Android sweat shop

first meeting 23.6

Me - Serj Smorodinsky

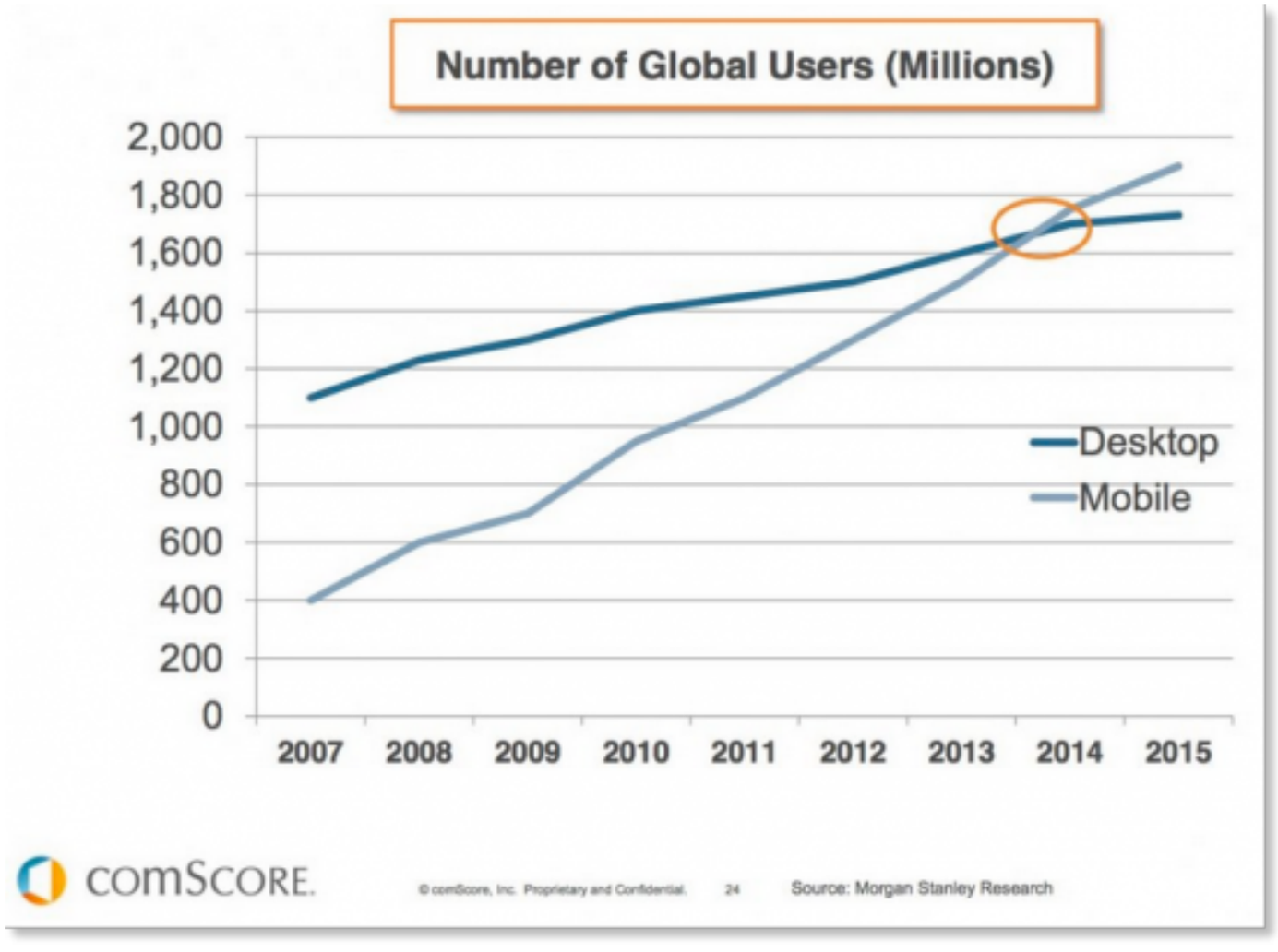
- M.S.c Bioinformatics
- 3 Years of Android development
- Moto: practice first theory second
- Serjsmor@gmail.com
- Github: https://github.com/serjSmor
- Blog: https://serjdroid.wordpress.com

Sweat shop info

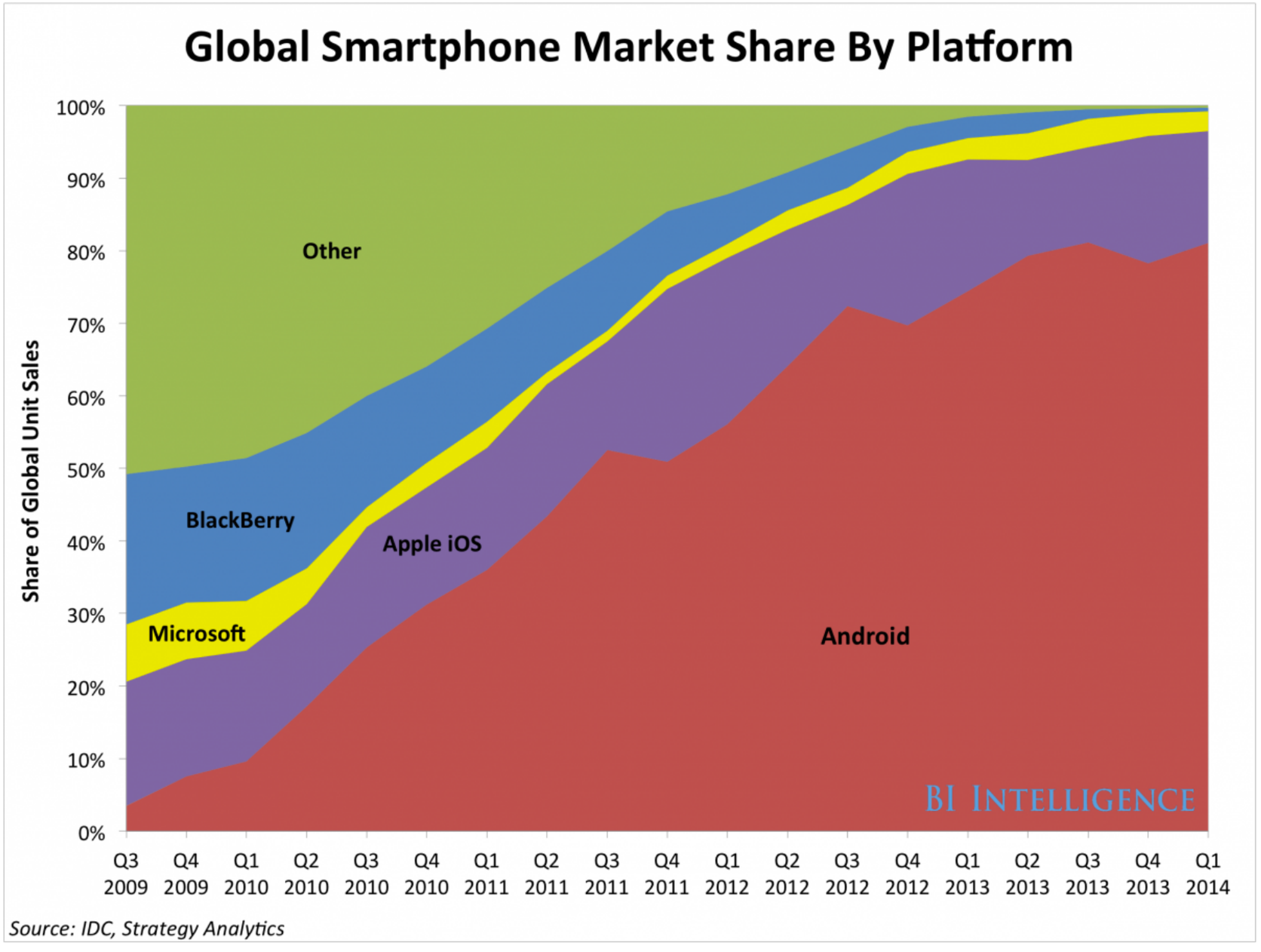
- I count that you are familiar with some OOP language
- Emphasis on coding first
- Project oriented - we are learning in order to accomplish a task
- Projects and slides will be available on Github
- It’s a headstart, and not a complete Android course

