



Android sweat shop

first meeting 23.6



Me - Serj Smorodinsky

- M.S.c Bioinformatics
- 3 Years of Android development
- Moto: practice first theory second
- Serjsmor@gmail.com
- Github: <https://github.com/serjSmor>
- Blog: <https://serjdroid.wordpress.com>





Sweat shop info

- I count that you are familiar with some OOP language
- Emphasis on coding first
- Project oriented - we are learning in order to accomplish a task
- Projects and slides will be available on Github
- It's a headstart, and not a complete Android course

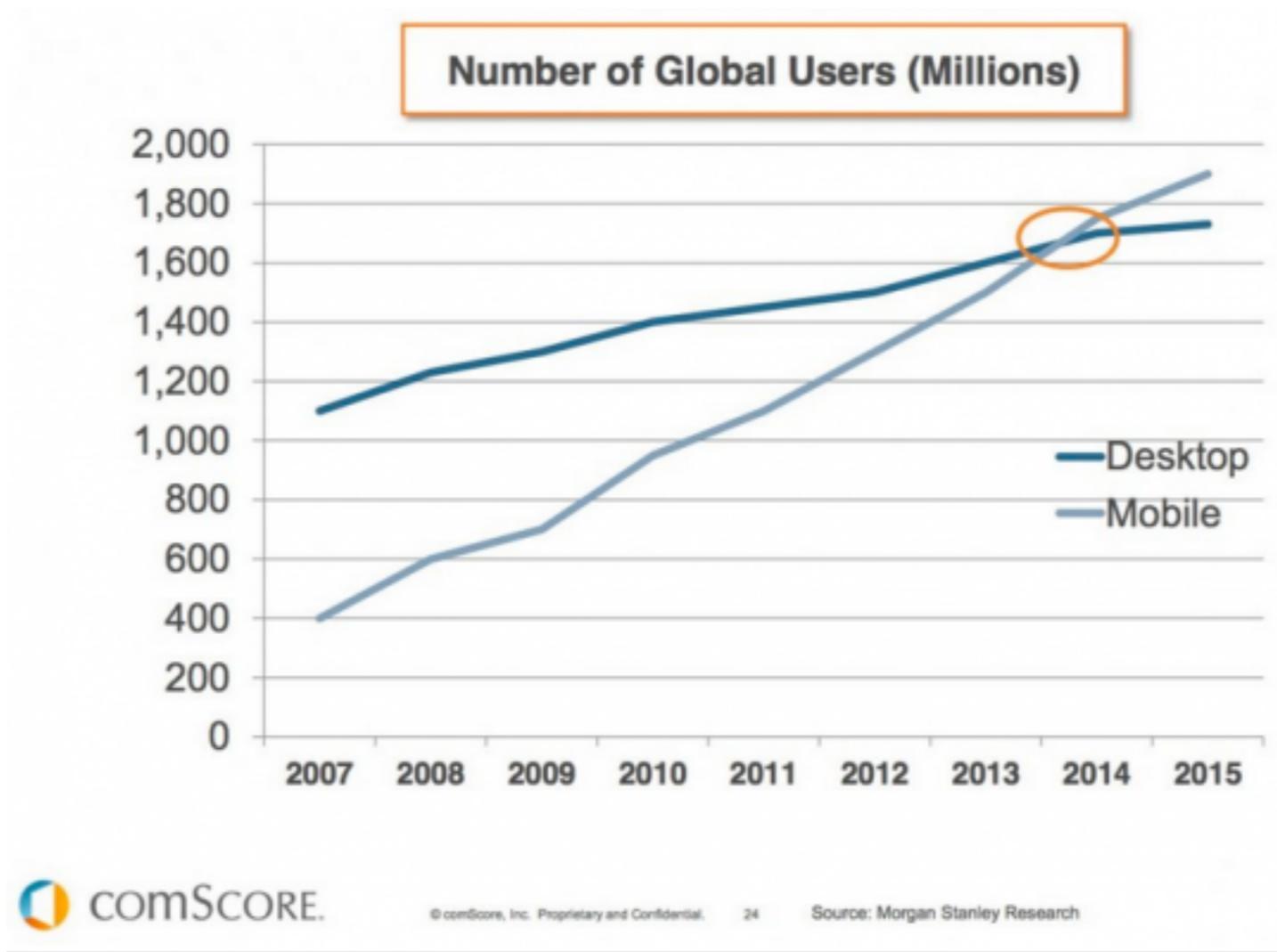


Table of contents

- **Why Android**
- **What is Android**
- **HelloWorld**
- **Activity**
 - Lifecycle
 - Logcat
 - Assignment
- **Views**
 - Layouts
 - Attributes
 - Assignment
- **Buttons**
 - OnClickListener
 - Toast
 - Assignment
- **Summary**

