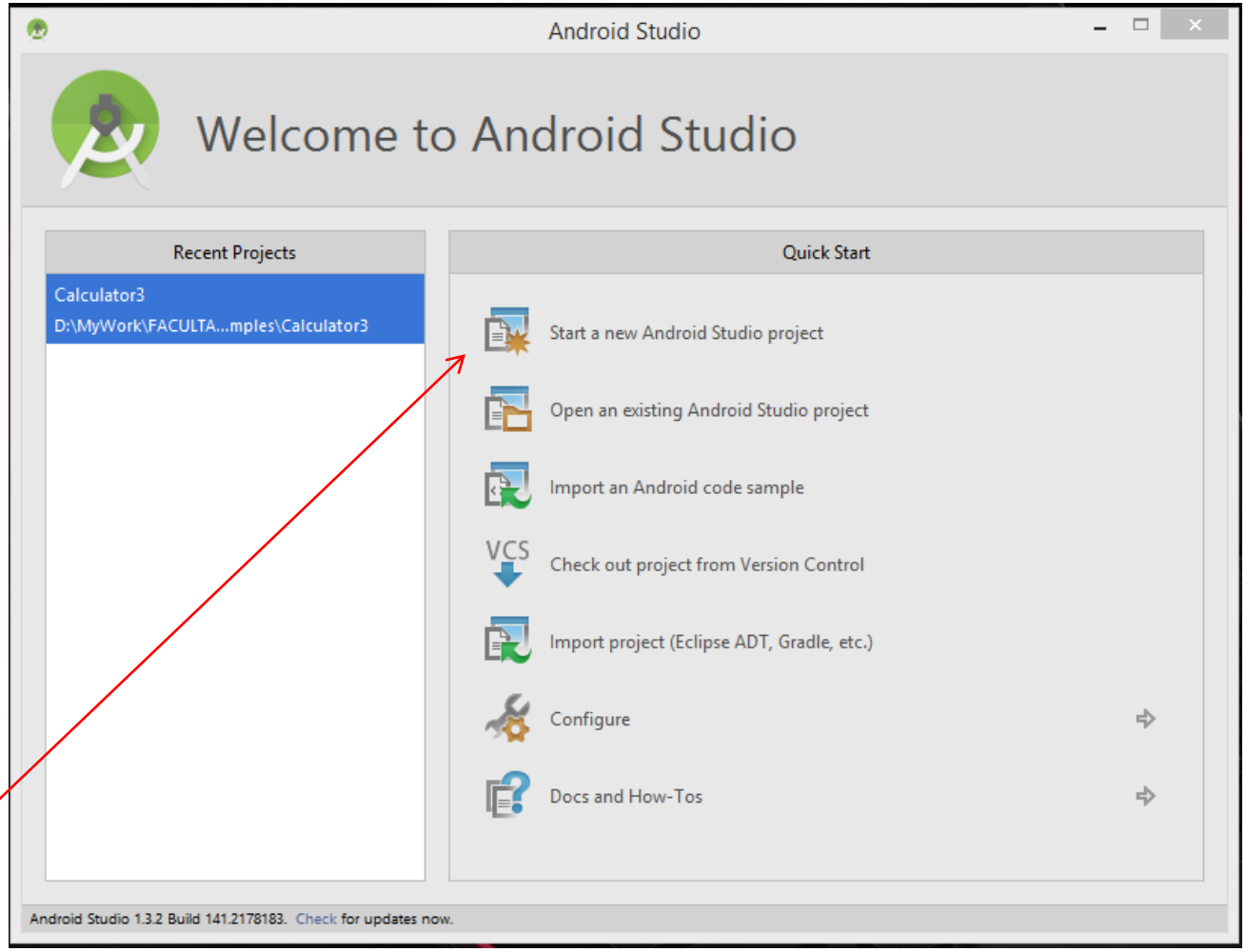
Android Studio

West University of Timisoara, Romania
Computer Science Department
IE3, Fall 2015
Dr. Liviu Octavian Mafteiu-Scai