Table of contents

(Why)
Desktop to mobile

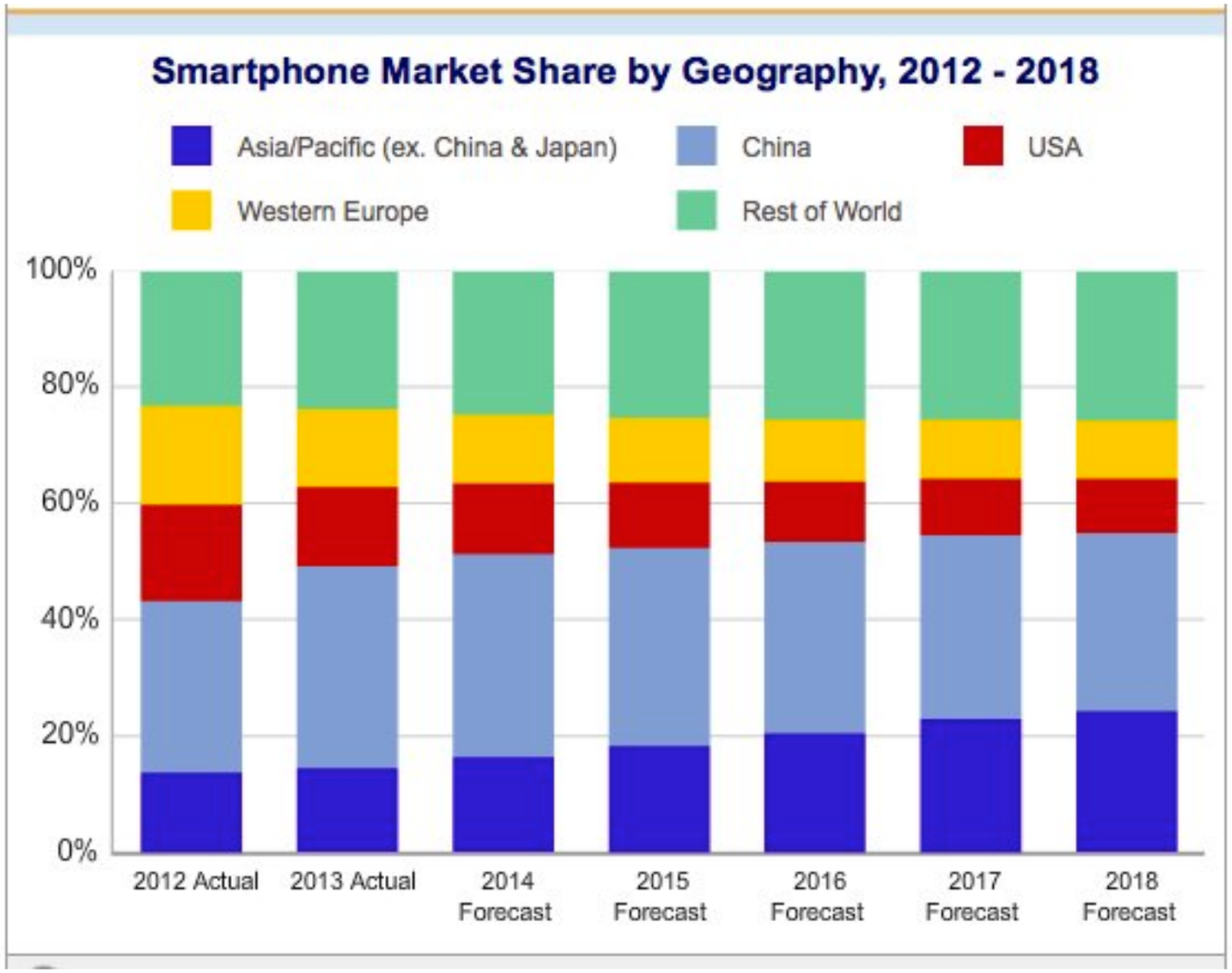


From iOS to Android



Source: IDC, Strategy Analytics

West to east

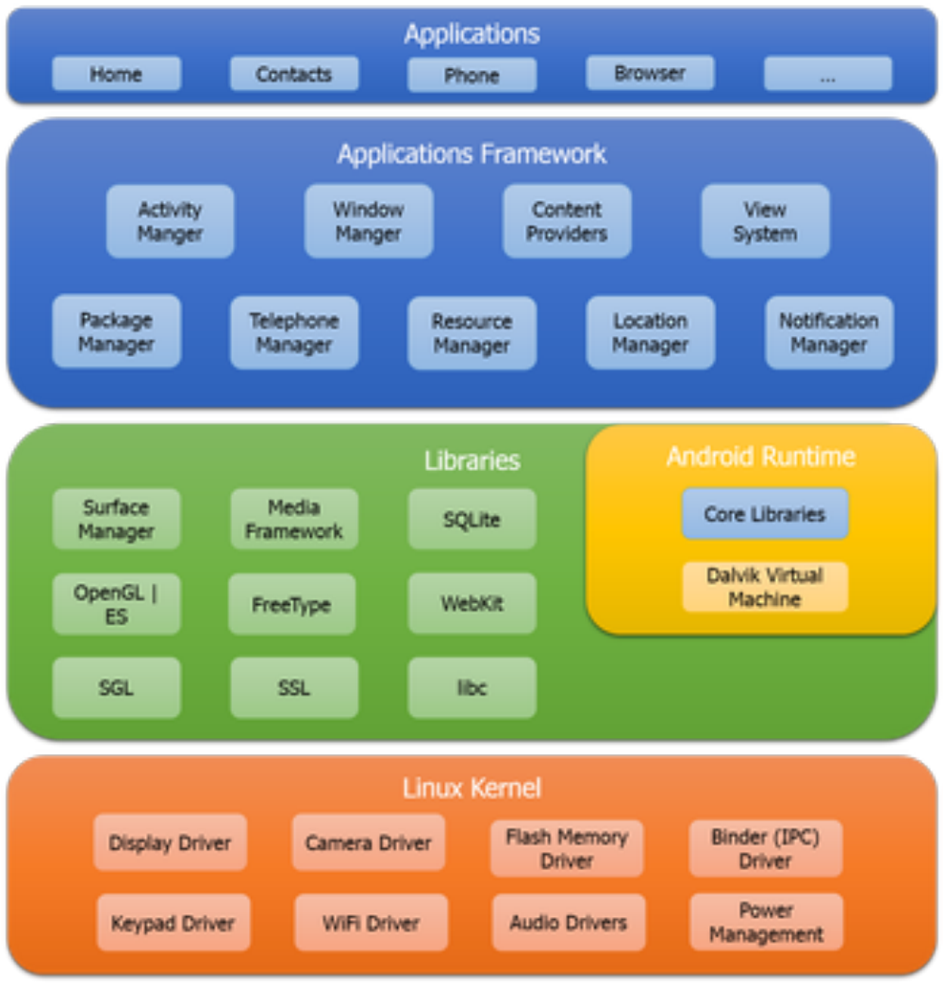

Smartphone Market Share by Geography, 2012 - 2018
Asia/Pacific (ex. China & Japan)
China
USA
Western Europe
Rest of World
100%
80%
60%
40%
20%
0%
2012 Actual
2013 Actual
2014 Forecast
2015 Forecast
2016 Forecast
2017 Forecast
2018 Forecast

