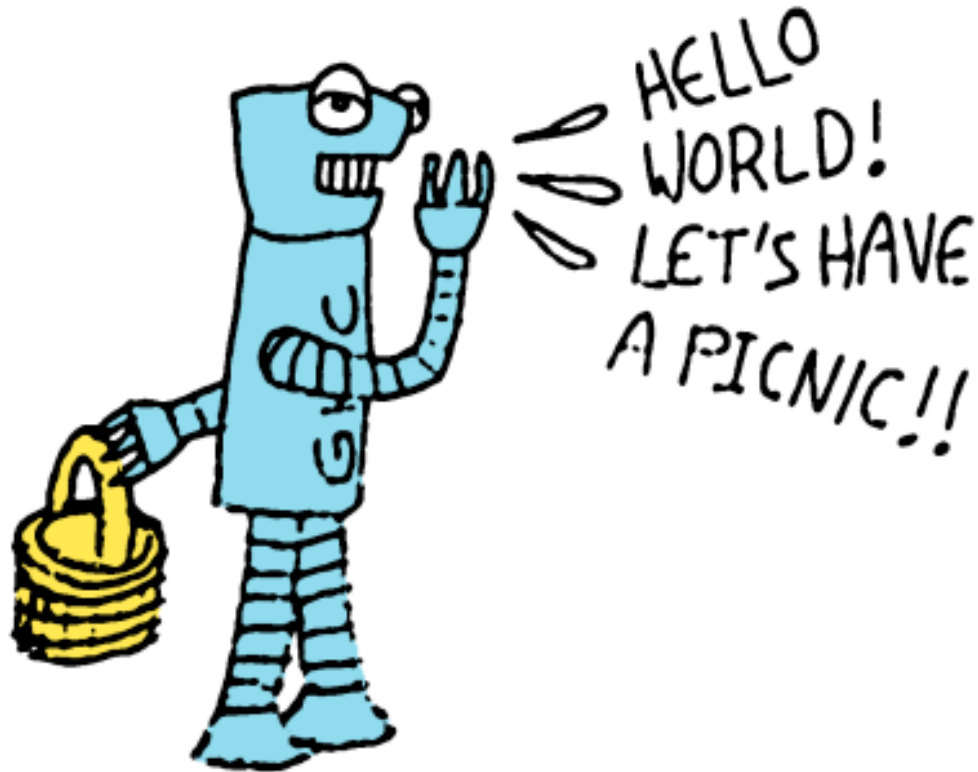




Android User Interface

Marko Gargenta
Marakana



HELLO WORLD!

Create New Project

Use the Eclipse tool to create a new Android project.

Here are some key constructs:

Project	Eclipse construct
Target	minimum to run
App name	whatever
Package	Java package
Activity	Java class

New Android Project
Creates a new Android Project resource.

Project name: HelloAndroid

Contents

- Create new project in workspace
- Create project from existing source
- Use default location

Location: /Users/marko/Workspace/Android/HelloAndroid

Create project from existing sample

Samples: ApiDemos

Build Target

<input type="checkbox"/>	Target Name	Vendor	Platform	AF
<input type="checkbox"/>	Android 1.1	Android Open Source Project	1.1	2
<input type="checkbox"/>	Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/>	Android 1.6	Android Open Source Project	1.6	4
<input checked="" type="checkbox"/>	Android 2.0	Android Open Source Project	2.0	5
<input type="checkbox"/>	Google APIs	Google Inc.	1.5	3
<input type="checkbox"/>	Google APIs	Google Inc.	1.6	4
<input type="checkbox"/>	Google APIs	Google Inc.	2.0	5

Standard Android platform 2.0

Properties

Application name: Hello, Android!!!

