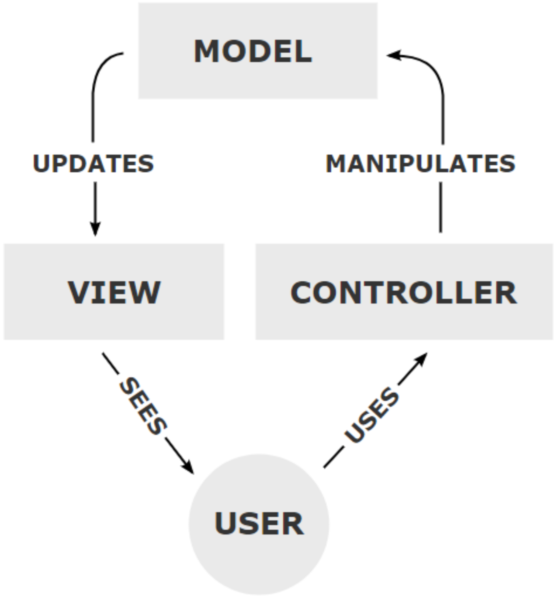
# MVC

- Android UI framework is organized around the common *Model-View-Controller* **design pattern**.



# Model-View-Controller

- Three types of objects:
  - model objects,
  - view objects,
  - controller objects



- designing an application, is choosin custom classes for—objects that fall into one of these three groups.