Future of platform

Trivia

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

(What)
Android architecture

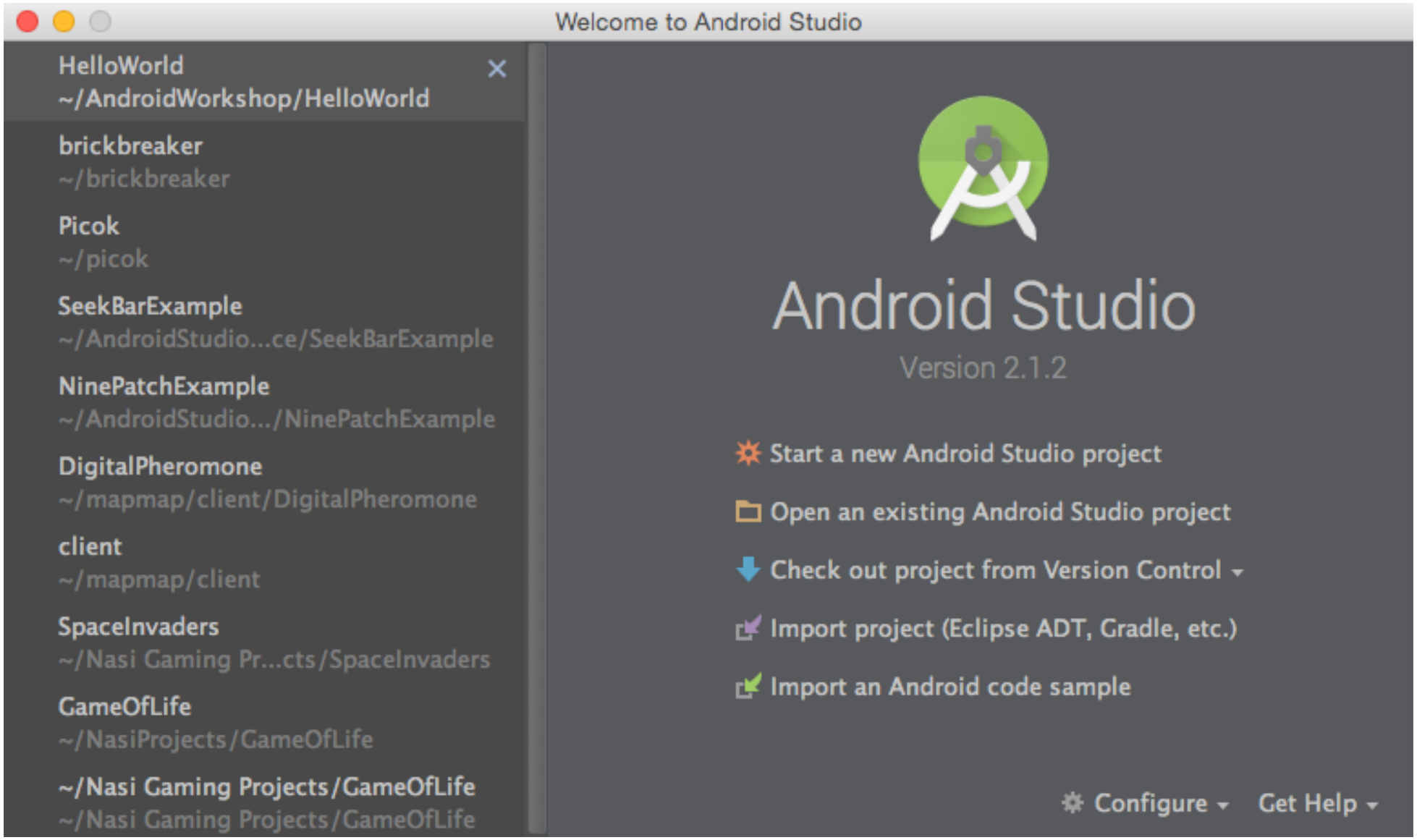


From: https://lokeshv.wordpress.com/tag/android/

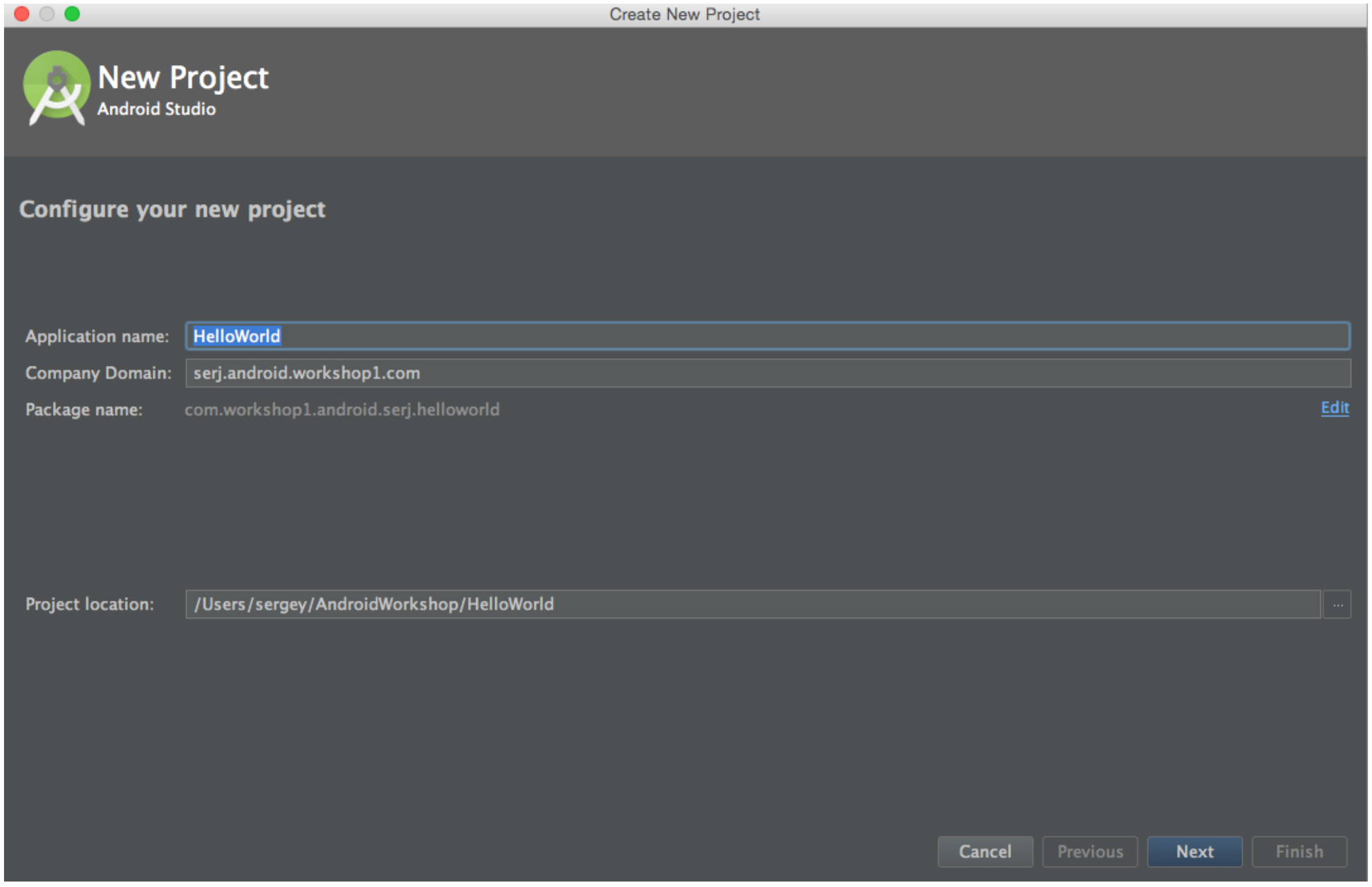
(What)