Package name: com.marakana

Create Activity: HelloAndroid

Min SDK Version: 5

The Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.marakana"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity android:name=".HelloAndroid"
      android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
  <uses-sdk android:minSdkVersion="5" />
</manifest>
```



The Layout Resource

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```



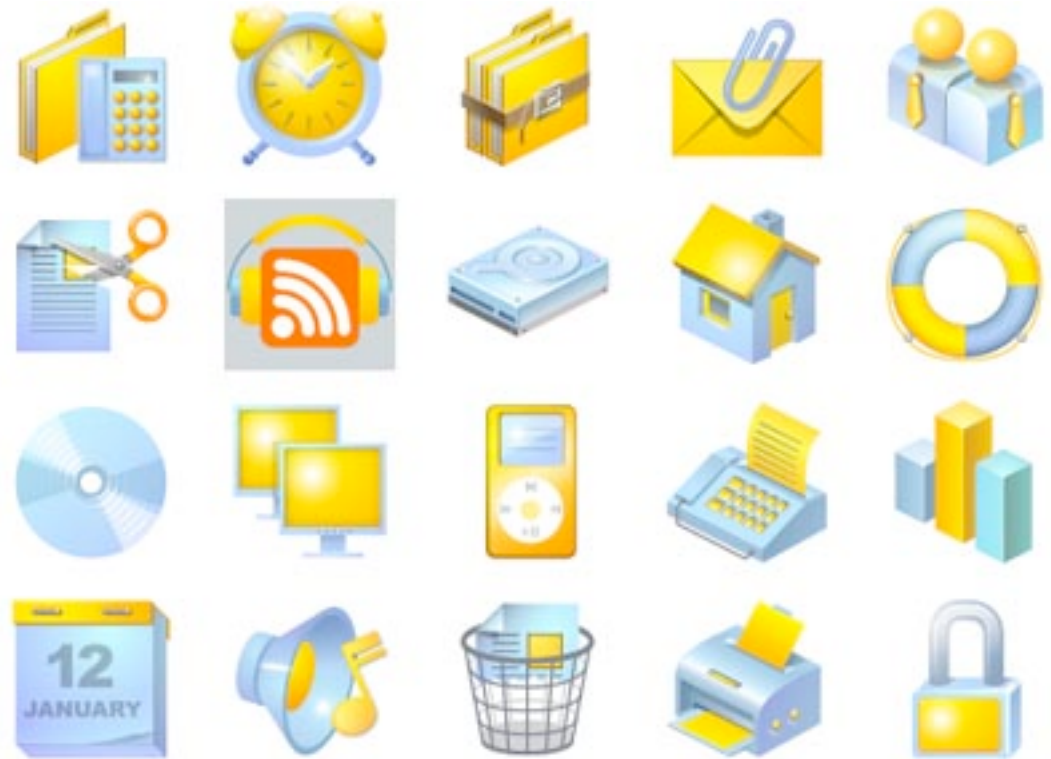
The Java File

```
package com.marakana;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```