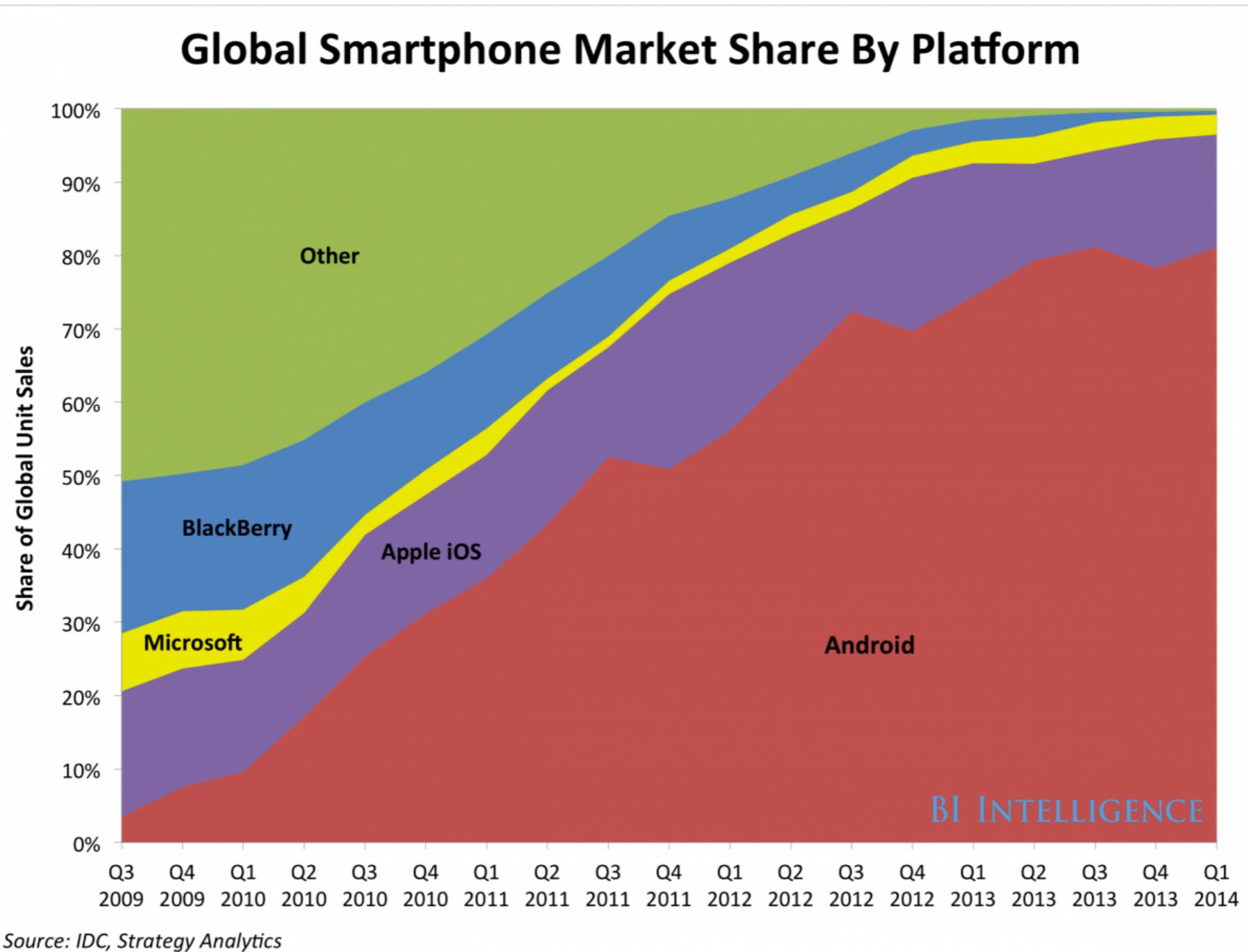


(Why) Desktop to mobile



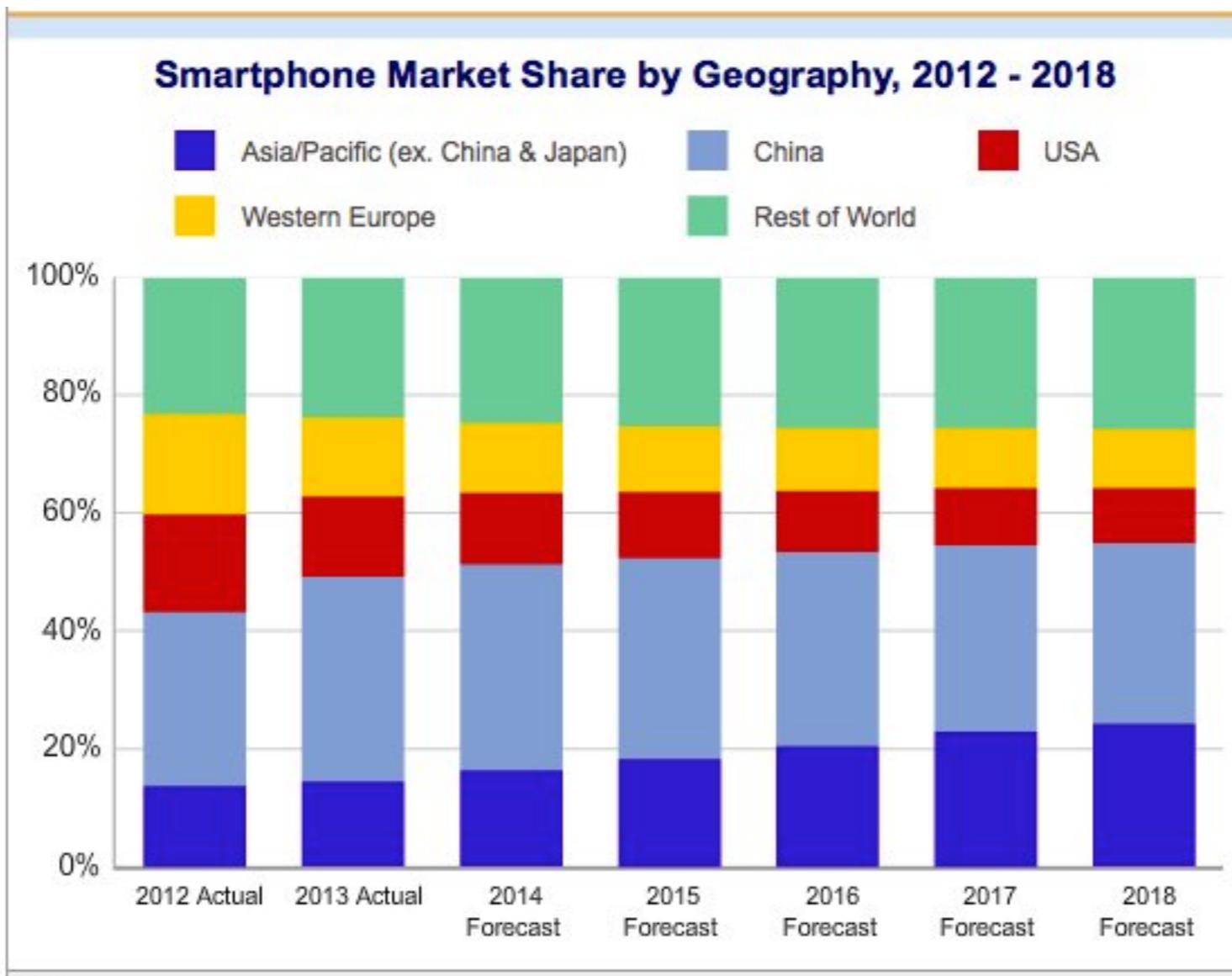


From iOS to Android





West to east





Future of platform





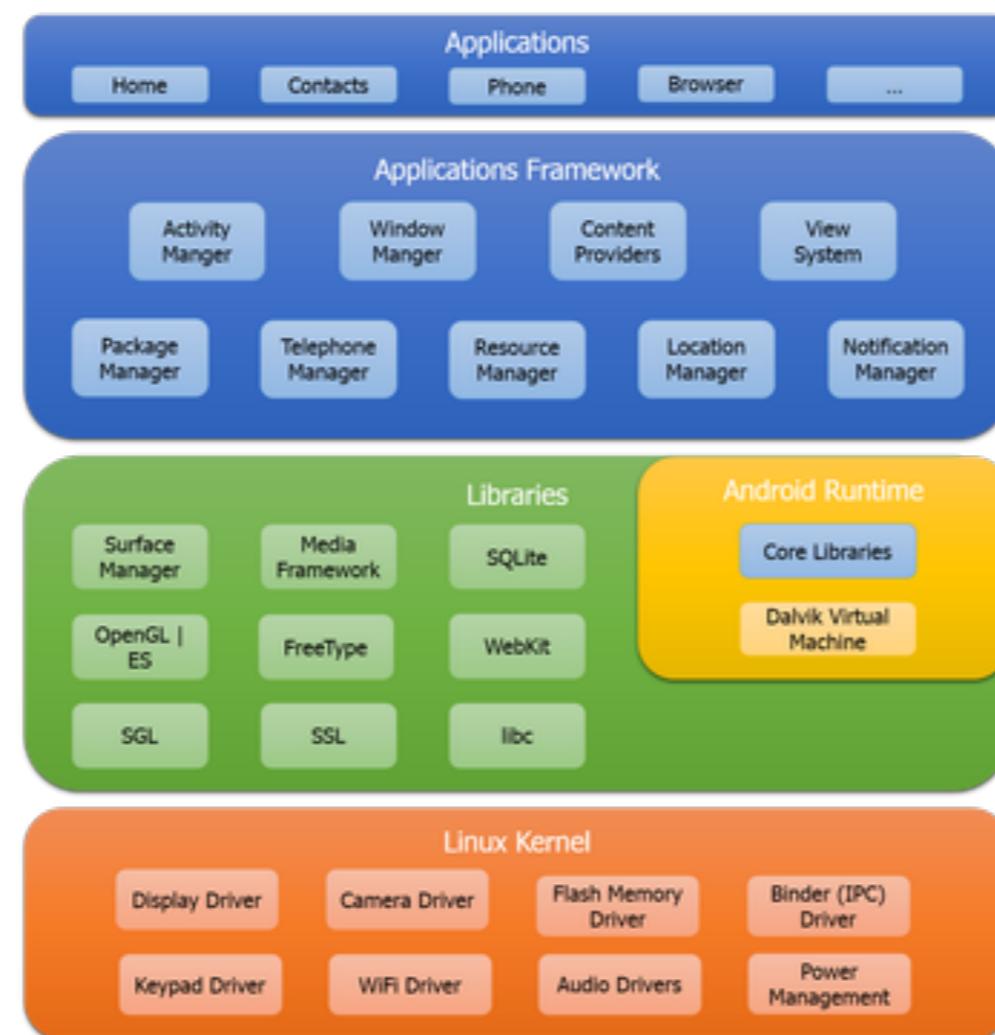
Trivia



From: <https://lokeshv.wordpress.com/tag/android/>



(What) Android architecture



From: <https://lokeshv.wordpress.com/tag/android/>



(What)