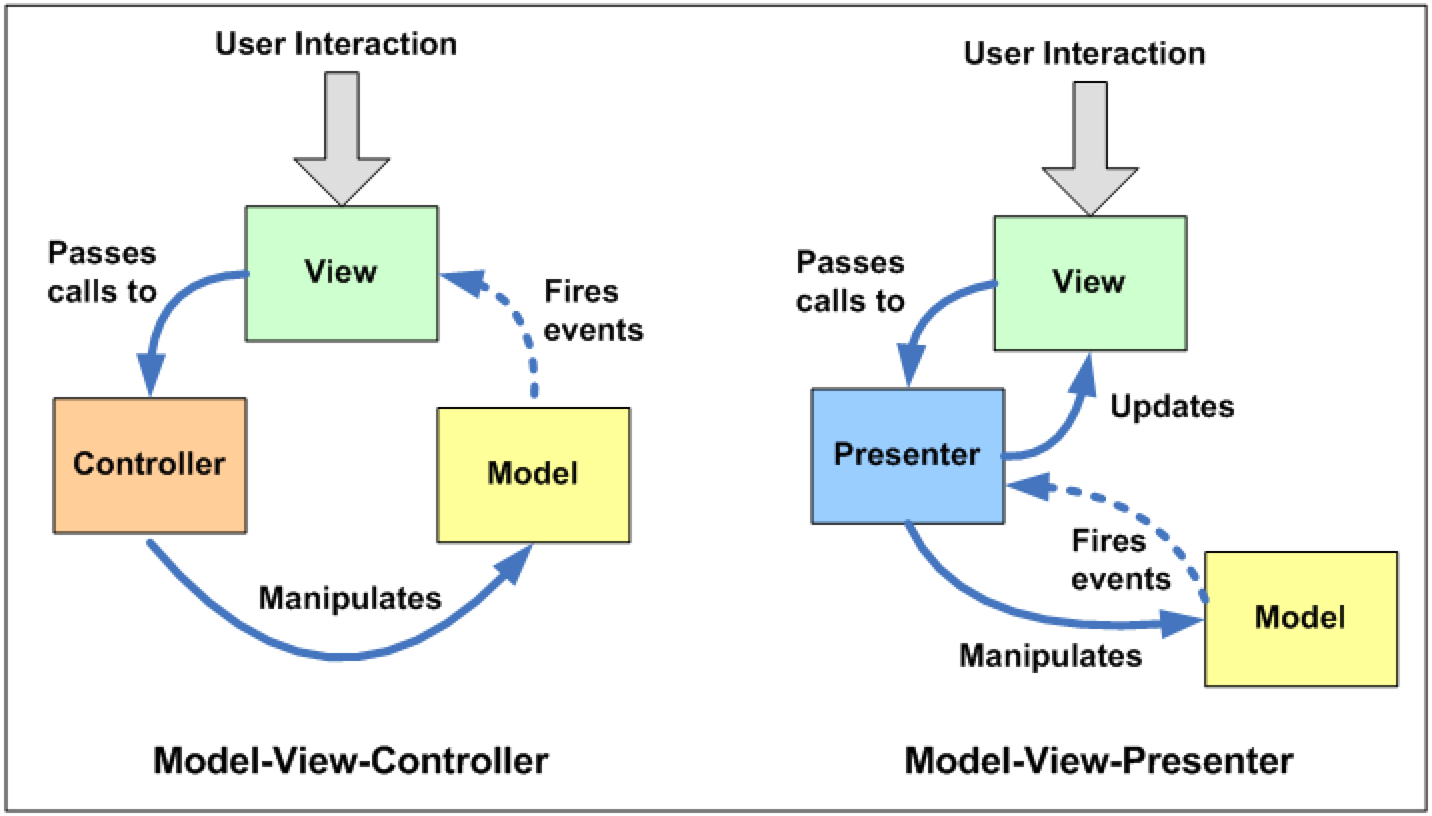
# Model-View-Controller

concerns with the **global architecture** of an application

objects in these programs tend to be more reusable and their **interfaces tend to be better defined**

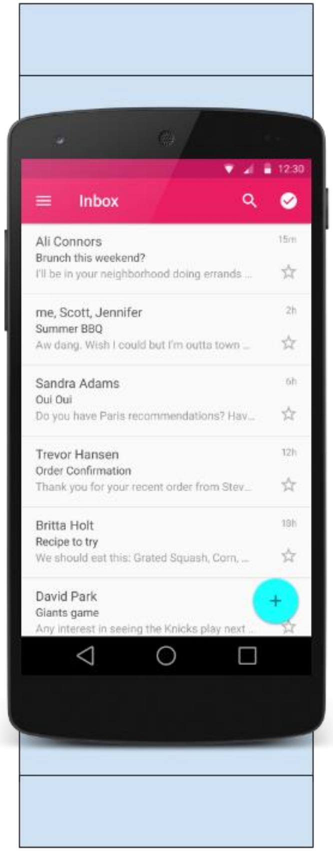
programs overall are **more adaptable to changing requirements**

# Or MVP? MVVM?



Stack Overflow: MVC vs MVP

# RecyclerView



Ali Connors 15m
Brunch this weekend?
I'll be in your neighborhood doing errands ...

me, Scott, Jennifer 2h
Summer BBQ
Aw dang. Wish I could but I'm outta town ...

Sandra Adams 6h
Oui Oui
Do you have Paris recommendations? Hav...

Trevor Hansen 12h
Order Confirmation
Thank you for your recent order from Stev...

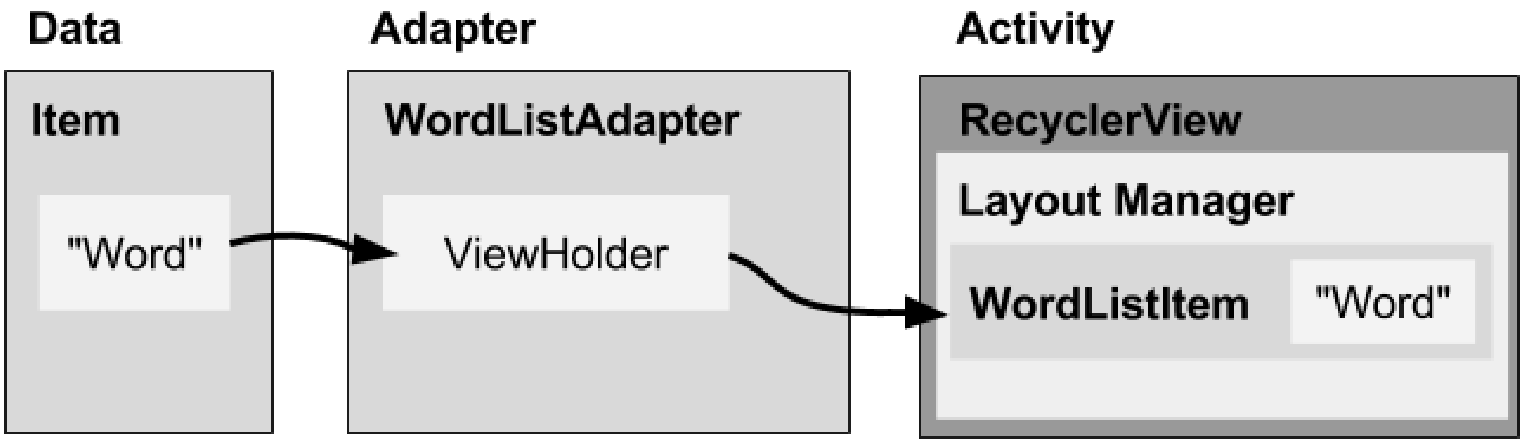
Britta Holt 18h
Recipe to try
We should eat this: Grated Squash, Corn, ...