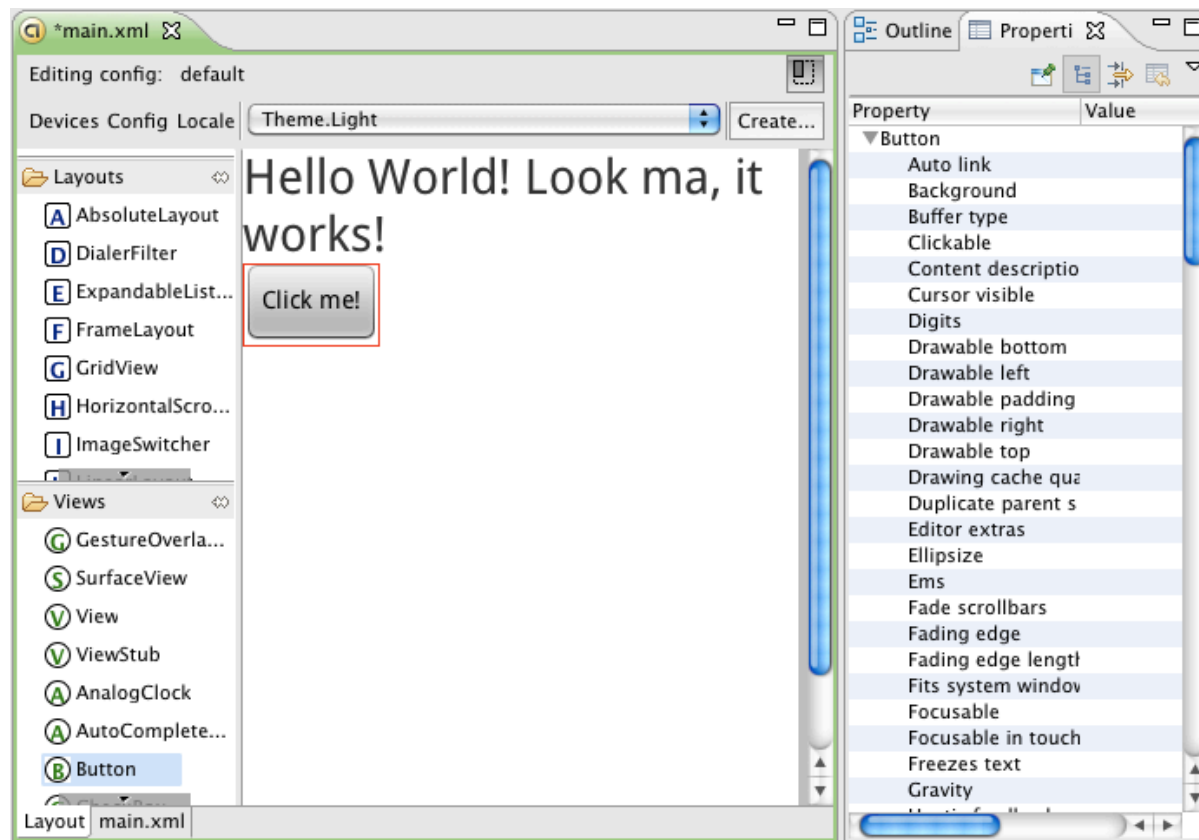
ANDROID USER INTERFACE

Two UI Approaches

Procedural	Declarative
You write Java code Similar to Swing or AWT	You write XML code Similar to HTML of a web page

You can mix and match both styles.
Declarative is preferred: easier and more tools