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

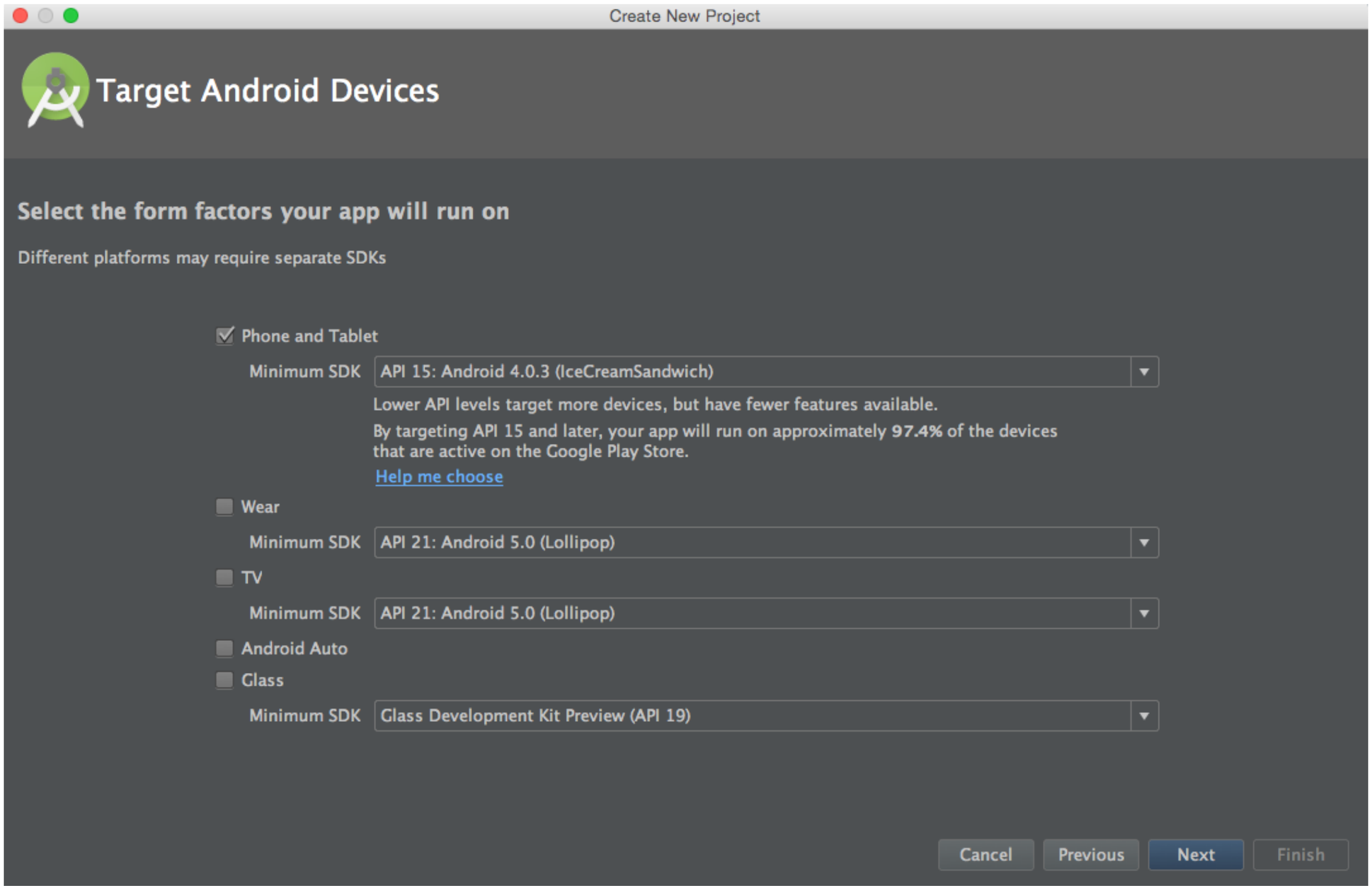
HelloWorld - create new project



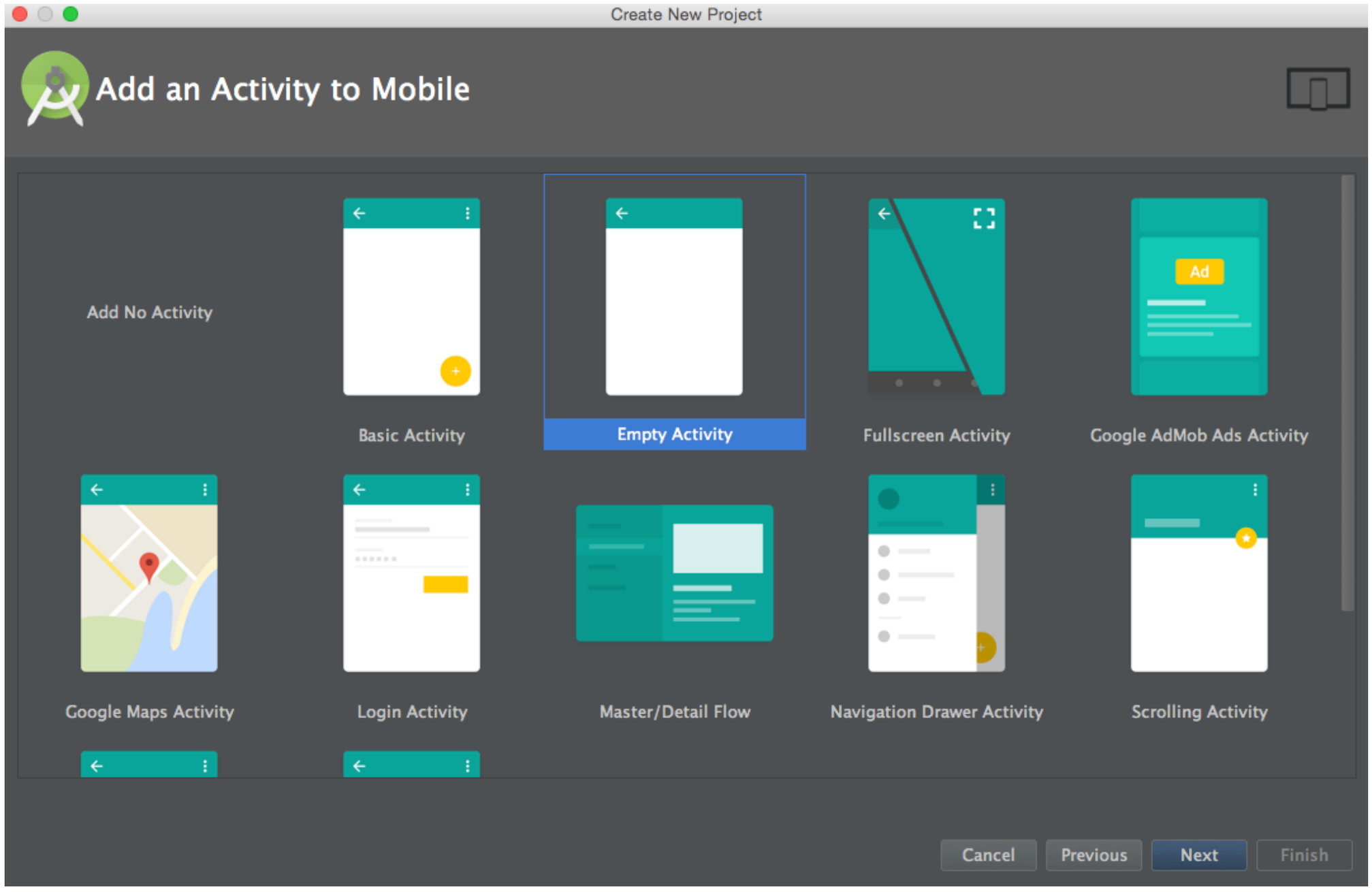
HelloWorld - name it



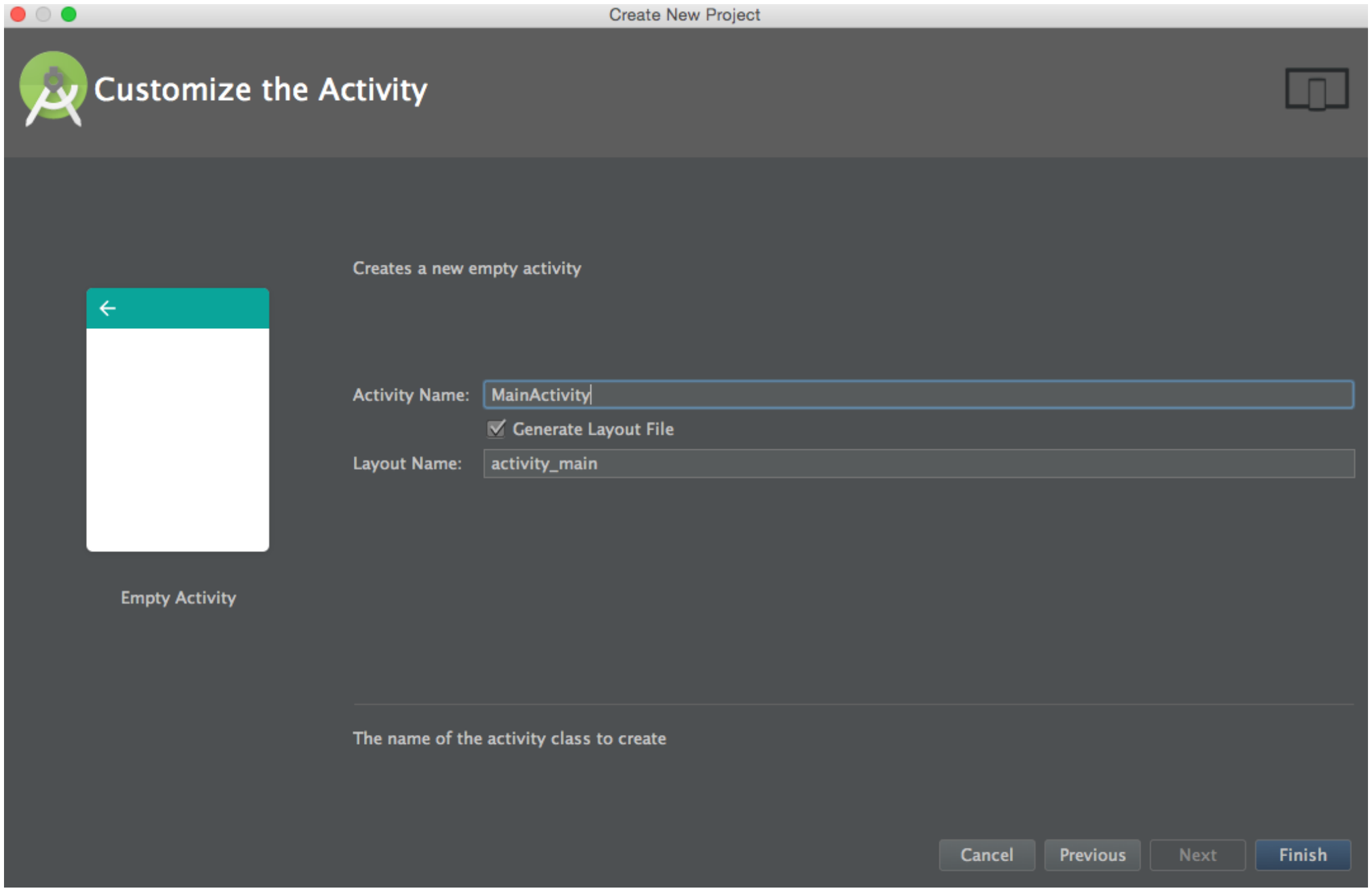
HelloWorld - targets



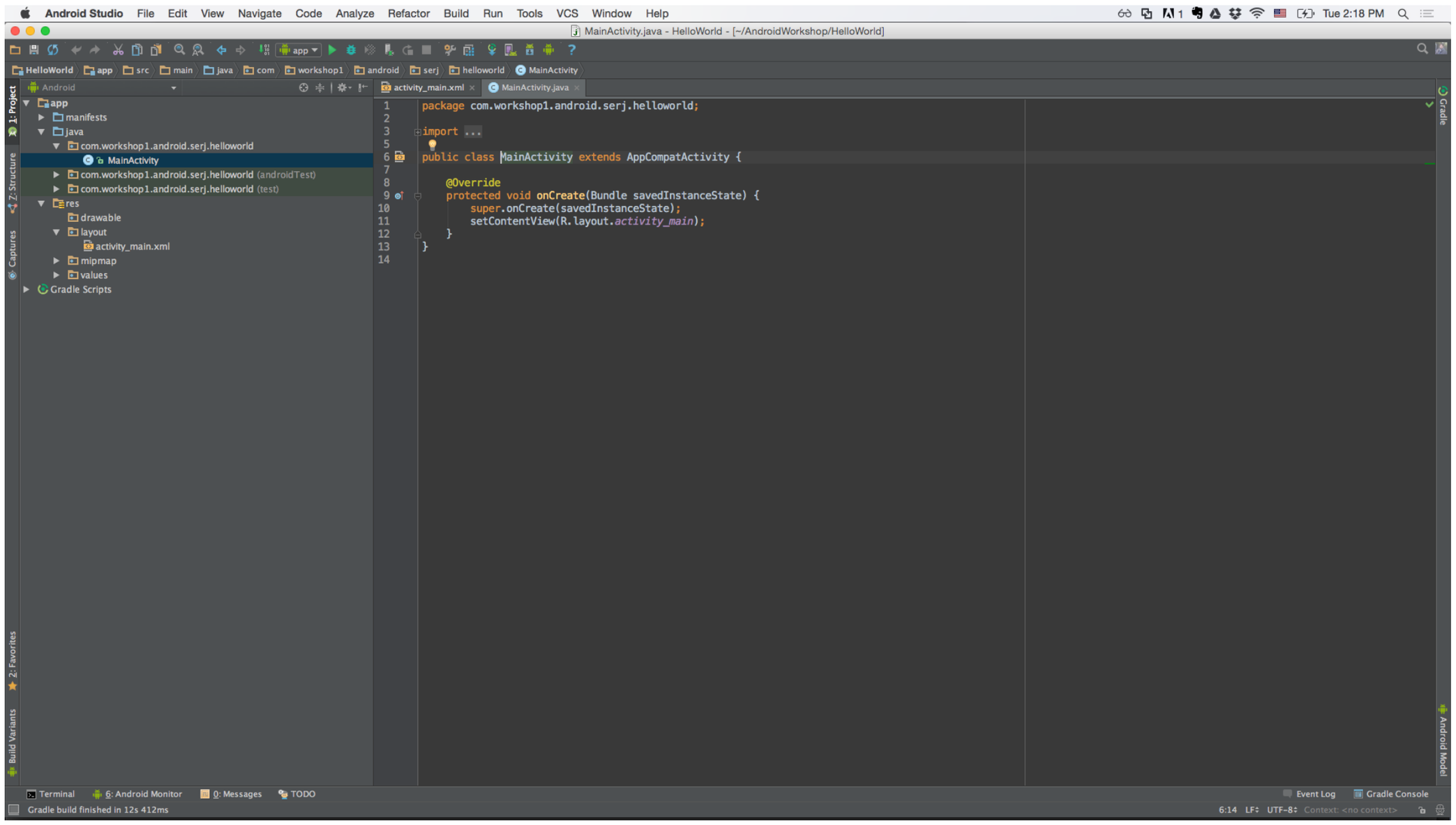
HelloWorld - templates



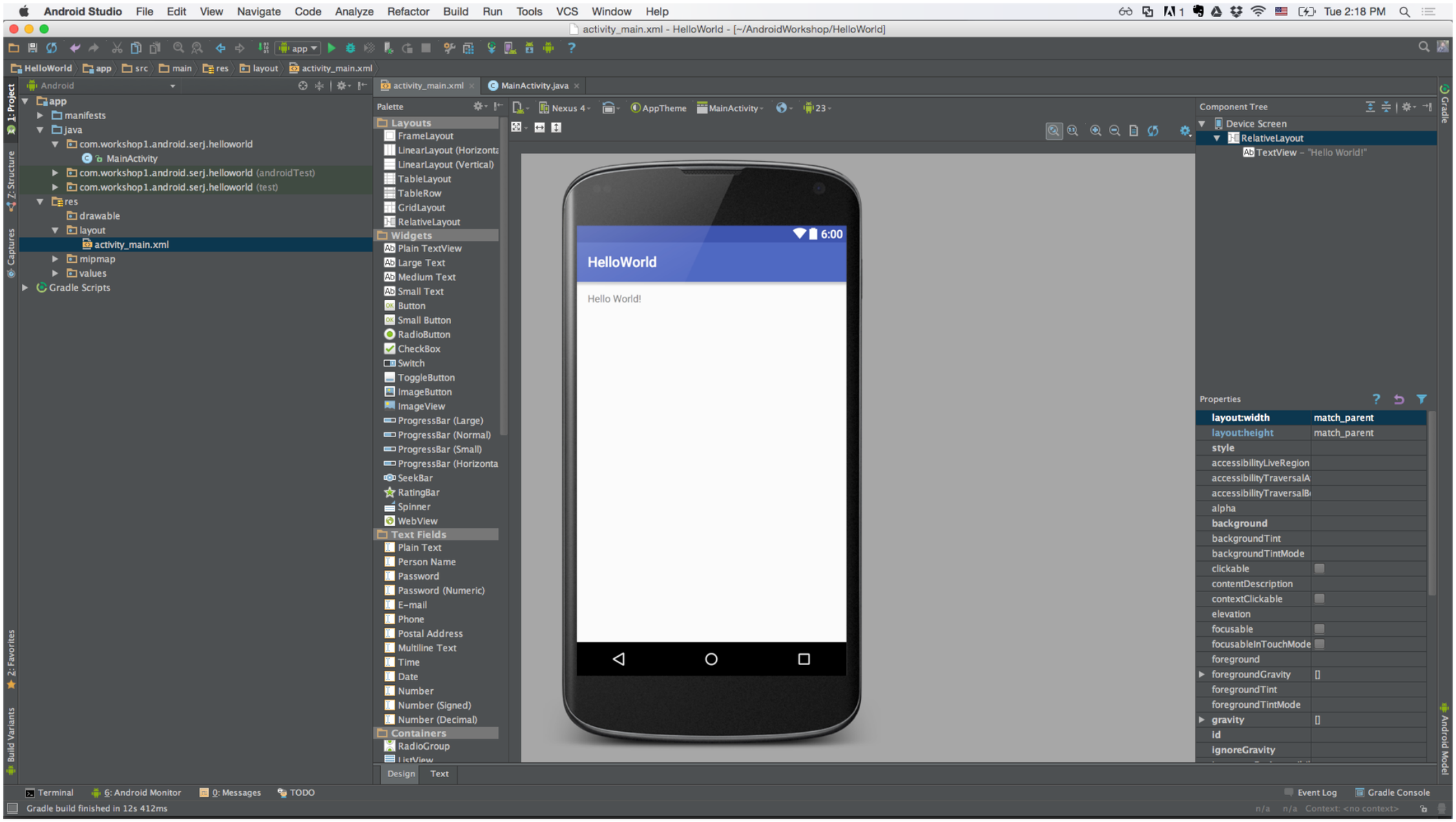
HelloWorld - name activity