- The Android stack is basically a linux kernel
- Android has NDK - allows combining C++ code
- Lots of interprocess communication - Binder.
Example - GPS service



HelloWorld - create new project

Welcome to Android Studio



The screenshot shows the Android Studio interface. On the left, there is a sidebar titled "HelloWorld" containing a list of projects:

- HelloWorld
~/AndroidWorkshop/HelloWorld
- brickbreaker
~/brickbreaker
- Picok
~/picok
- SeekBarExample
~/AndroidStudio...ce/SeekBarExample
- NinePatchExample
~/AndroidStudio.../NinePatchExample
- DigitalPheromone
~/mapmap/client/DigitalPheromone
- client
~/mapmap/client
- SpaceInvaders
~/Nasi Gaming Pr...cts/SpaceInvaders
- GameOfLife
~/NasiProjects/GameOfLife
- ~/Nasi Gaming Projects/GameOfLife
~/Nasi Gaming Projects/GameOfLife

The main area displays the Android Studio logo and version information:

Android Studio
Version 2.1.2

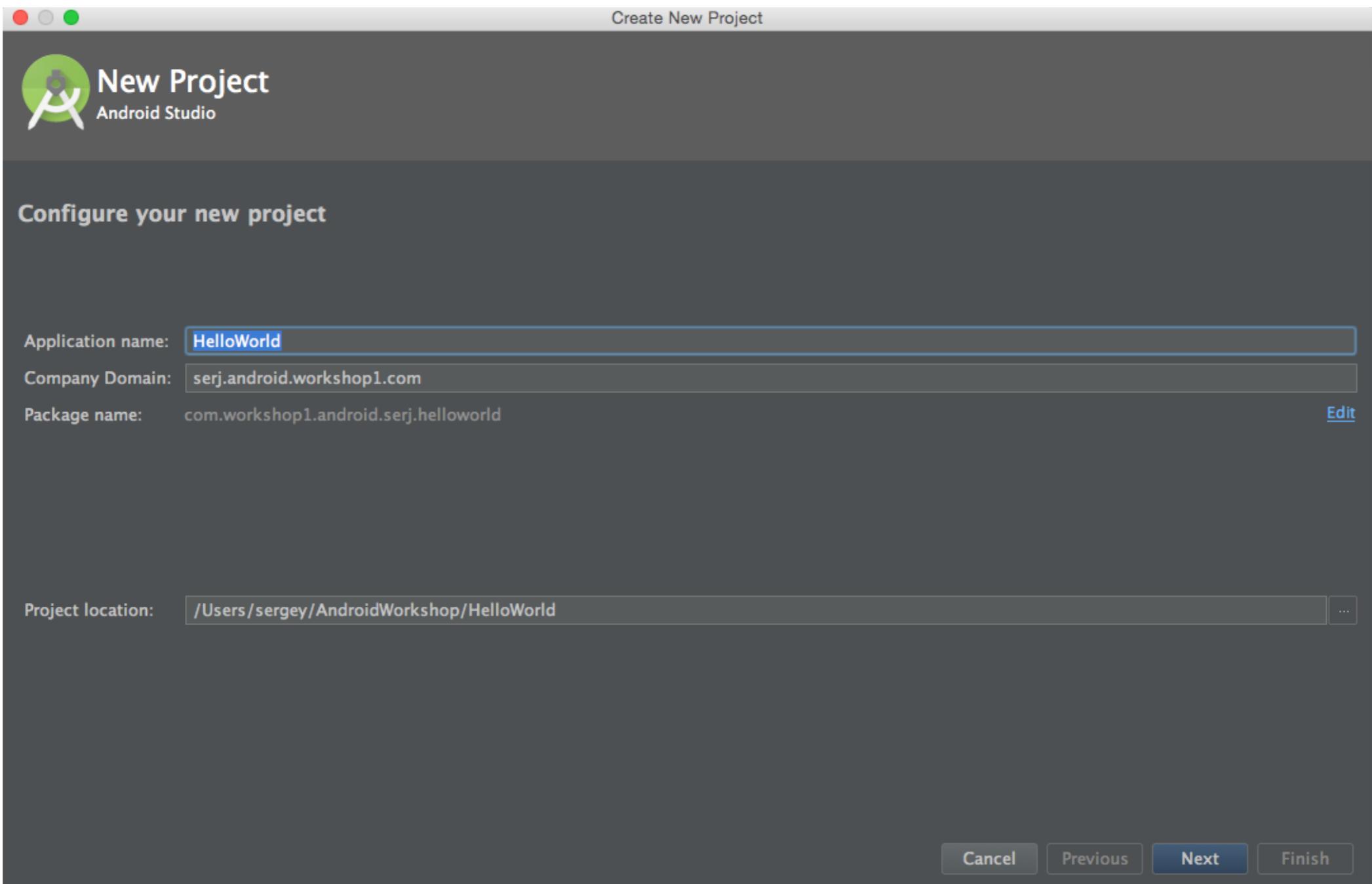
With the following options:

- ★ Start a new Android Studio project
- 📂 Open an existing Android Studio project
- ⬇ Check out project from Version Control ▾
- ⚡ Import project (Eclipse ADT, Gradle, etc.)
- ⚡ Import an Android code sample

At the bottom right are "Configure ▾" and "Get Help ▾" buttons.



HelloWorld - name it





HelloWorld - targets

Create New Project

 Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

Phone and Tablet
Minimum SDK API 15: Android 4.0.3 (IceCreamSandwich) ▾
Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately 97.4% of the devices that are active on the Google Play Store.
[Help me choose](#)