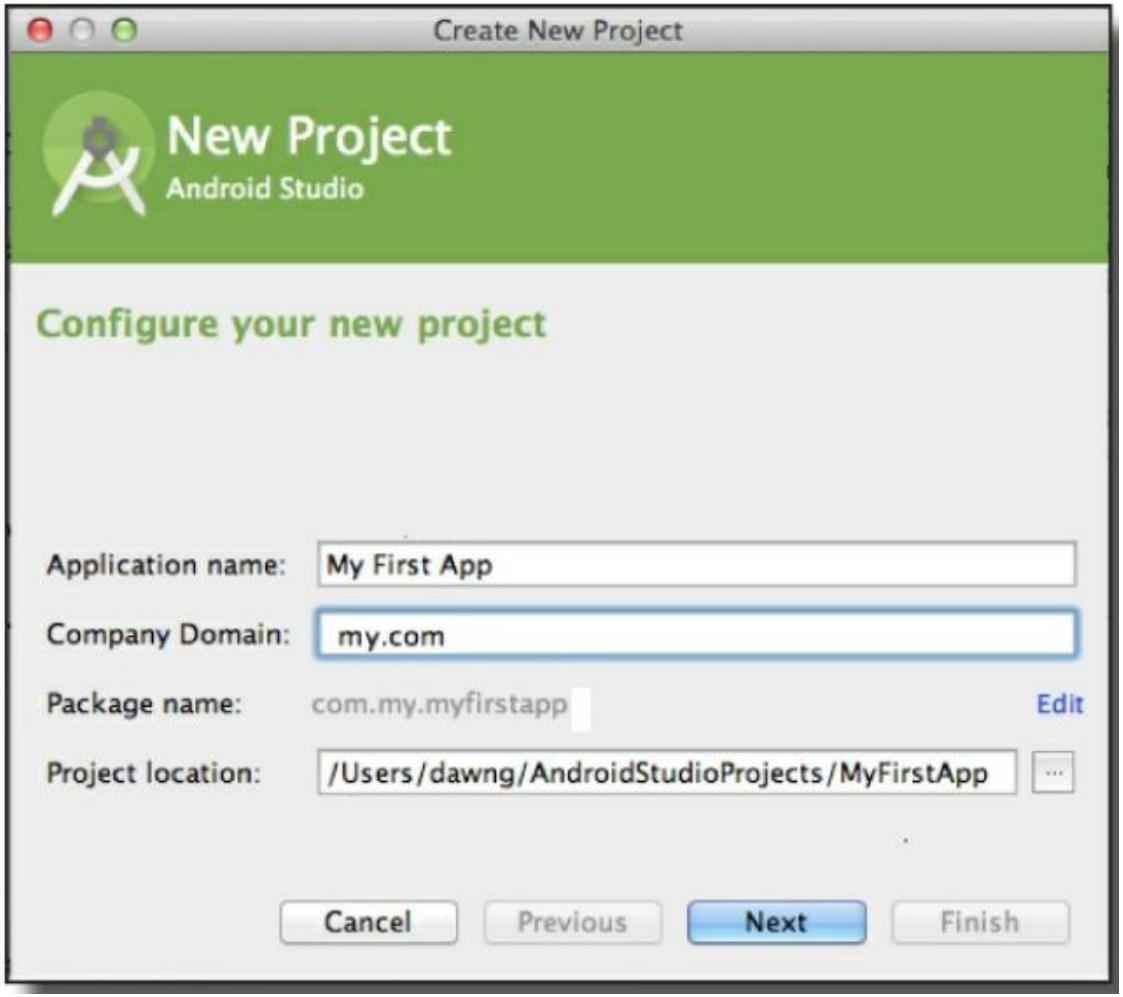
Build a First Basic App

1. Open Android Studio



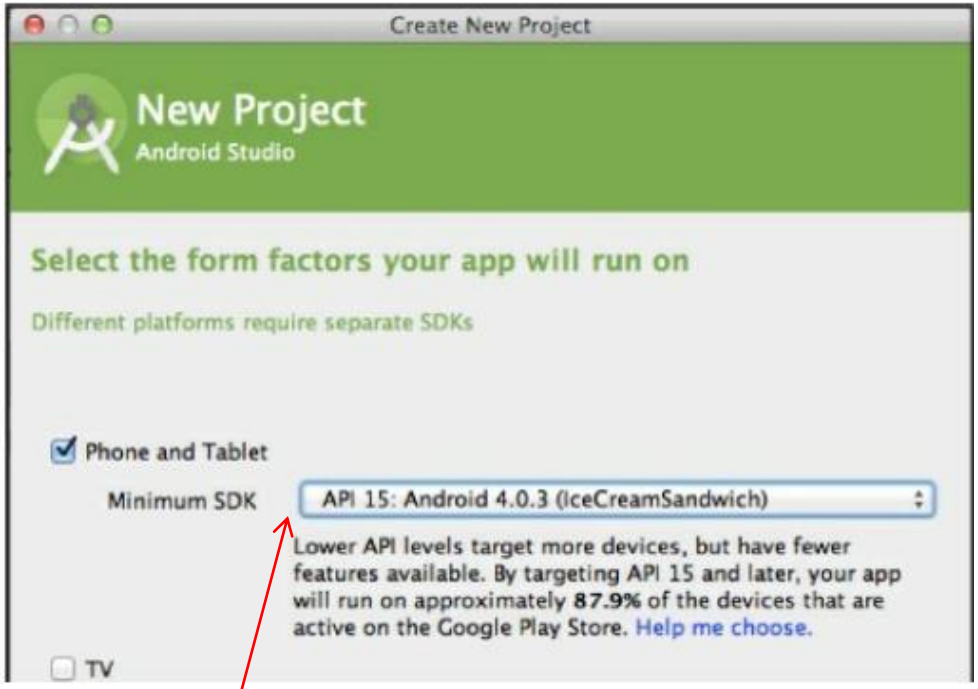
2. Start a new Android Studio Project

3. Enter an application name of “My First App”, a company name of “my.com”, and accept/change the default project location.