XML-Based User Interface

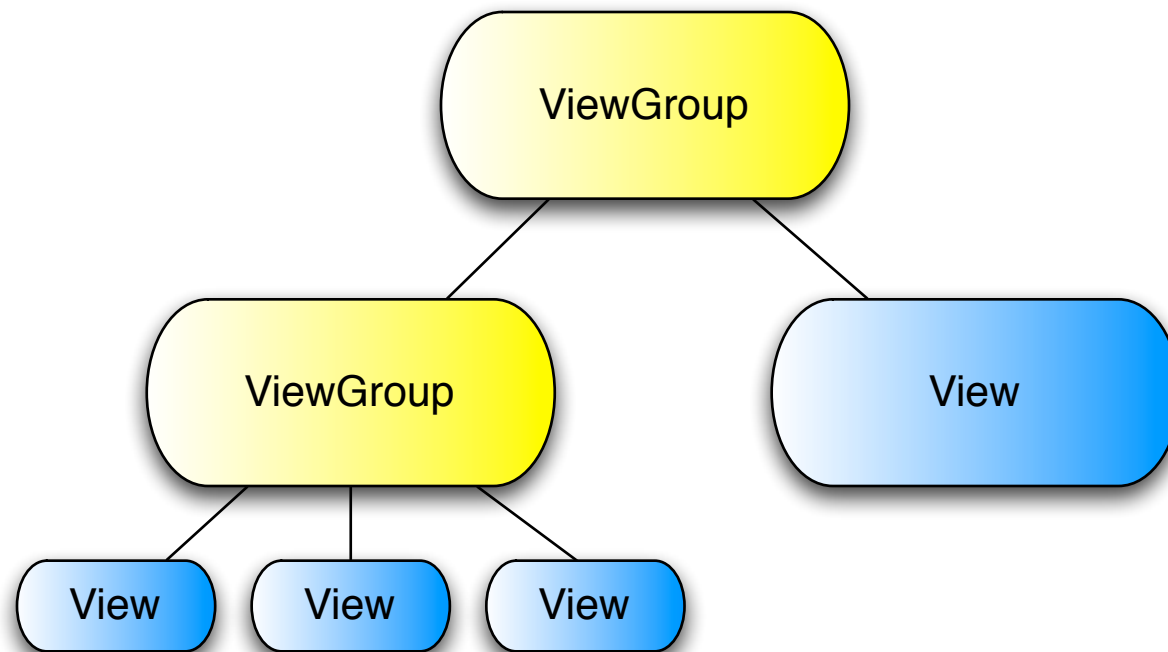


Use WYSIWYG tools to build powerful XML-based UI. Easily customize it from Java. Separate concerns.

Dips and Sps

px (pixel)	Dots on the screen
in (inches)	Size as measured by a ruler
mm (millimeters)	Size as measured by a ruler
pt (points)	1/72 of an inch
dp (density-independent pixel)	Abstract unit. On screen with 160dpi, 1dp=1px
dip	synonym for dp and often used by Google
sp	Similar to dp but also scaled by users font size preference

Views and Layouts



ViewGroups contain other Views but are also Views themselves.

Common UI Components

Android UI includes many common modern UI widgets, such as Buttons, Tabs, Progress Bars, Date and Time Pickers, etc.



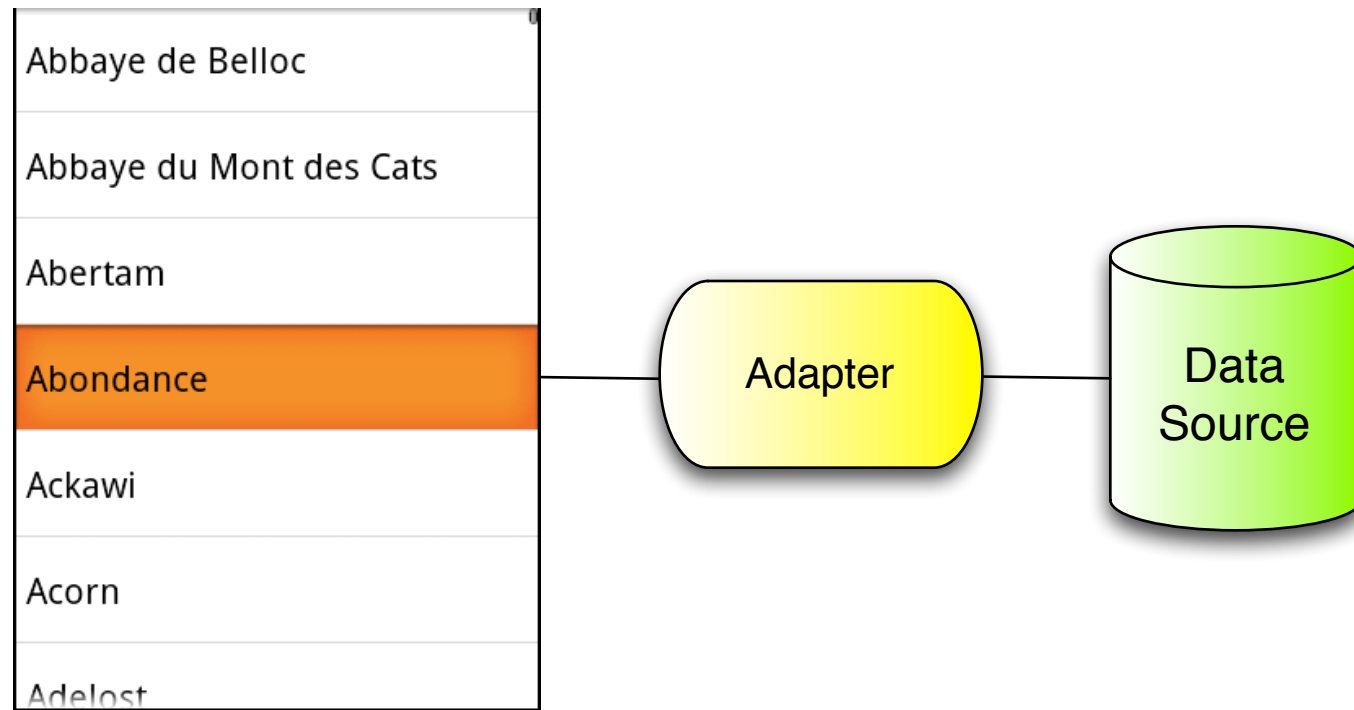
Selection Components

Some UI widgets may be linked to zillions of pieces of data.