Wear
Minimum SDK API 21: Android 5.0 (Lollipop) ▾

TV
Minimum SDK API 21: Android 5.0 (Lollipop) ▾

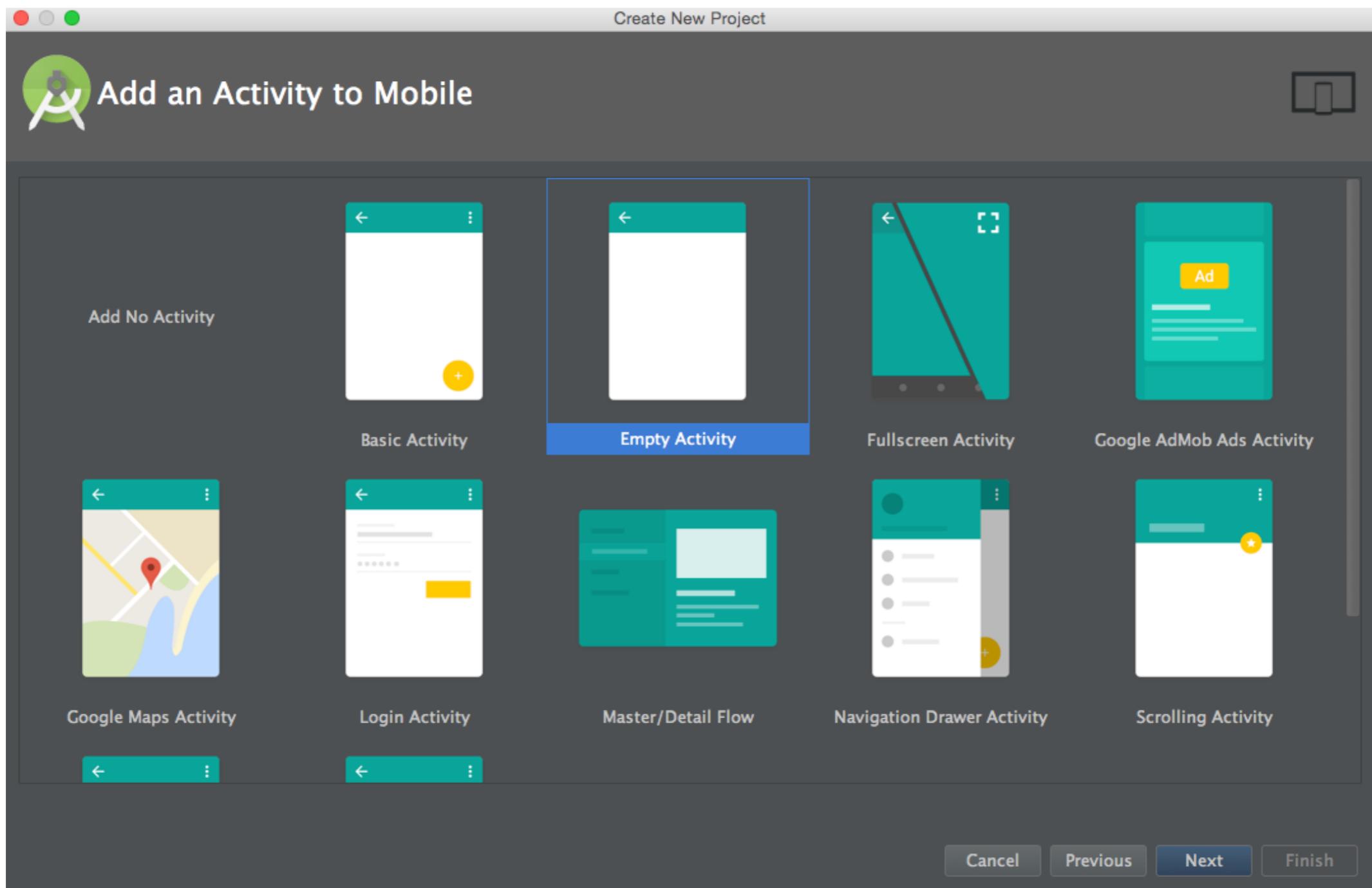
Android Auto

Glass
Minimum SDK Glass Development Kit Preview (API 19) ▾

Cancel Previous Next Finish

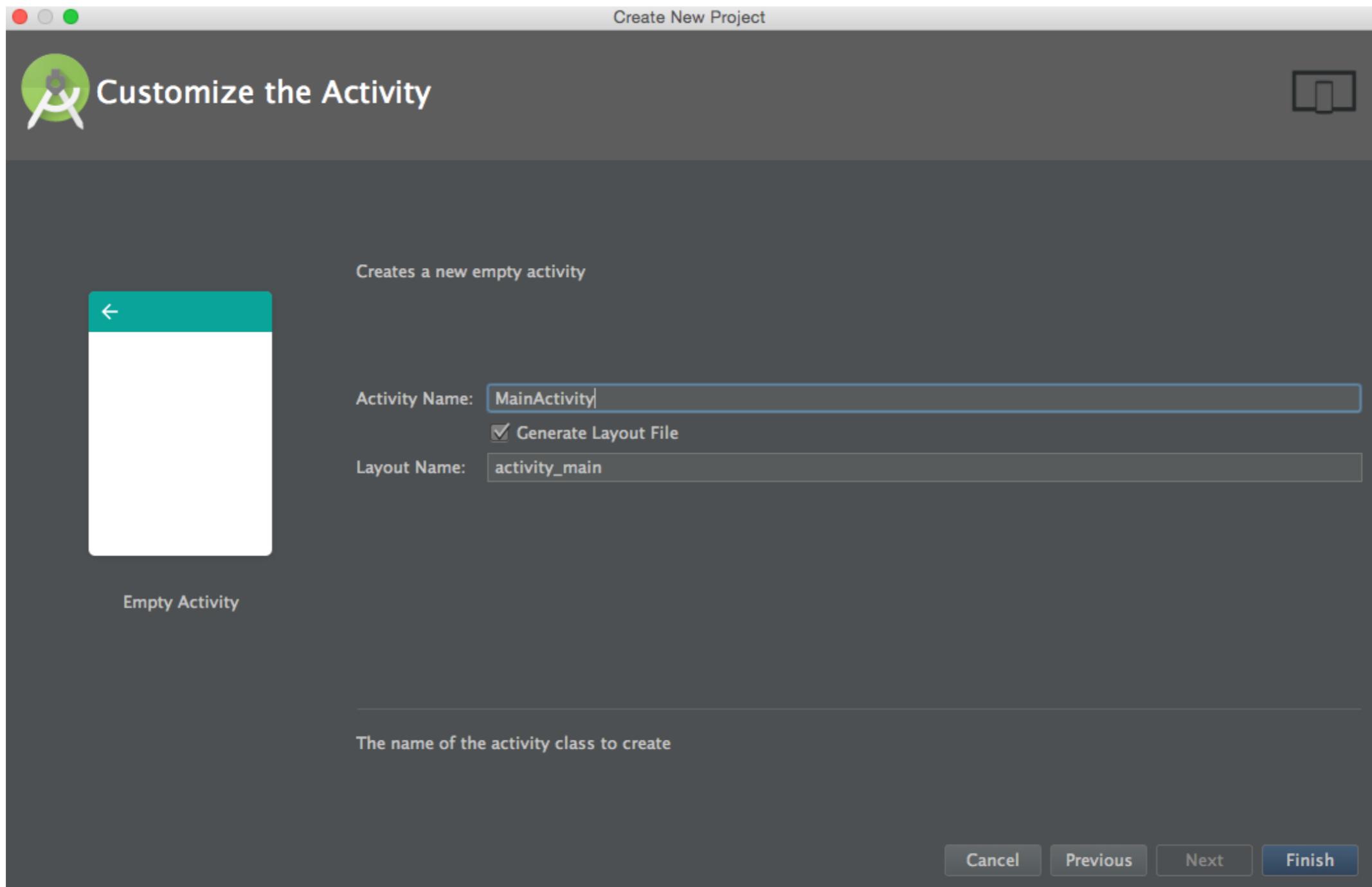


HelloWorld - templates





HelloWorld - name activity





HelloWorld - MainActivity

The screenshot shows the Android Studio interface with the following details:

- Toolbar:** Standard Android Studio menu items: File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Title Bar:** MainActivity.java - HelloWorld - [~/AndroidWorkshop/HelloWorld]
- Project Structure:** Shows the project tree under "app":
 - Java: com.workshop1.android.serj.helloworld (MainActivity)
 - res: drawable, layout (activity_main.xml), mipmap, values
- Code Editor:** Displays the MainActivity.java code:

```
1 package com.workshop1.android.serj.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12 }
```
- Bottom Bar:** Includes tabs for Terminal, Android Monitor, Messages, TODO, Event Log, Gradle Console, and a status bar showing "Gradle build finished in 12s 412ms", "6:14", "LF+", "UTF-8", "Context: <no context>", and battery level.