package name = combination(comp.domain, app name)

4. Specify the API level

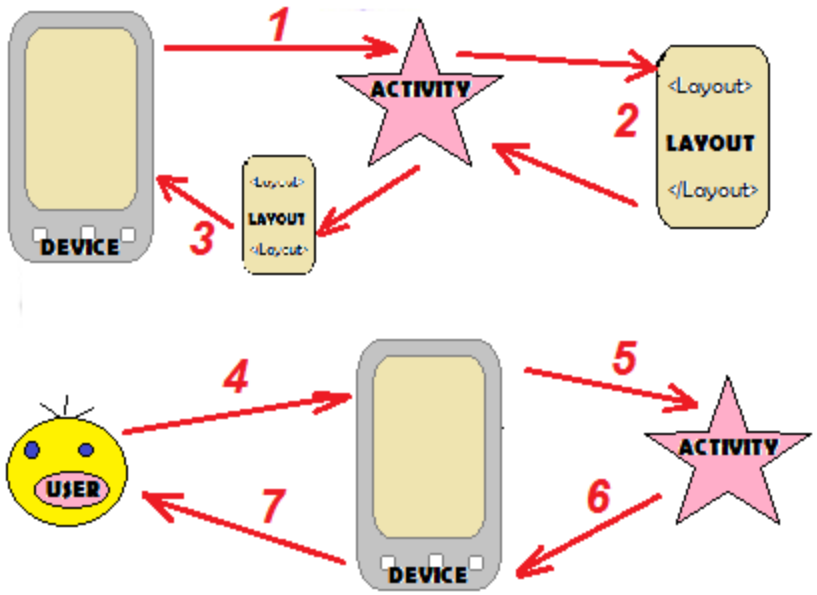


The app will run only on devices with specified API level or higher

Version	Codename	API level
1.0		1
1.1		2
1.5	Cupcake	3
1.6	Donut	4
2.0	Eclair	5
2.01	Eclair	6
2.1	Eclair	7
2.2.x	Froyo	8
2.3 - 2.3.2	Gingerbread	9
2.3.2 - 2.3.7	Gingerbread	10
3.0	Honeycomb	11
3.1	Honeycomb	12
3.2	Honeycomb	13
4.0 - 4.0.2	Ice Cream Sandwich	14
4.0.3-4.0.4	Ice Cream Sandwich	15
4.1	Jelly Bean	16
4.2	Jelly Bean	17
4.3	Jelly Bean	18
4.4	KitKat	19
4.4	KitKat (with wearable extensions)	20
5.0	Lollipop	21

5. Specify Layouts and Activities

- Layouts define how the user interface is presented.
- Activities define actions.