Examples are ListView and Spinners (pull-downs).

Action	<input type="checkbox"/>
Adventure	<input type="checkbox"/>
Animation	<input type="checkbox"/>
Children	<input checked="" type="checkbox"/>
Comedy	<input checked="" type="checkbox"/>
Documentary	<input type="checkbox"/>
Drama	<input type="checkbox"/>

Adapters



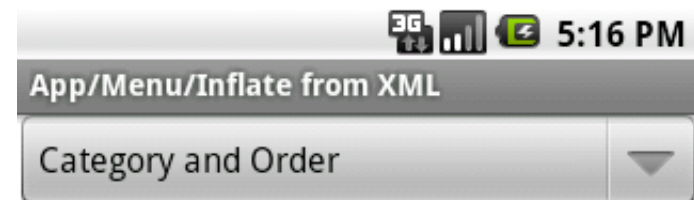
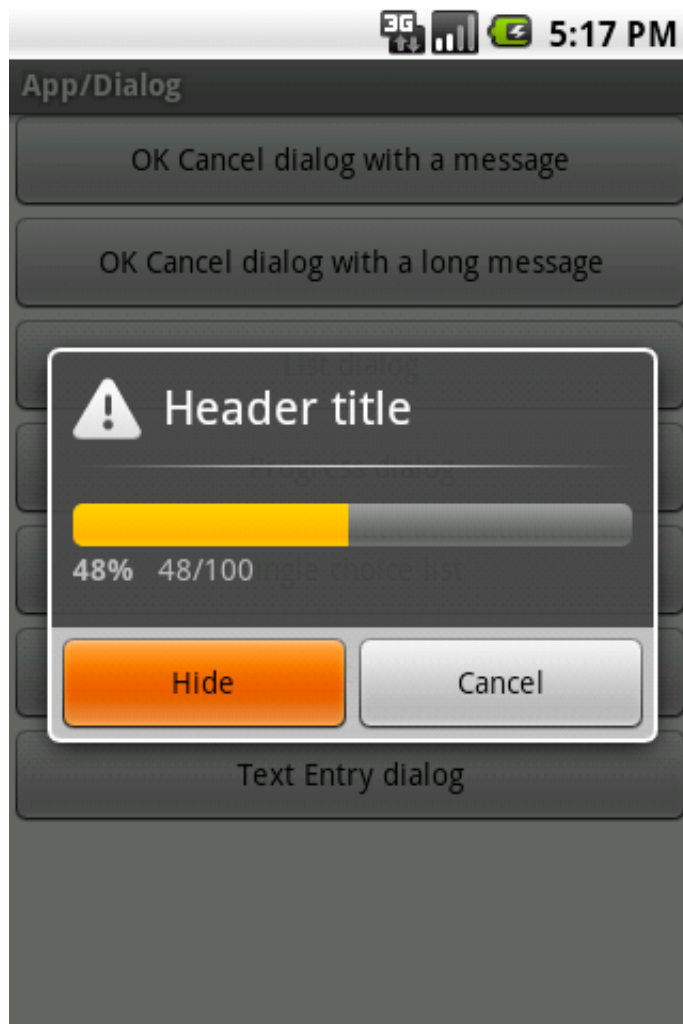
To make sure they run smoothly, Android uses Adapters to connect them to their data sources. A typical data source is an Array or a Database.

Complex Components

Certain high-level components are simply available just like Views. Adding a Map or a Video to your application is almost like adding a Button or a piece of text.



Menus and Dialogs



If you want to choose another menu resource, go back and re-run this activity.