HelloWorld - activity_main preview

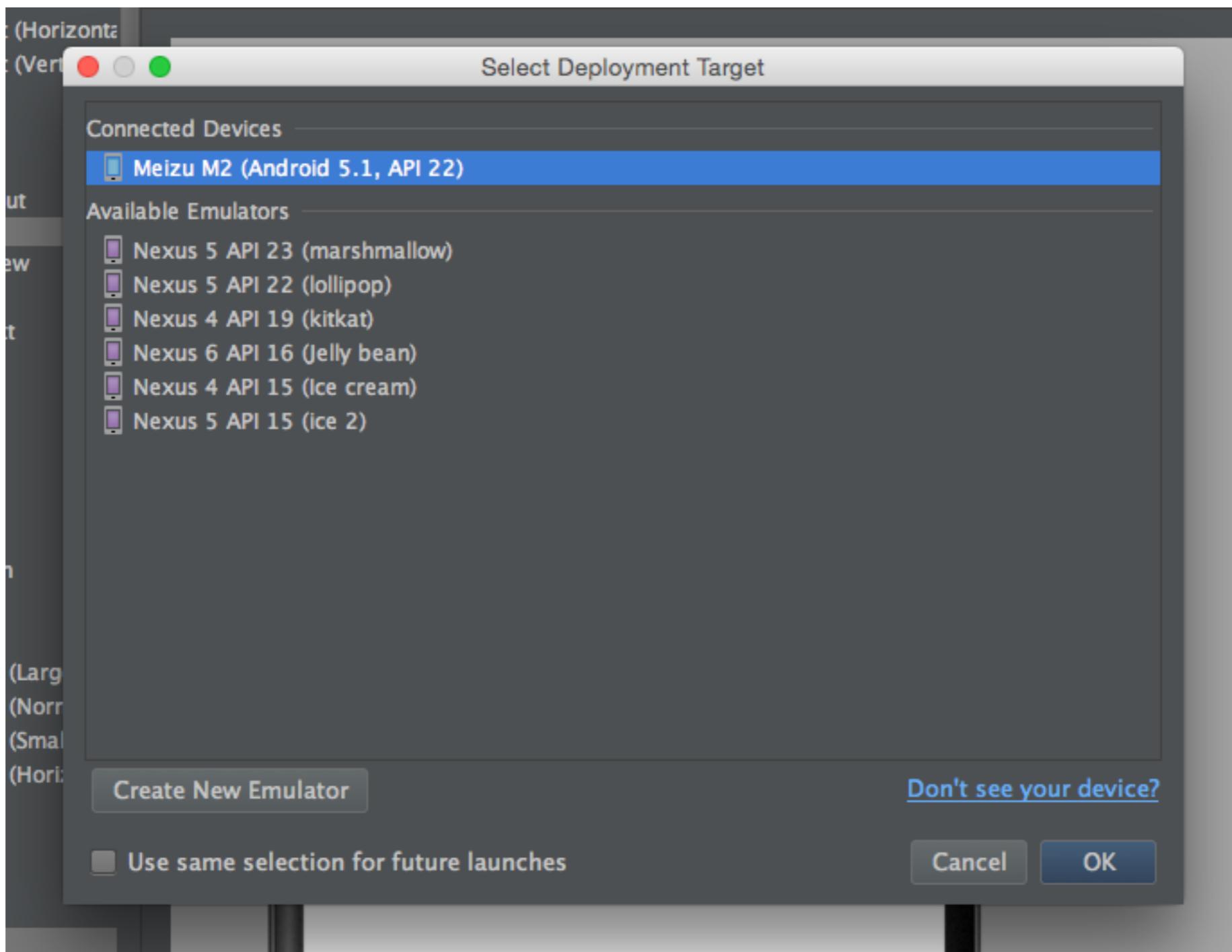
The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "HelloWorld". The "app" module contains "src/main/res/layout/activity_main.xml".
- Toolbars and Menus:** Standard Android Studio menus like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help are visible.
- Preview Area:** A smartphone-shaped preview window displays the layout. The screen shows a blue header bar with the title "HelloWorld" and a white content area with the text "Hello World!".
- Palette:** The left sidebar includes sections for "Android", "Layouts", "Widgets", "Text Fields", "Containers", and "Text".
- Component Tree:** Shows the hierarchy of the layout, starting with a "RelativeLayout" containing a "TextView" with the text "Hello World!".
- Properties:** A table showing properties for the selected "TextView":

layout_width	match_parent
layout_height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalOrder	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	
contentDescription	
contextClickable	
elevation	
focusable	
focusableInTouchMode	
foreground	
foregroundGravity	
foregroundTint	
foregroundTintMode	
gravity	
id	
ignoreGravity	
- Bottom Bar:** Includes tabs for Terminal, Android Monitor, Messages, TODO, Event Log, and Gradle Console. It also shows build status: "Gradle build finished in 12s 412ms".