1. **The device launches your app and creates an activity object.**

2. **The activity object specifies a layout.**

3. **The activity tells Android to display the layout on screen.**

4. **The user interacts with the layout that's displayed on the device.**

5. **The activity responds to these interactions by running application code.**

6. **The activity updates the display...**

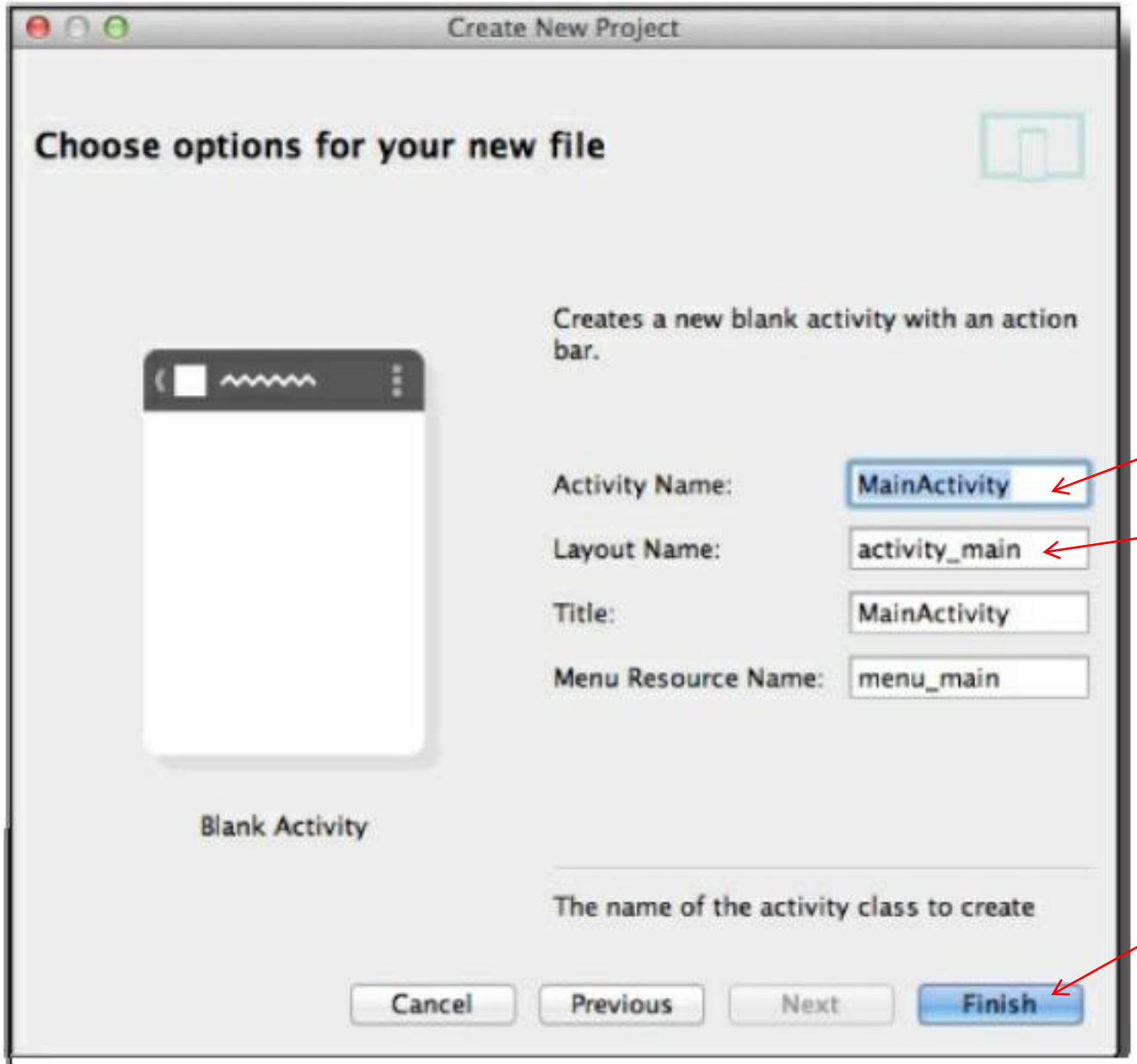
7. **...which the user sees on the device.**