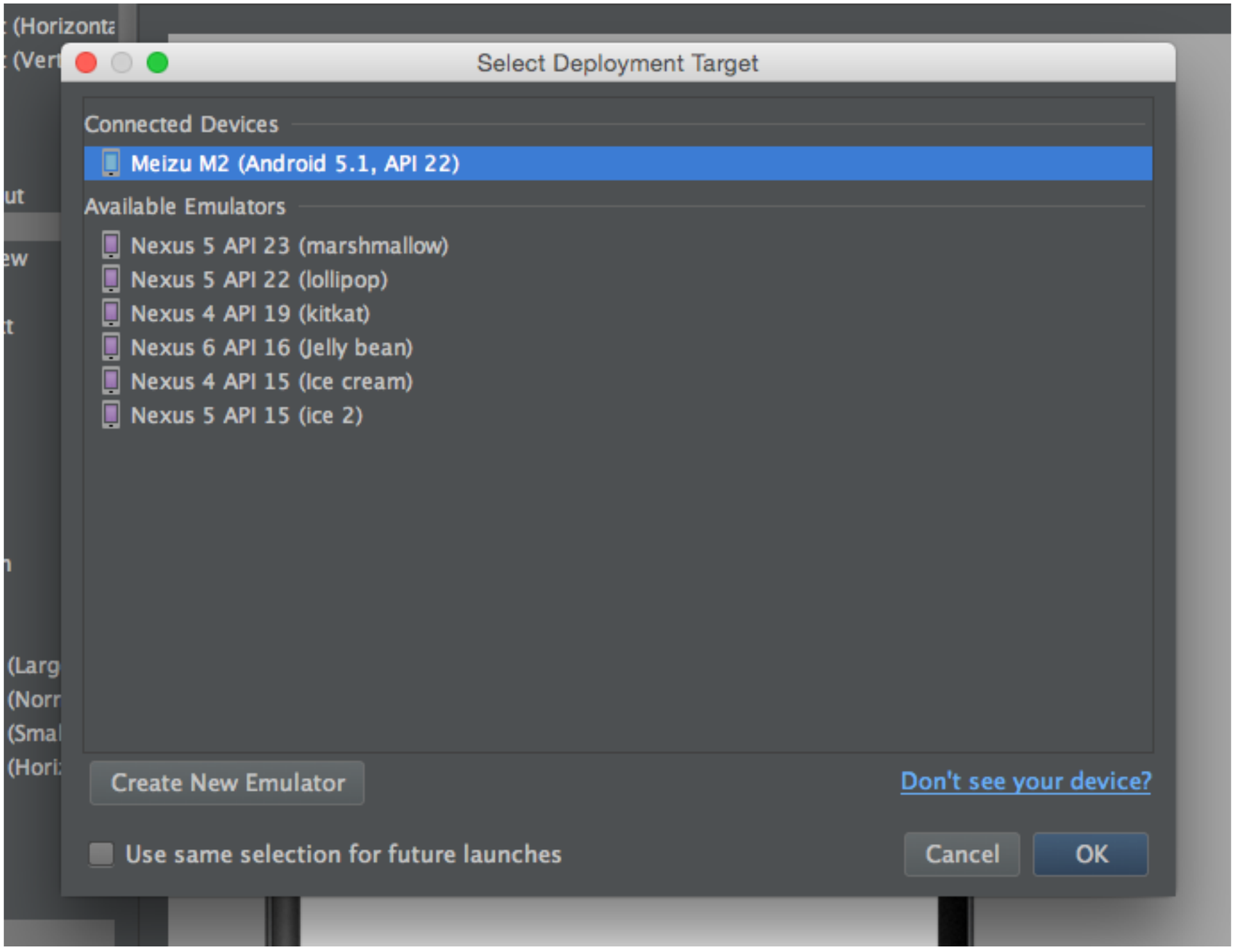
HelloWorld - MainActivity



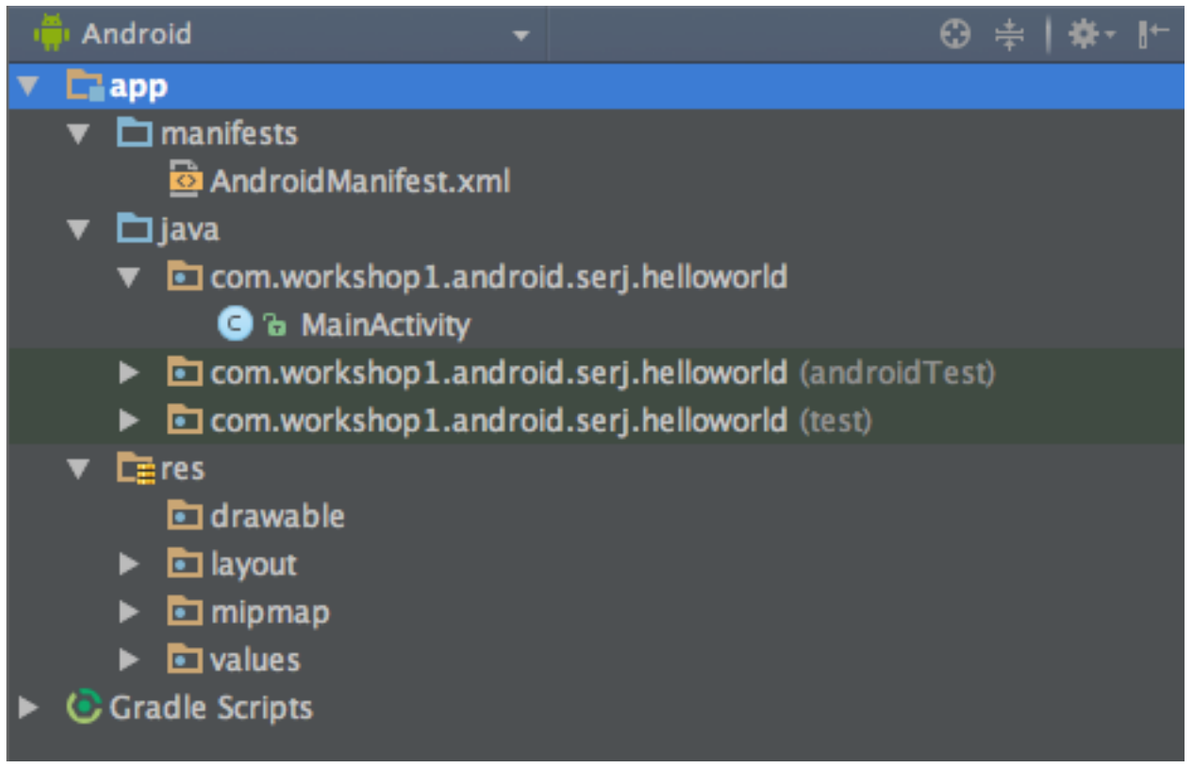
HelloWorld - activity_main preview



HelloWorld - run configuration



HelloWorld - Project structure



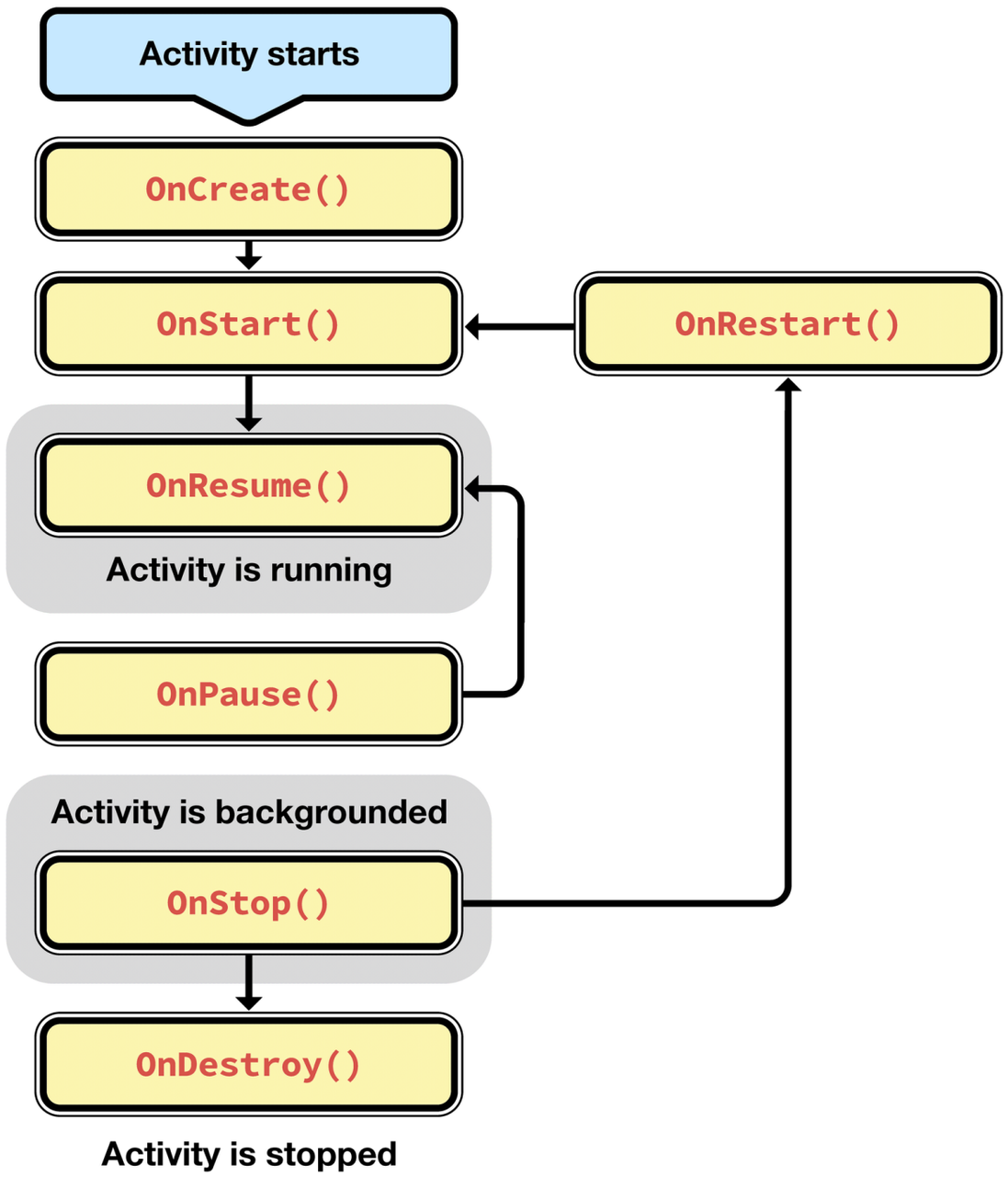
HelloWorld - AndroidManifest.xml

```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 ❤️

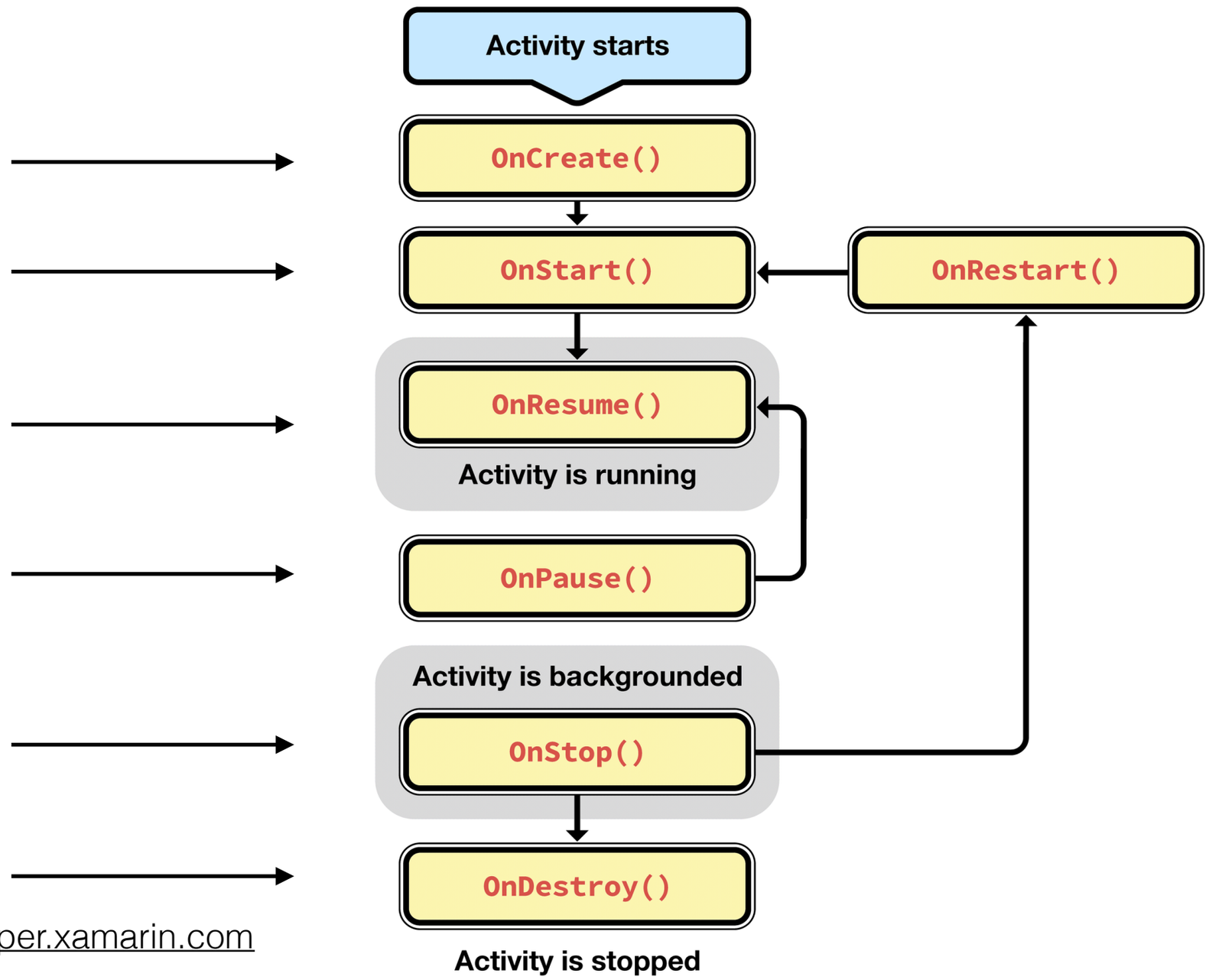
- Single window (mostly full screen)