First most often	Middle most often
Last most often	First least often
Middle least often	Last least often

Graphics & Animation

Android has rich support for 2D graphics.
 You can draw & animate from XML.
 You can use OpenGL for 3D graphics.

Left
 Center
 Right

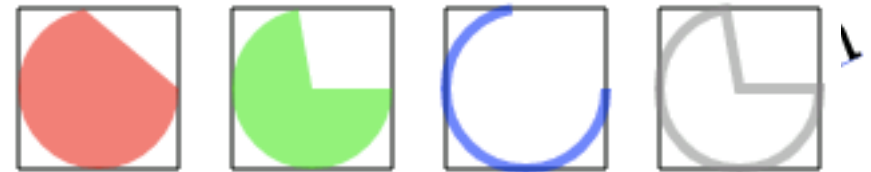
Positioned
 Positioned
 Positioned

Along a path
 Along a path



Default

Custom



Multimedia

AudioPlayer lets you simply specify the audio resource and play it.

VideoView is a View that you can drop anywhere in your activity, point to a video file and play it.

XML:

```
<VideoView
  android:id="@+id/video"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:layout_gravity="center" />
```

Java:

```
player = (VideoView) findViewById(R.id.video);
player.setVideoPath("/sdcard/samplevideo.3gp");
player.start();
```



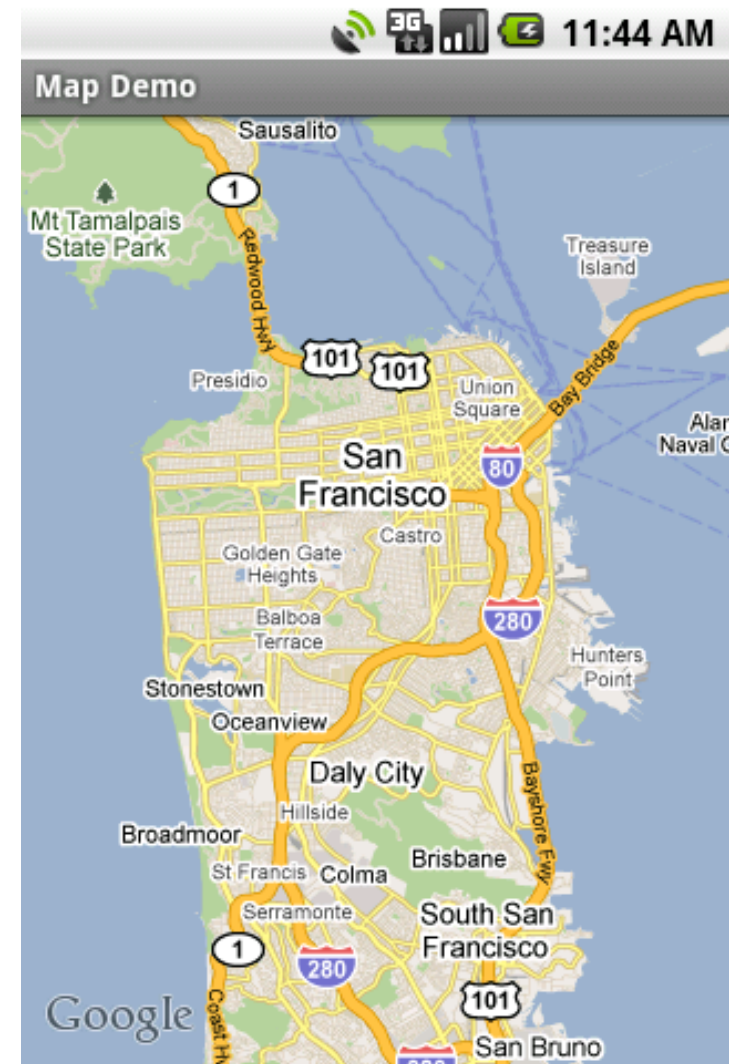
Google Maps

Google Maps is an add-on in Android.
It is not part of open-source project.

However, adding Maps is relatively
easy using **MapView**.