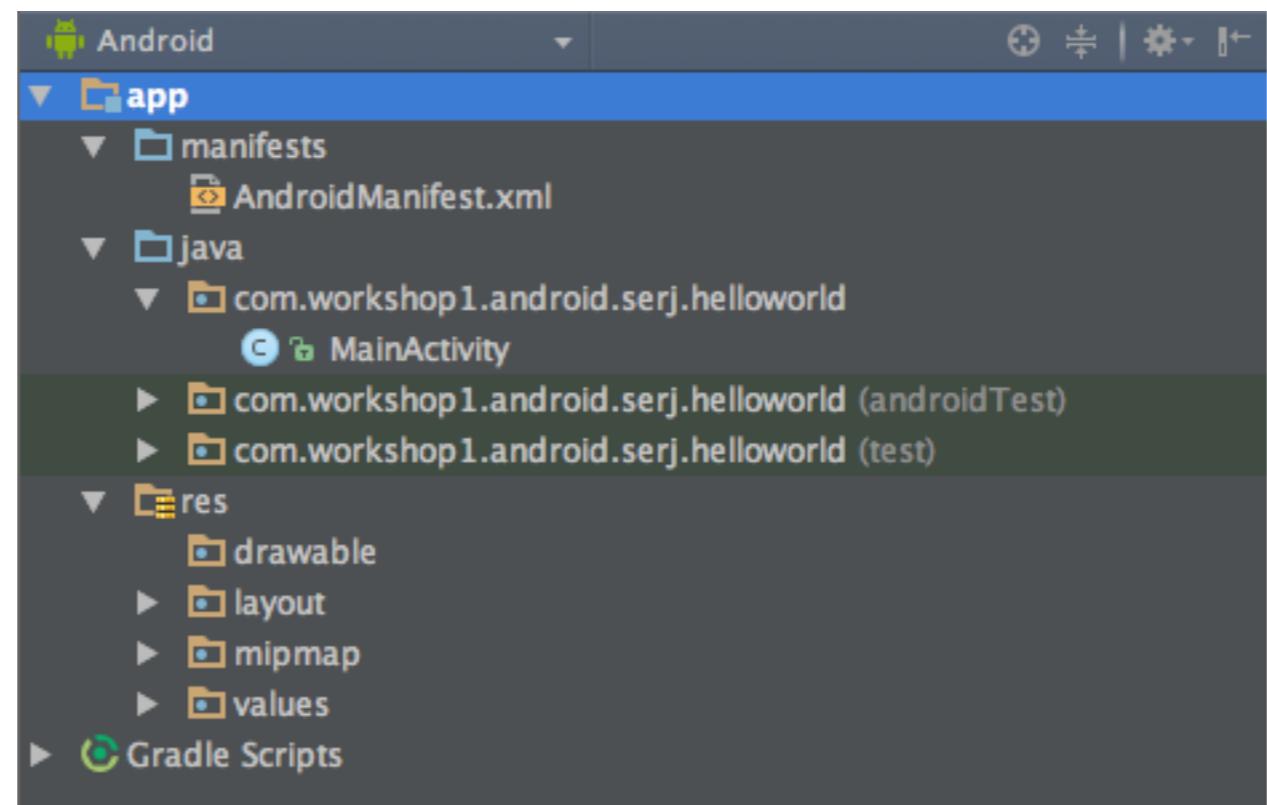


HelloWorld - run configuration





HelloWorld - Project structure





HelloWorld - AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.workshop1.android.serj.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        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>
```

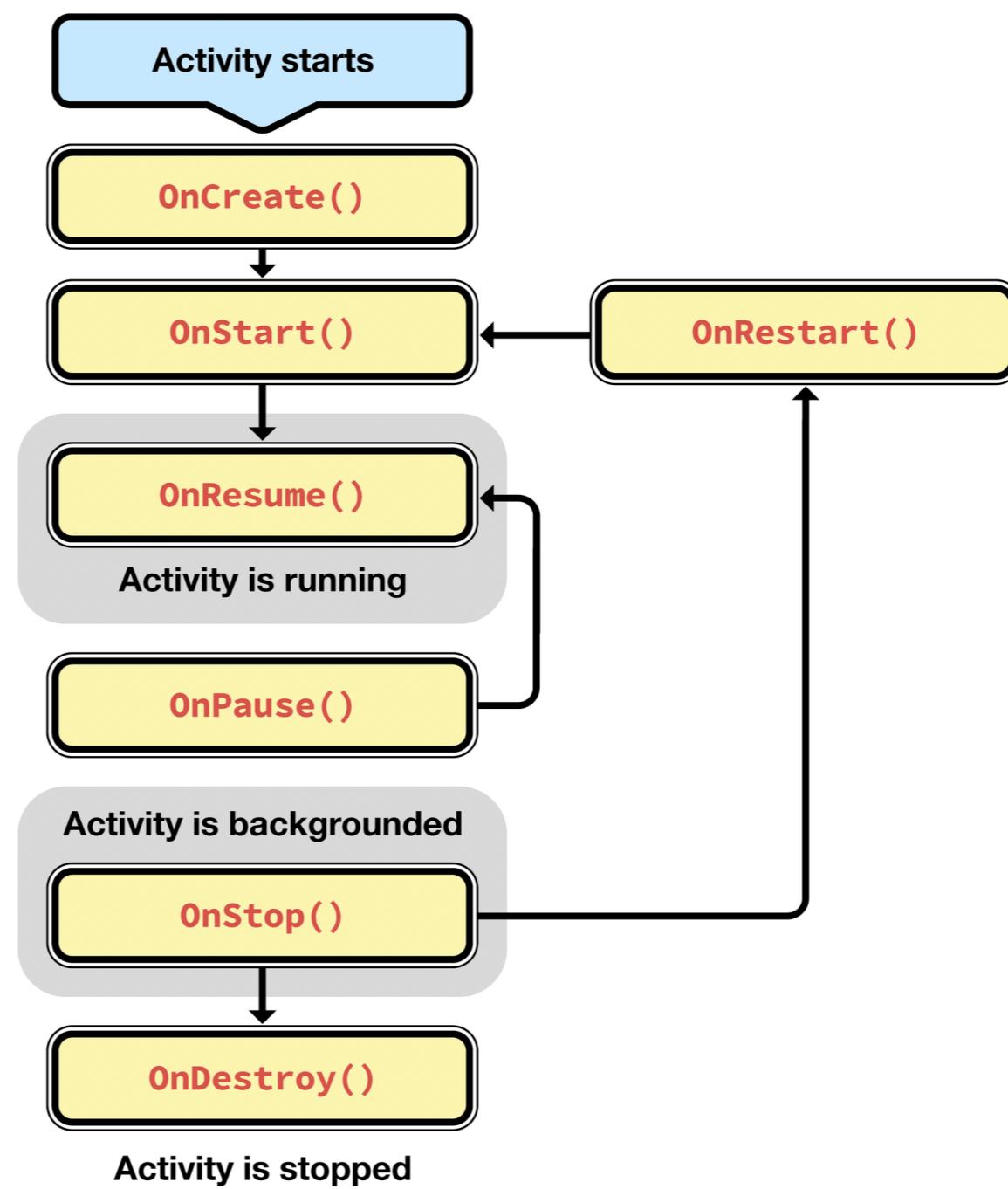


Activity is the ❤️

- Single window (mostly full screen)
- You place your view in `setContentView(View)`
- MVC - Activity is the controller
- It has lifecycle methods
- 1/4 of the basic components of Android apps

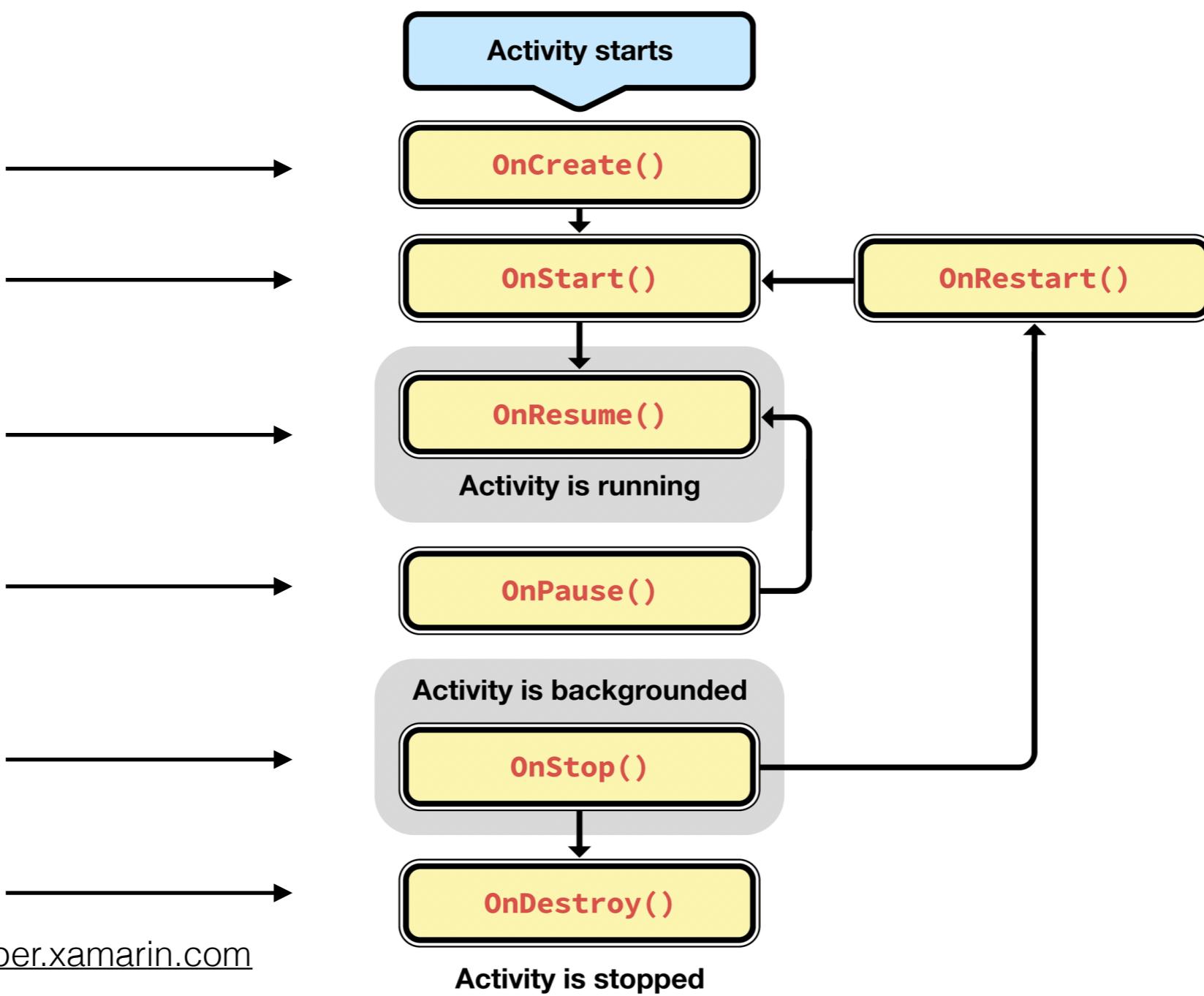


Activity life cycle (state machine)



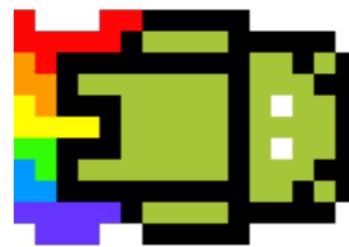


We will implement a couple





Logcat is our friend



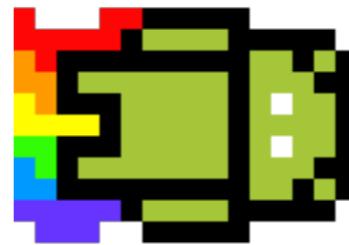
The screenshot shows the Android Monitor interface with the 'logcat' tab selected. The title bar indicates the device is a 'Meizu M2 Android 5.1, API 22' and there are 'No Debuggable Applications'. The logcat window displays several lines of log entries:

```
06-23 16:22:06.643 969-969/? D/SIMHelper: isSimInserted(0), SimInserted=true
06-23 16:22:06.644 969-969/? D/SIMHelper: updateSimInsertedStatus, simInserted(0) = true
06-23 16:22:06.645 969-969/? D/SIMHelper: updateSimInsertedStatus, simInserted(1) = false
06-23 16:22:06.645 969-969/? D/StatusBar.NetworkController: refreshing carrier name: Pelephone
06-23 16:22:06.646 969-969/? D/StatusBar.NetworkController: refreshing carrier name: No SIM
06-23 16:22:06.648 969-1336/? D/Surface: Surface::setBuffersDimensions(this=0x7f9f255a00, w=1080, h=1920)
06-23 16:22:06.649 763-787/? D/PerfServiceManager: [PerfService] notifyFrameUpdate - bDuration=1.000000ms
06-23 16:22:06.652 253-6727/? I/BufferQueueProducer: [StatusBar](this:0x7f9cbe8000, id:6, app:com.miui.home)
06-23 16:22:06.673 253-253/? I/SurfaceFlinger: [Built-in Screen (type:0)] fps:0.969165,duration:1.000000ms
06-23 16:22:06.833 763-854/? D/AutomaticBrightnessController: onSensorChanged: not significant
06-23 16:22:06.914 16871-16871/? I/SurfaceView: updateWindow — OnPreDrawListener, mHaveFrame=false
06-23 16:22:07.033 763-854/? D/AutomaticBrightnessController: onSensorChanged: not significant
```

The bottom navigation bar includes tabs for 'TODO', '6: Android Monitor' (selected), 'Terminal', and '9: Version Control'.



Logcat is our friend



```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = MainActivity.class.getSimpleName();

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

        Log.d(TAG, "onCreate()");
    }
}
```

- We will use Log.d
- There are 5 more(v, i, w, e, wtf)
- TAG string



First assignment

- Print debug messages from life cycle methods:
 - `onCreate()`
 - `onResume()`
 - `onPause()`
 - `onDestroy()`



Questions

- How's the life cycle is different from a C,C++,Java program ?
- What are the unique challenges of mobile ?



Answers

- Q: How's the life cycle is different from a C,C++,Java program ?
- A: C,C++ etc programs have a single entry point - static void main(), vs Android multi entry point
- Q: What are the unique challenges of mobile ?
 - A: Multi task - switching applications constantly. Like a phone call.
 - A: Low on resources, things can get destroyed on demand



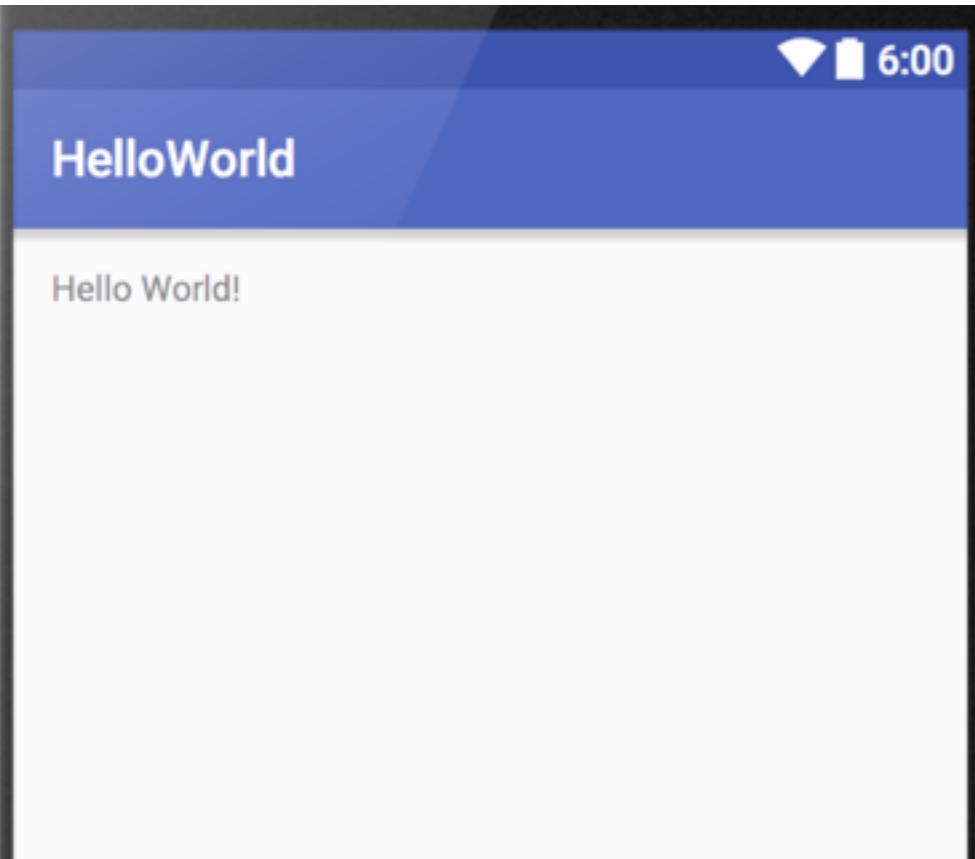
Bonus question

- Q: If we would print the lifecycle messages on screen, which ones we will see ?



Views - activity_main.xml

Outcome



XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.workshop1.android.serj.helloworld.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

*layouts conventions. activity_..



Views - activity_main.xml

XML

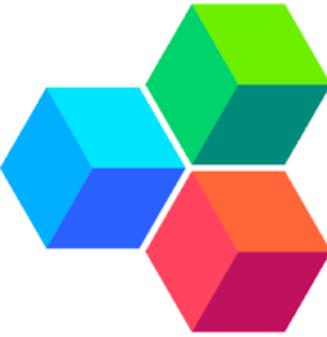
Parent

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.workshop1.android.serj.helloworld.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

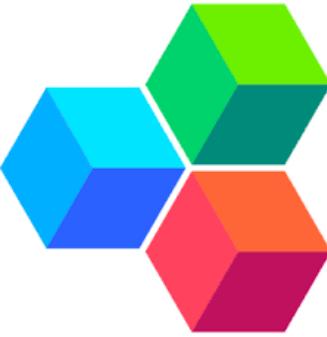
Children

*layouts conventions. activity_..



Layouts

- Layouts describe how their children will be ordered.
- **RelativeLayout** - children location are relative to one another. (Not the best parenting)
- **LinearLayout** - one after another



Layouts example

LinearLayout VS. RelativeLayout

LinearLayout



RelativeLayout





Important attributes

- Size of layout:
- layout_width
- layout_height

```
<RelativeLayout 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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world!" />

```



wrap_content

wrap_content

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    ...  
/>
```





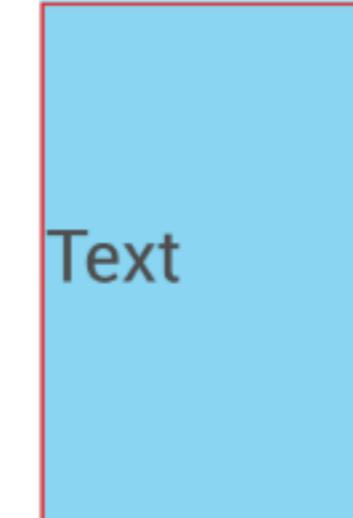
match_parent

match_parent

`android:layout_width="wrap_content"`
 `android:layout_height="match_parent"`

`android:layout_width="match_parent"`
 `android:layout_height="wrap_content"`

`android:layout_width="match_parent"`
 `android:layout_height="match_parent"`





layout_width, layout_height

- Can be also constant size.
- layout_width = “15dp”
- layout_height = “20dp”



Java vs XML

- **Java** is a multi paradigm language: **Object oriented, procedural** and more. It describes **how** things are **done**.
- **XML** is **declarative**. It describes **how** things **look**.
- All of the things we can do with XML, are also Java compatible.