- You place your view in *setContentView(View)*
- MVC - Activity is the controller
- It has lifecycle methods
- 1/4 of the basic components of Android apps

Activity life cycle (state machine)



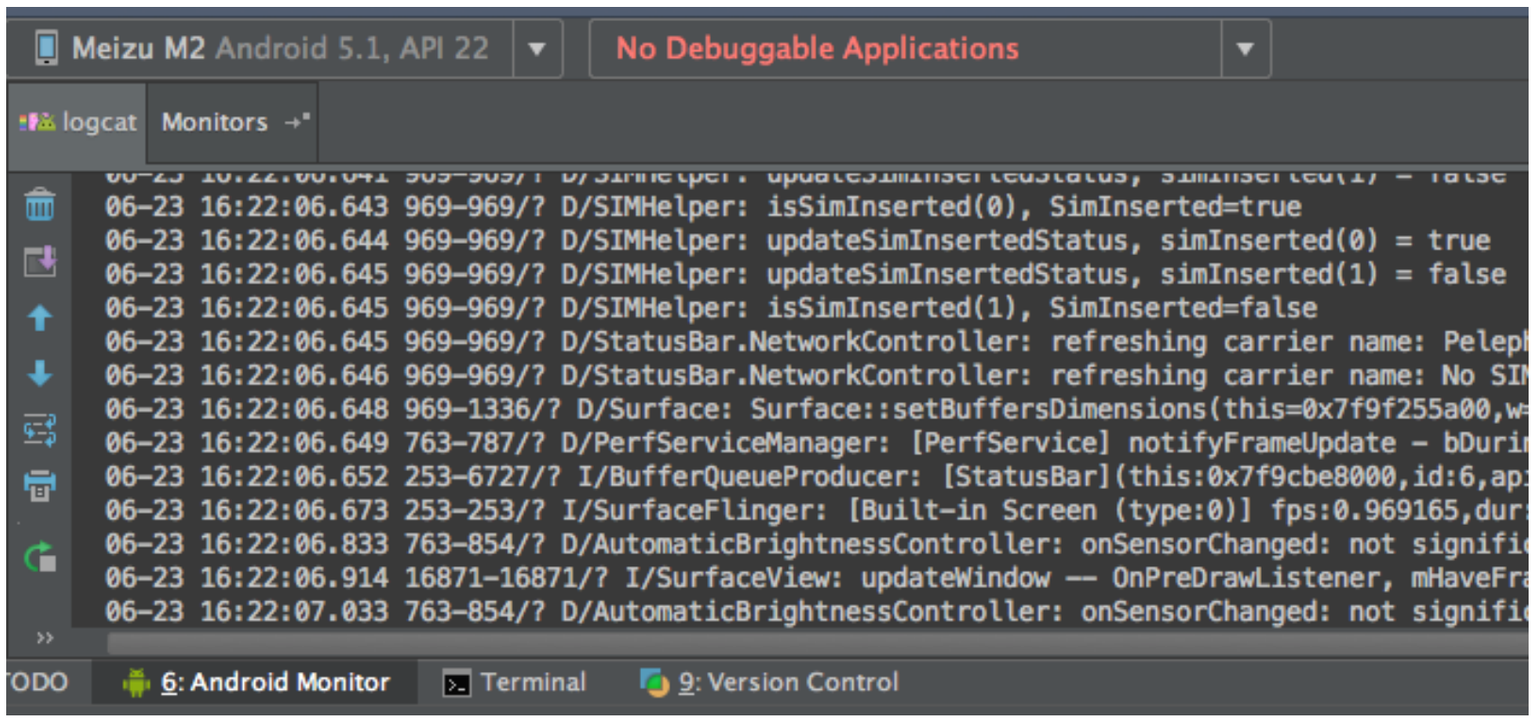
from: developer.xamarin.com

We will implement a couple



from: developer.xamarin.com

Logcat
is our friend



from: elektronengehirn.blogspot.com

Logcat
is our friend