5.1 Create an Activity

create an app with a basic activity

Select Blank Activity option

5.2 Configure the Activity

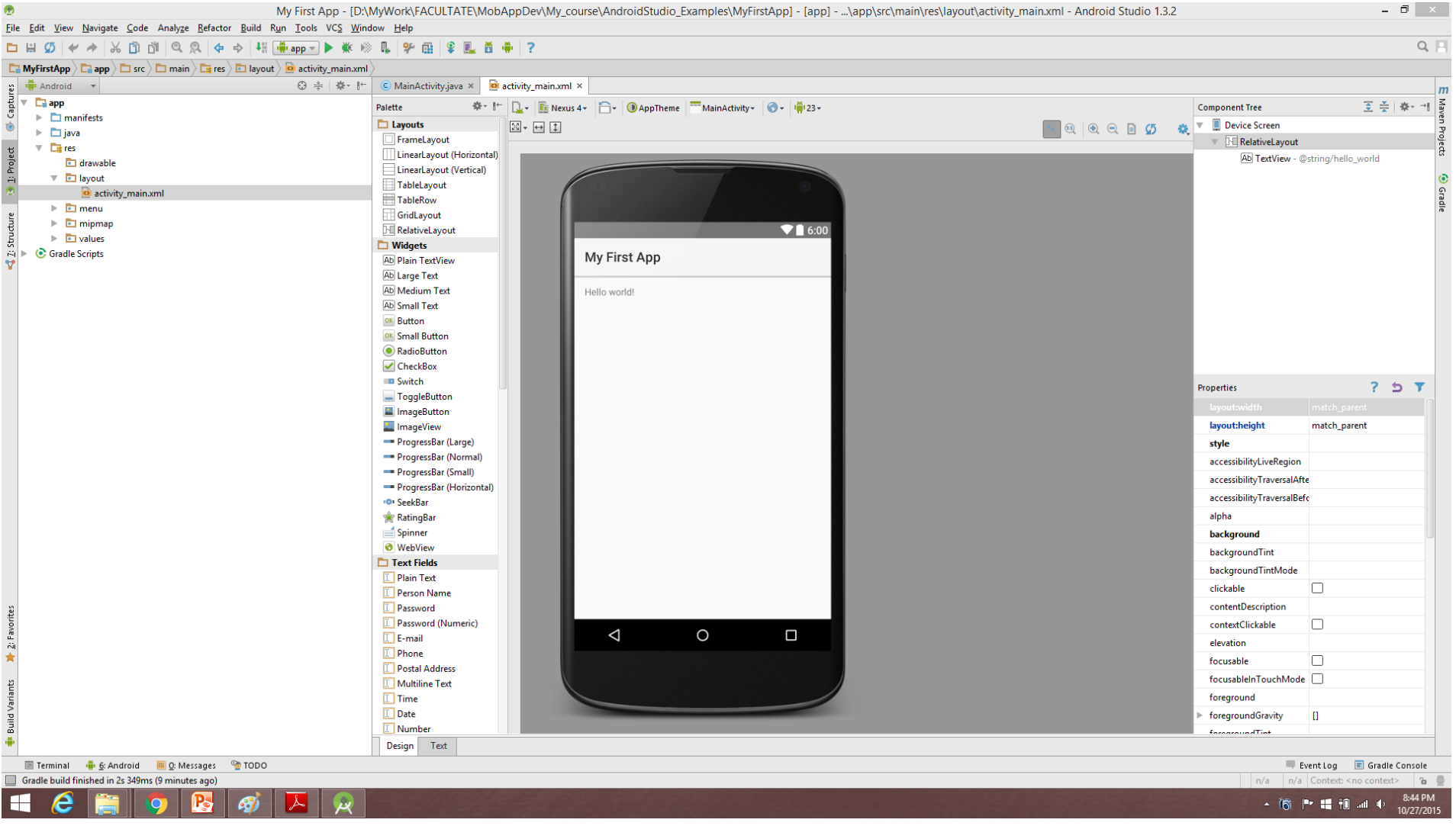


MainActivity.java class file

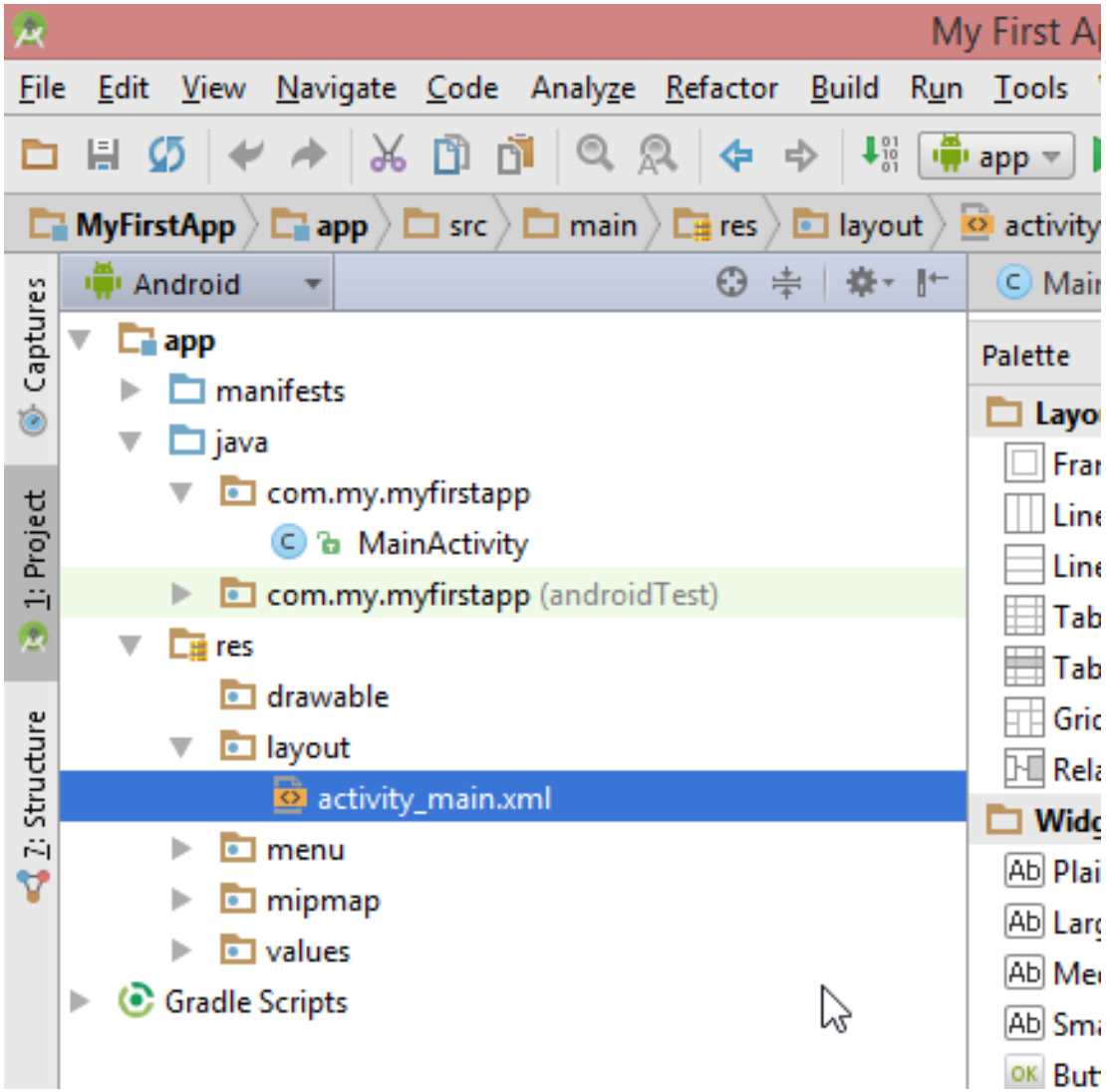
activity_main.xml XML file

Android Studio will build the app.

Review: **The Android Studio wizard created a project for MyFirstApp, configured to some specifications. It created a basic activity and layout with template code, with sample "Hello world!" text in the layout.**



Android Studio creates a complete folder structure for MyFirstApp Project that includes different types of files (with collapse and expand function)



Android Studio creates a complete folder structure for MyFirstApp. For example:

-the *root* folder has the same name as project;