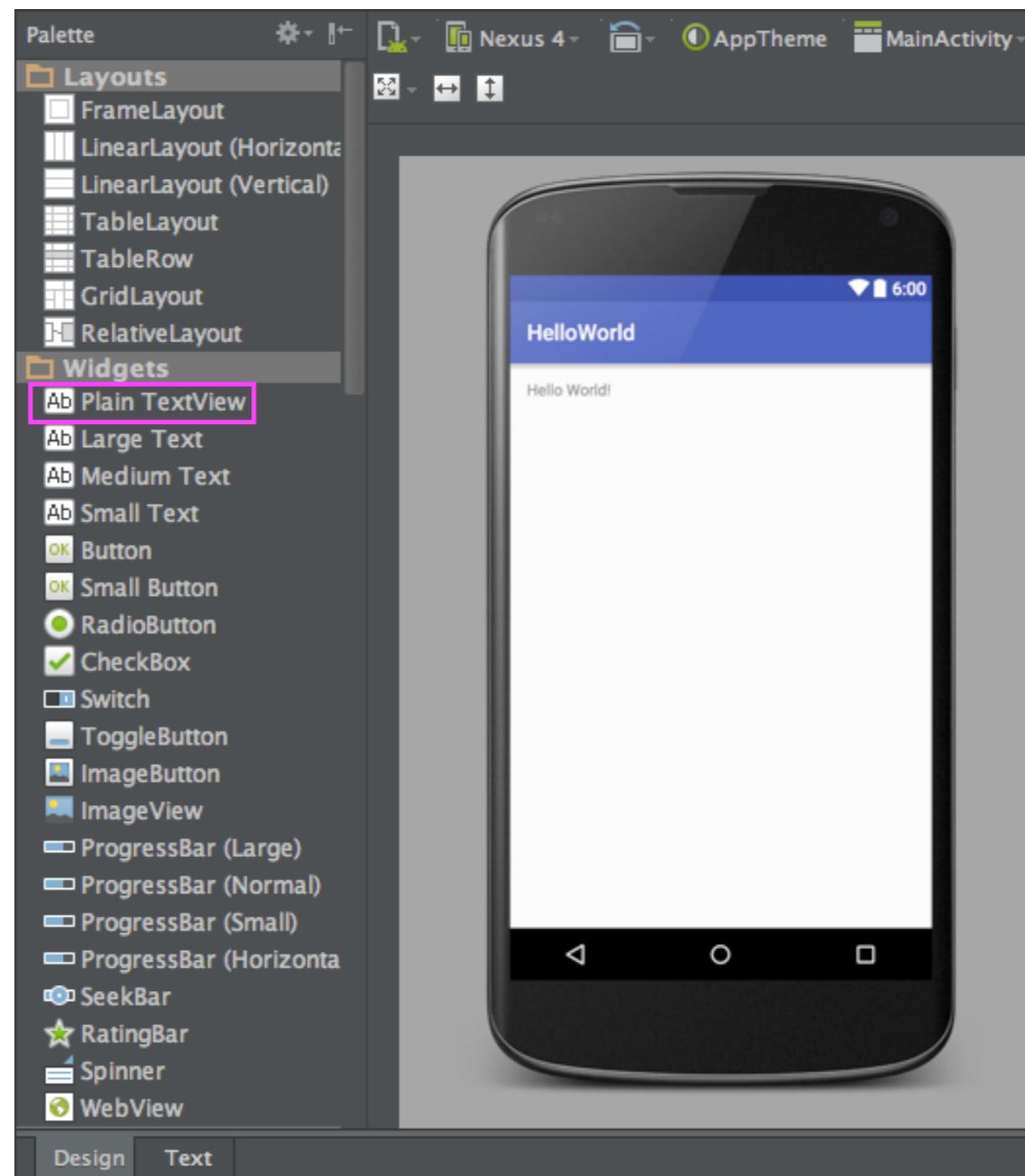


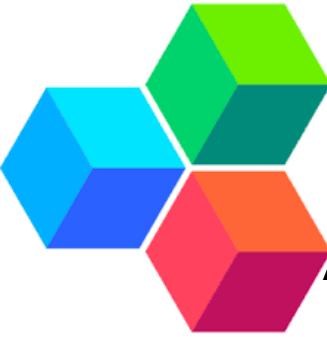
Third assignment - activity_main.xml

- **Add another TextView** - dragging for the design screen
- **Add another TextView** - coding by hand
- **Switch** the **RelativeLayout** to **LinearLayout**



Reminder - design screen





Android Studio shortcuts - more to come

- cmd + o - open class
- cmd + f12 - structure + methods

More: <http://www.developerphil.com/android-studio-tips-tricks-moving-around/>



Start second assignment



Buttons



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.workshop1.android.serj.helloworld.MainActivity">

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```



OnClickListener

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = MainActivity.class.getSimpleName();

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

        Log.d(TAG, "onCreate()");

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.d(TAG, "button onClick()");
            }
        });
    }
}
```



OnClickListener

```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.d(TAG, "button onClick()");
    }
});
```



OnClickListener

- Does what it says
- It's an Anonymus Class
- Similar to Java Swing - GUI programming - left the procedural realm



Message to screen - Toast

```
Toast.makeText(MainActivity.this, "button onClick()",  
Toast.LENGTH_SHORT).show();
```



Third assignment - A click counter

- Add a counter variable
- Show it using the *TextView*
- Manipulate the counter after a button click
- Set the new result



R class

- Is generated for us
- Lots of constants we can use



findViewById(R.id.view)

- Does what it says
- It's a method we inherit from Activity
- Can only find what's in our layout (setContentView)



Important attribute - id

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

- That's how we find our views programmatically.

```
final TextView textView = (TextView) findViewById(R.id.textView);
```



Start third assignment



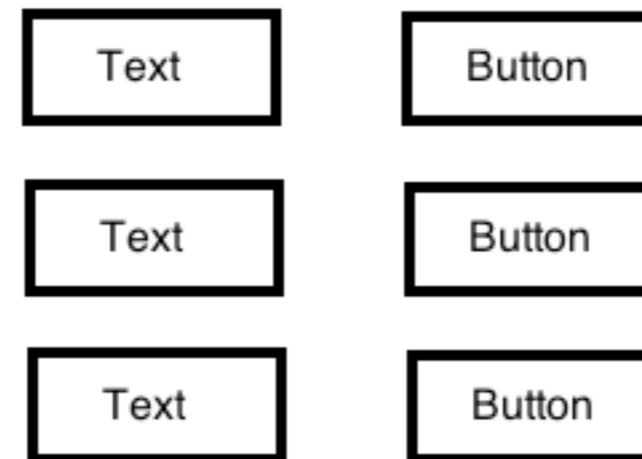
Summary

- Android is kewl and China is gonna rule around 2018
- Creating a project
- Activity is the ❤
- Logcat is console like
- Views - layouts are parents, views are children
(layouts can be children too)
- Button OnClickListener



Home work

- A client asked for a grid like app, each TextView should hold a separate counter
- Bonus - use three horizontal LinearLayout for each row, and a vertical LinearLayout for their parent





Next time

- Goal - TODO app