XML:

```
<com.google.android.maps.MapView  
android:id="@+id/map"  
android:clickable="true"  
android:layout_width="fill_parent"  
android:layout_height="fill_parent"  
android:apiKey="0EfLSgdSCWIN...A"  
>
```



Building UI for Performance

The screenshot displays the Hierarchy Viewer tool interface. The main window shows a hierarchical tree of UI components. The root node is 'PhoneWindowDecorView', which branches into 'LinearLayout', 'FrameLayout', 'ScrollView', and another 'LinearLayout'. The bottom 'LinearLayout' contains six 'Button' nodes. The right-hand panel shows performance metrics for operations and properties.

Operation	Duration (ms)
measure	2.927
layout	1.789
draw	37.076

Property	Value
----------	-------

At the bottom of the interface, there are controls for 'On White', 'On Black', and 'Show Extras'. A zoom slider is set to 20%, and the view shows 14 views.

A handy Hierarchy Viewer tool helps with optimizing the UI for performance

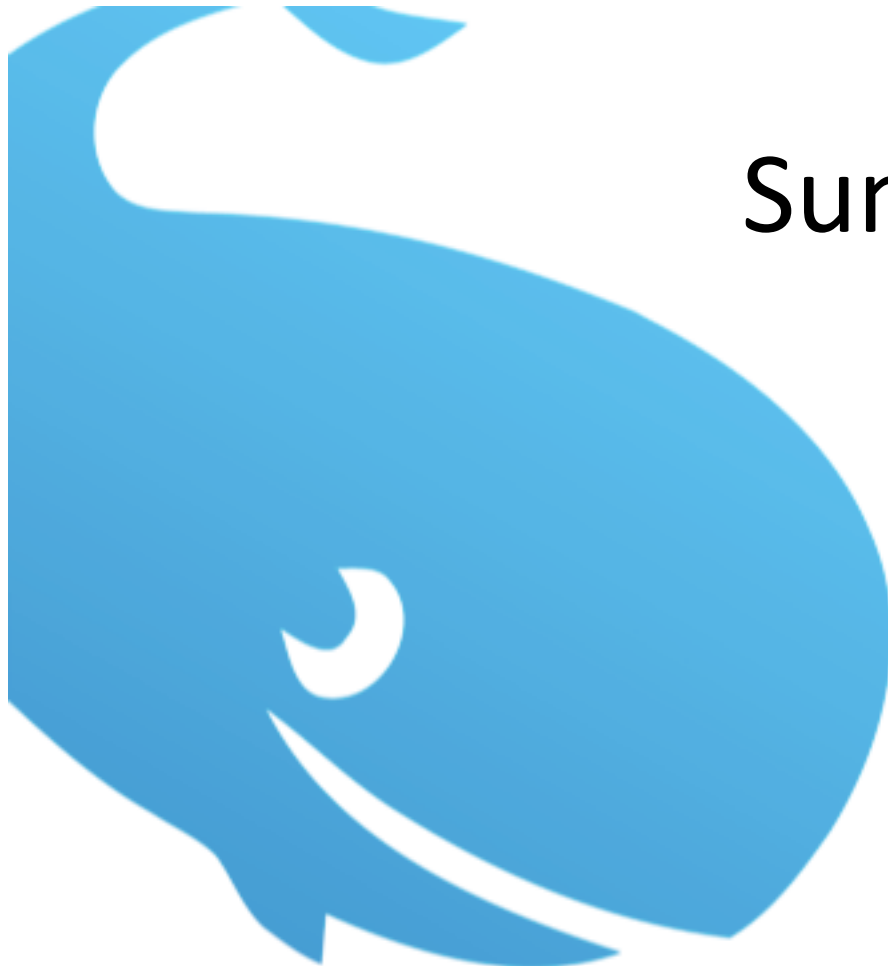
Summary

Main difference between Java and Android is User Interface.

Android has two approaches to UI: programmatic and declarative. Best practice is to use both.

Lifecycle of an activity is very important for overall performance of your app. So, optimize your UI.

Marko Gargenta, Marakana.com
marko@marakana.com
+1-415-647-7000



ANDROID

Licensed under Creative Commons License (cc-by-nc-nd). **Please Share!**