-the *build* folder contains files that Android Studio creates for you. You don't usually edit anything in this folder;

-the *source* folder contains source code you write and edit;

-every Android project needs a file called *R.java*, which is created for you and it lives in the generated folders. Android uses it to help it keep track of resources in the app;

-the *java* folder contains any Java code you write. Any activities you create live here;

-the *res* folder contains system resources. The *layout* folder contains layouts and the *values* folder contains resource files for values such as strings;

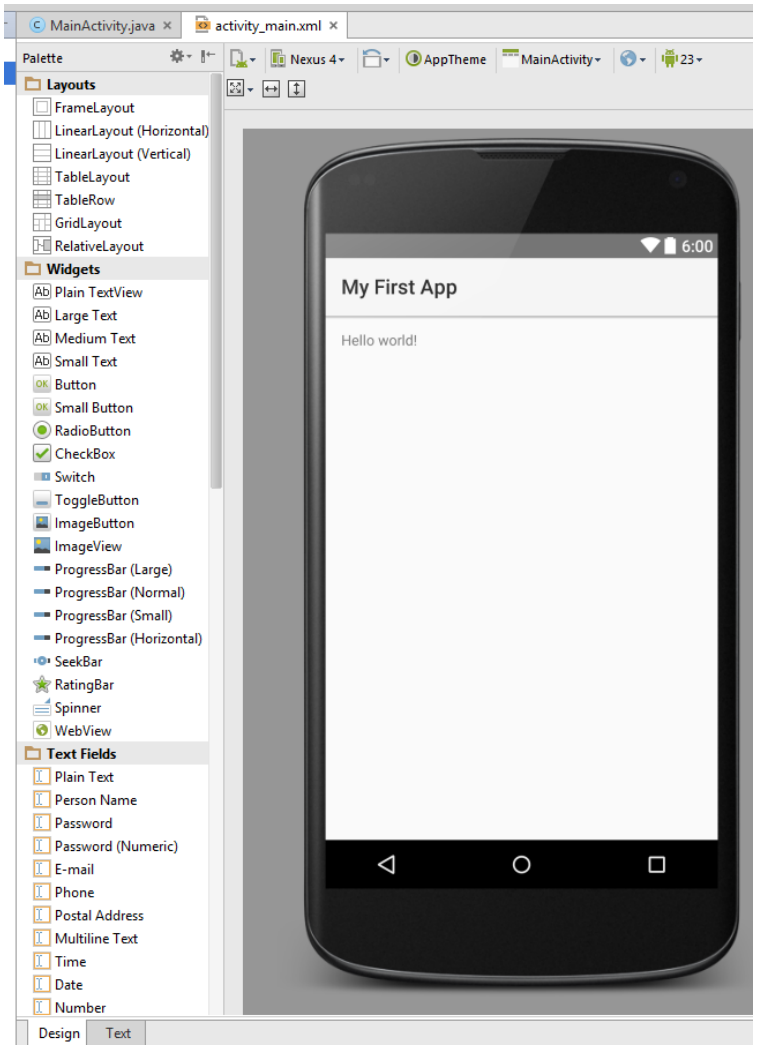
-*MainActivity.java* defines an activity that tells Android how the app should interact with the user;