```java
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

from: elektronengehirn.blogspot.com

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.

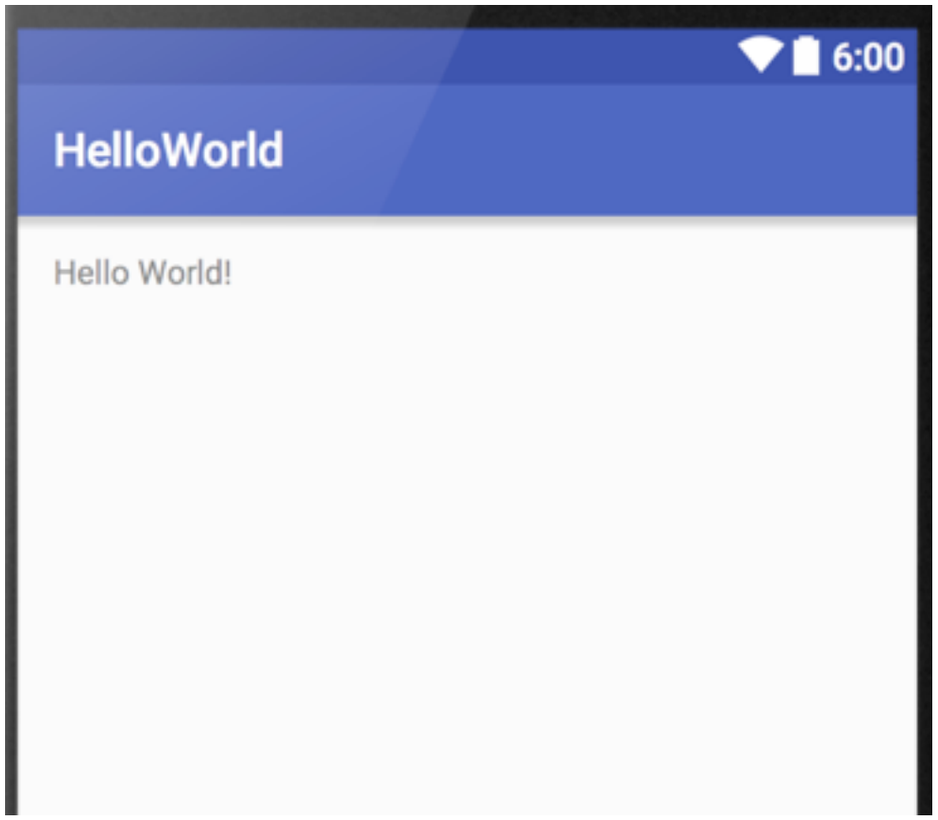
  - A: Low on resources, things can get destroyed on demand

Bonus question

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

Views - activity_main.xml

Outcome



XML

```xml
<?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

Children

```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_..