David Park
Giants game
Any interest in seeing the Knicks play next ...

- Scrollable container for large data sets
- Efficient
  - Uses and reuses limited number of View elements
  - Updates changing data fast

# RecyclerView components

- **RecyclerView** scrolling list for list items
- **Adapter** connects data to the RecyclerView
- **ViewHolder** has view information for displaying one item

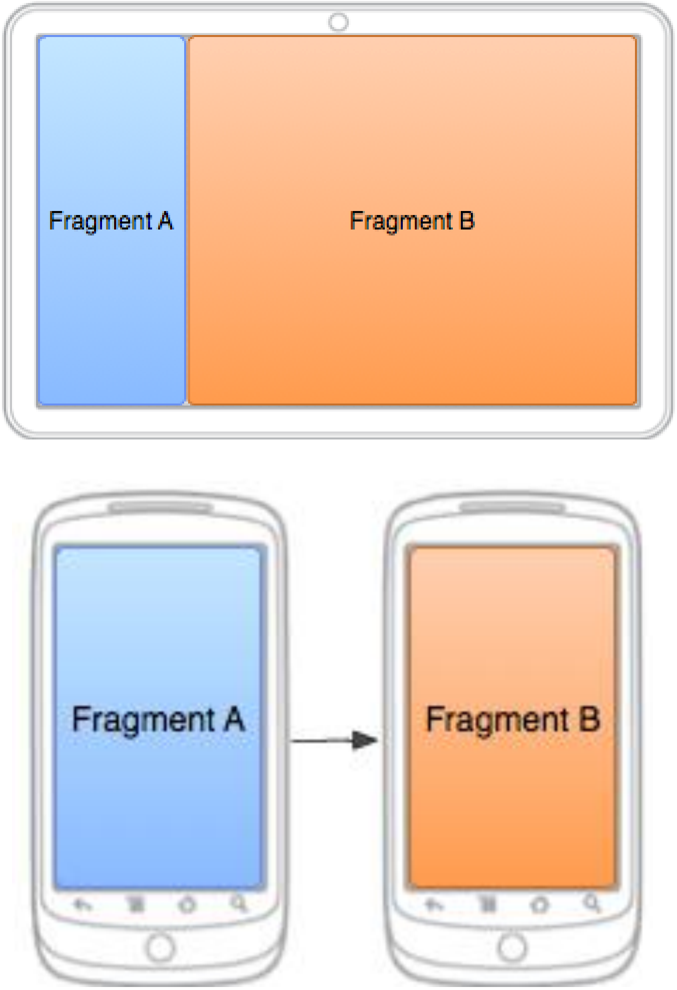


A topic to be discussed later.

# Fragment

- Behaves like a nested activity that can define its own layout and manage **its own lifecycle**
  - Fragment can be added to an Activity at Runtime
  - the fragment must have a container View in the layout
- Fragment-to-Fragment communication is done through the associated Activity - two Fragments should never communicate directly

A topic to be discussed later.



# Reference

- https://developer.android.com/guide/topics/ui/declaring-layout.html
- https://developer.android.com/reference/android/view/LayoutInflater.html
- https://developer.android.com/guide/topics/ui/how-android-draws

- https://stackoverflow.com/questions/45347761/android-view-system