-*activity_main.xml* defines a layout that tells Android how the app should look;

-every app must include *AndroidManifest.xml* file that contains essential informations about the app (what components it contains, required library and other declarations);

-*string.xml* file contains strings such as app name and any defaults text values;

An important feature for layouts editing: Code Editor vs Design Editor



Understanding *activity_main.xml*

```
<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:paddingLeft="@dimen/activity_horizontal_margin“
         android:paddingRight="@dimen/activity_horizontal_margin"
         android:paddingTop="@dimen/activity_vertical_margin"
         android:paddingBottom="@dimen/activity_vertical_margin"
        tools:context=".MainActivity">

   <TextView
      android:text="@string/hello_world"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

</RelativeLayout>
```

Make the layout the same width and height as the screen size on the device.

Add padding to the screen margins.

Include a `TextView` GUI component for displaying text.

Display the text value of a string resource called `hello_world`.

Make the text wrap horizontally and vertically.

Understanding *MainActivity.java*

```
package com.my.myfirstapp;

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

public class MainActivity extends Activity {

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

package name.

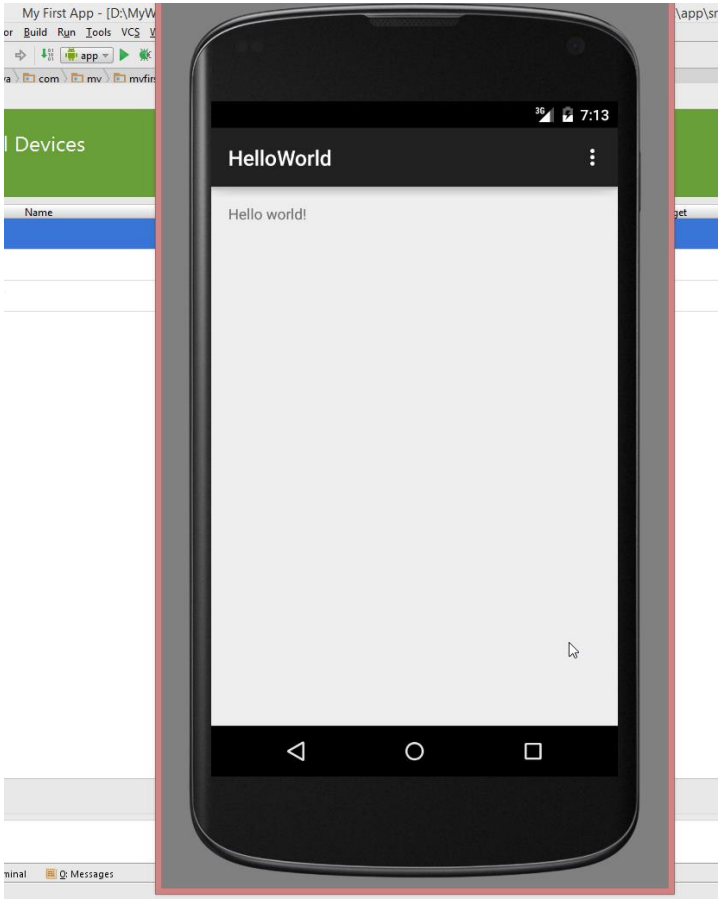
Android classes used in `MainActivity`.

`MainActivity` extends the Android class `android.app.Activity`.

Implement the `onCreate()` method from the `Activity` class. This method is called when the activity is first created.

Specifies which layout to use.

Run MyFirstApp in the Android emulator : allows to run app on an Android virtual device (AVD). The AVD behaves just like a physical Android device. Thus, can be set up numerous AVDs, each emulating a different type of device.