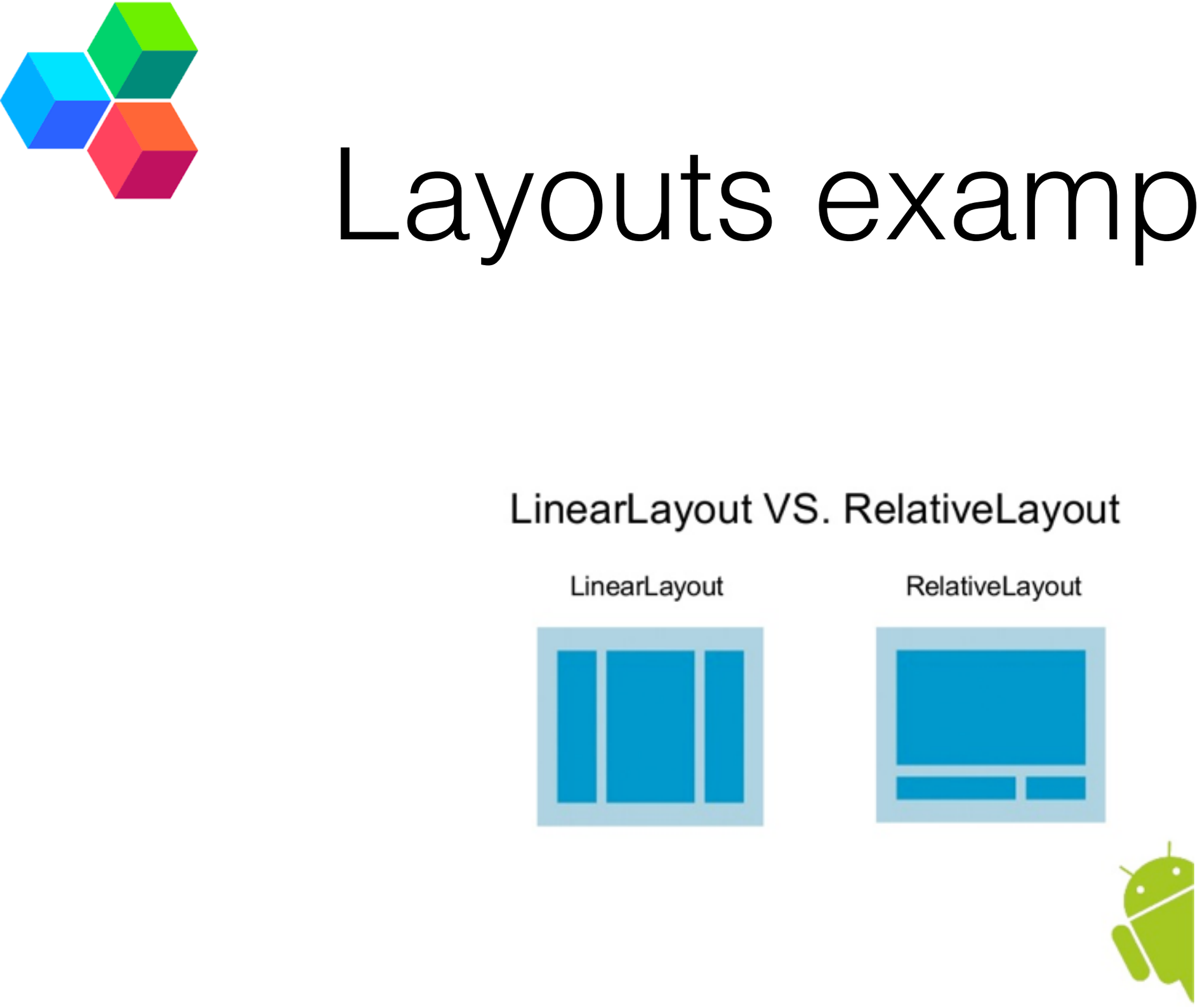
Layouts

- 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



From: http://www.slideshare.net/vivekbhusal/android-training-day-2

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!" />
</RelativeLayout>
```

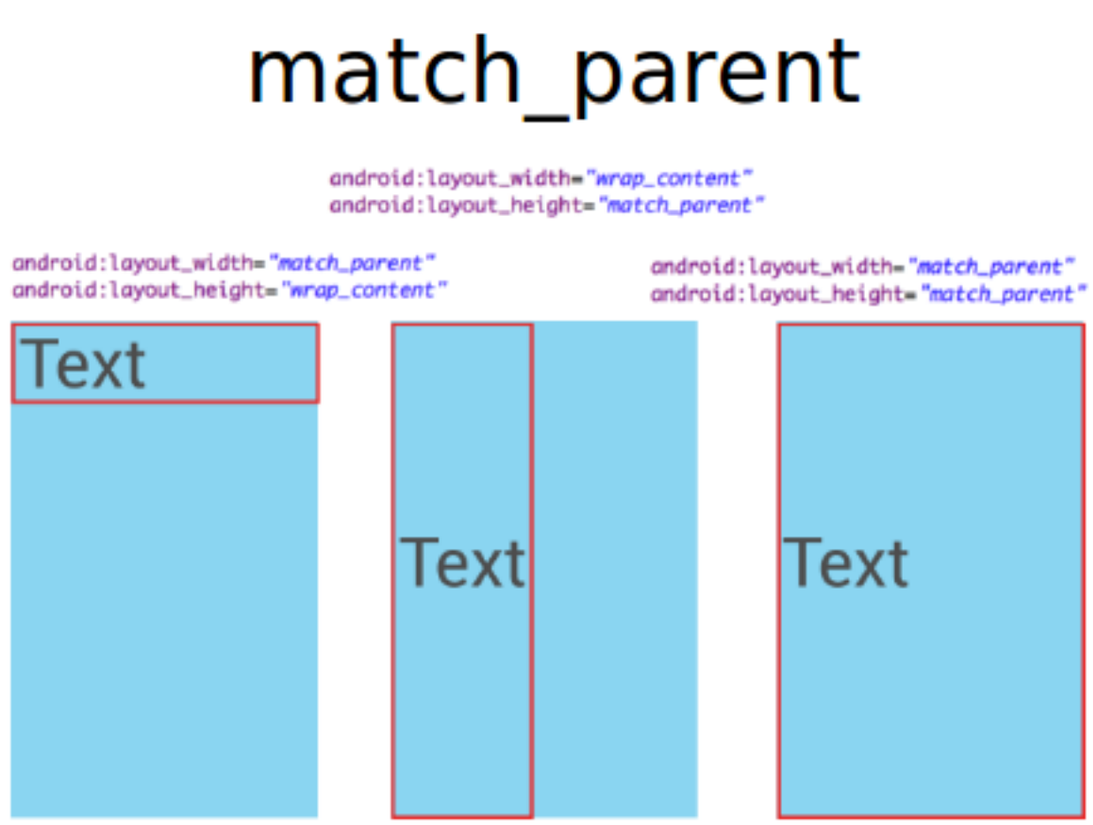
wrap_content

wrap_content

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

From: http://www.vogella.com/tutorials/Android/article.html

match_parent



From: http://www.vogella.com/tutorials/Android/article.html

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**.

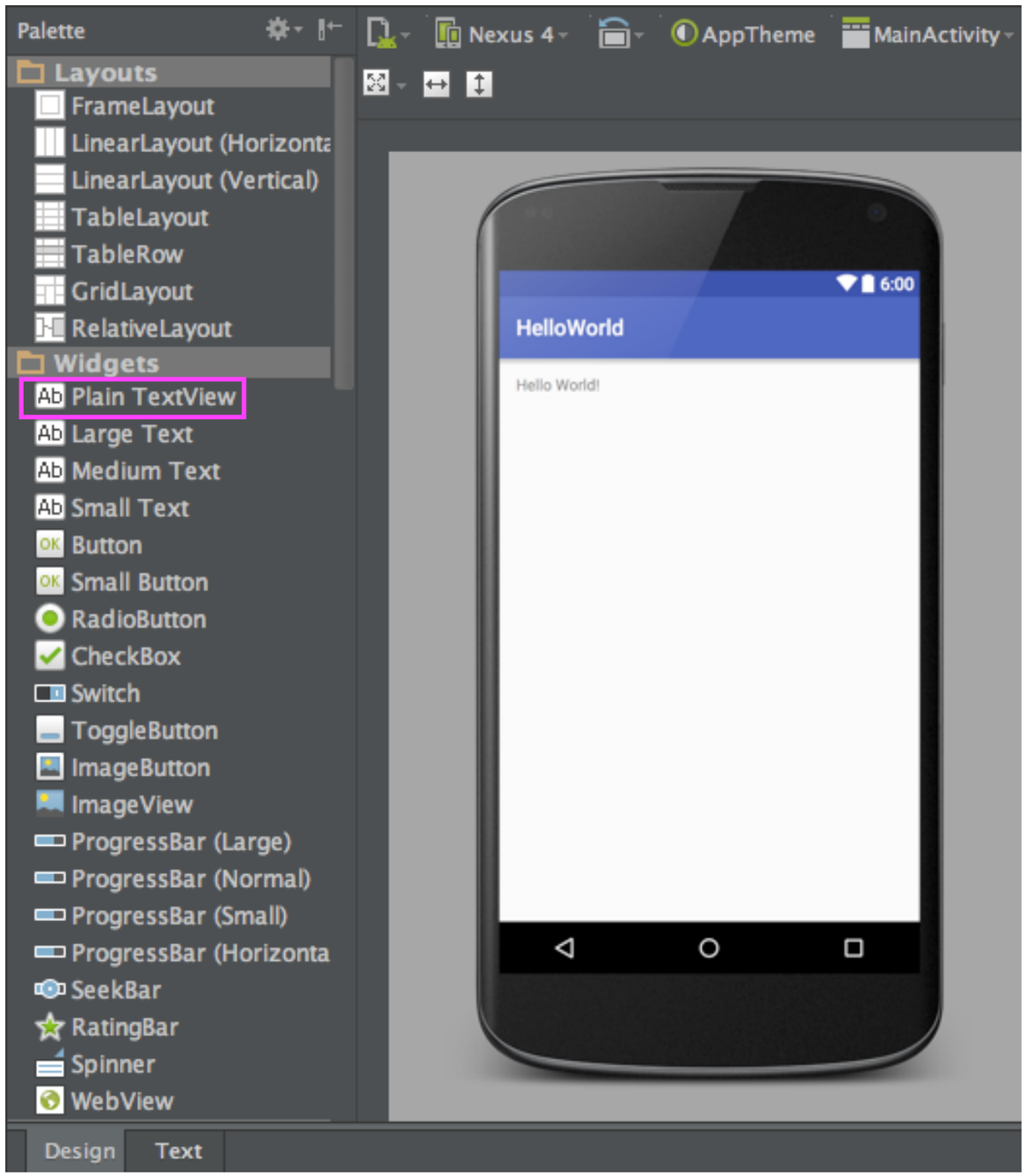
- **XML** is **declarative**. It describes **how** things **look**.

- 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

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

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

Start second assignment

Buttons


```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!"
        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

```java
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

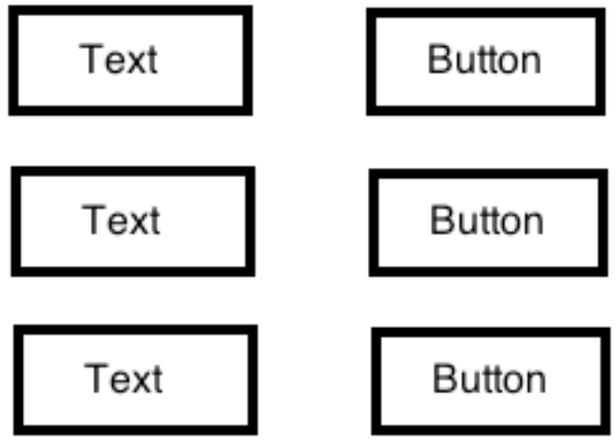
Summary

- 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