The emulator is an application.

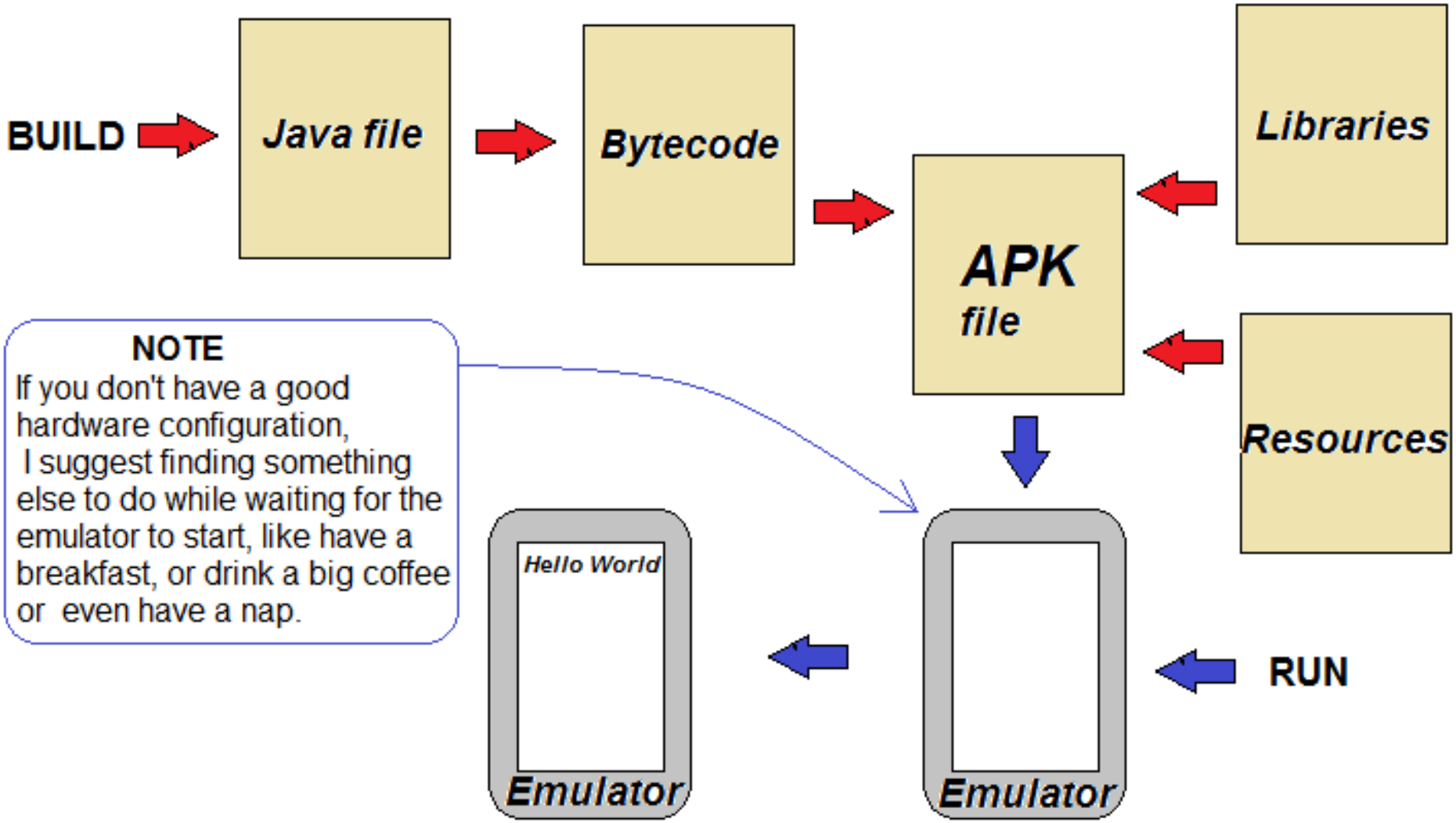
It looks just like a phone running on your computer.

Thus, it recreates the exact hardware environment of an Android device (CPU, memory, sound chips, video display etc)

It is built on an existing emulator called QEMU (similar to other virtual machine like VirtualBox).

Understanding the Android emulator *: Compile, Package, Deploy, Run*

An APK file is an Android application package. It's basically a JAR or ZIP file for Android app.



? Why the Android Emulator is so slooooooow ? An Android Emulator uses an open source application called QEMU (or Quick Emulator) to emulate the entire Android hardware device, so it has to do a lot of work for each operation in part.

! On the other side, the iPhone Simulator runs much faster because all of the code for iOS is compiled to run natively on the Mac and the iPhone Simulator runs at Mac-native speed.

? How to speed up Android development ?

1. Use a real device: *"Developer options" and check the "Stay Awake" option.*

2. Use an emulator snapshot: *Tools→Android→AVD Manager→Edit AVD and check the "Store a snapshot for faster startup" option.* (improve booting Emulator using a copy of memory)

3. Use Host GPU: *Tools→Android→AVD Manager→Edit AVD and check the "Use host GPU" option.* (by using PC's graphic card for OpenGL)

4. Use hardware acceleration: On(ly) Intel x86 CPU machine, the Emulator can run the Android machine code instructions directly on Intel CPU using HAXM (Intel's Hardware Accelerated Execution Manager) HAXM is a hypervisor ⇔ switch CPU into a special mode to run virtual machine instructions directly

Note: HAXM should be installed and will only run on Intel processors that support Intel Virtualization Technology

Lab & Homework

Based on source code provided to students, add new features to app.

Ta-Ta for now!